"H A S H I N G"

VISÃO GERAL E ESTUDO PARA MÁQUINAS COM MEMÓRIA VIRTUAL PAGINADA

Hélio de Menezes Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DO CENTRO DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE FEDERAL DA PARAÍBA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.).

COMISSÃO EXAMINADORA:

Orion de Oliveira Silva

ORION DE OLIVEIRA SILVA - M.Sc.
Presidente e Orientador

Bruno Correia da N. Queiroz

BRUNO CORREIA DA N. QUEIROZ - M.Sc.

ADRIAN JAMES WEERHEM - M. Math.

CAMPINA GRANDE
ESTADO DA PARAÍBA - BRASIL
MAIO - 1976

S586h Silva, Hélio de Menezes.
"Hashing" : visão geral e estudo para máquinas com memória virtual paginada / Hélio de Menezes Silva. - Campina Grande, 1976.
138 f.

Dissertação (Mestrado em Ciências) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1976.
"Orientação : Prof. M.Sc. Orion de Oliveira Silva".
Referências.

1. Hashing - Chaves de Dados. 2. Método de Hash. 3. Transformação Chave-Endereço. 4. Máquinas - Memória Virtual Paginada. 5. Dissertação - Ciências. I. Silva, Orion de Oliveira. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.713(043)

"HASHING": VISAO GERAL E ESTUDO PARA MAQUINAS COM MEMORIA VIRTUAL PAGINADA.

DEDICO ESTE TRABALHO A:

MINHA MAE : MARIA DO CARMO;

MINHA ESPOSA : VALDENIRA ;

MEUS FILHOS : CARMEN RAQUEL,
SANDRA,
MAURO E
SERGIO.

AGRADECIMENTOS

AGRADECO A :

PROF. VALDENIRA NUNES DE MENEZES SILVA, MINHA ESPOSA , SEM O QUAL INESTIMAVEL AUXILIO, APOIO, CONFORTO E PACIENCIA , MEU TRABALHO NAO PODERIA TER SIDO FEITO;

PROF. ORION DE OLIVEIRA SILVA, MEU ORIENTADOR, PELA CONSTANTE PACIENCIA, DEDICACAO E AMIZADE NA ARDUA TAREFA DE ORIENTACAO, CRITICA E CORRECAO DESTE TRABALHO;

PROF. MARIO TOYOTARO HATTORI, PELO AUXILIO AMIGO QUE PRESTOU, TANTO NA CRITICA DOS MANUSCRITOS COMO PELO FRANQUEAMENTO DA SUA BIBLIOTECA PARTICULAR, SEM O QUE ESTE TRABALHO SERIA QUASE IMPRATICAVEL, NA EPOCA E PRAZO EM QUE FOI FEITO;

SRS. JOSE LEONIDAS MACIEL DA SILVA E FERNANDO MARQUES DE ARAUJO, PELO TRABALHO DE PERFURACAO;

SR. WILTON RAMOS DA SILVA, PELA EXECUCAO DOS DESENHOS.

RESUMO ("ABSTRACT")

RESUMO

INICIALMENTE E FEITO UM ESTUDO GERAL DOS MAIS VARIADOS ASPECTOS LIGADOS AO "HASHING", VISANDO OFERECER AO ANALISTA DE SISTEMAS TUDO O QUE ELE PRECISA SABER SOBRE O ASSUNTO. EM SEGUIDA E FEITA UMA ANALISE DO "HASHING" PARA MAQUINAS COM MEMORIA VIRTUAL PAGINADA, FAZENDO-SE INCLUSIVE UM ESTUDO EXPERIMENTAL-COMPARATIVO DE 4 SOLUCOES ALTERNATIVAS. SE BEM QUE ESSAS ALTERNATIVAS JA TIVESSEM SIDO VISUALIZADAS PELOS MAIS ESTUDIOSOS, FAZIA-SE CONVENIENTE UM ESTUDO DO MODO QUE FOI FEITO AQUI E QUE, AO MENOS NA BIBLIOGRAFIA CONSULTADA, AINDA NAO EXISTIA.

DIVERSOS ALGORITMOS SAO APRESENTADOS, A MAIORIA DELES SENDO CONSTITUIDA DE GENERALIZACOES DE ALGORITMOS JA DETALHADOS, MAS SO PARA "BUCKET SIZE" IGUAL A 1, O RESTANTE SENDO CONSTITUIDO DE ADAPTACOES, EXTENSOES E DETALHAMENTOS DE IDEIAS JA EXISTENTES.

ABSTRACT

INITIALLY, A SURVEY OF THE VARIOUS ASPECTS OF HASHING IS MADE, IN AN ATTEMPT TO PROVIDE THE SYSTEMS ANALYST WITH THE NECESSARY BACKGROUND. THEN, AN ANALYSIS OF HASHING IN PAGED VIRTUAL MACHINES IS PRESENTED, INCLUDING A COMPARATIVE, EXPERIMENTAL STUDY WITH FOUR ALTERNATIVE SOLUTIONS. EVEN THOUGH THESE ALTERNATIVES HAVE ALREADY BEEN DEVISED BY OTHERS, IT WAS DEEMED CONVENIENT TO PREPARE THIS STUDY AS THE APPROACH HEREIN WAS NOT FOUND IN ANY OF THE LITERATURE SURVEYED.

VARIOUS ALGORITHMS ARE PRESENTED, MOST ARE GENERALIZATIONS OF ALGORITHMS ALREADY GIVEN IN DETAIL FOR BUCKET SIZE EQUAL TO ONE. THE REST ARE ADAPTATIONS, EXTENSIONS AND DETAILING OF ALREADY EXISTING IDEAS.

I N D I C E

"MAS DE LA BUSCARAS AO SENHOR TEU DEUS, E O ACHARAS, QUANDO O BUSCARES DE TODO O TEU CORAÇÃO E DE TODA A TUA ALMA."

DEUTERONOMIO 4:29.

0 - ADVERTENCIAS. PALAVRAS CHA-VES.

0.1-ESCRITA DE NUMEROS

USAMOS AS REGRAS VIGENTES NOS PAISES DE LINGUA INGLESA PARA ESCRITA DE NUMEROS: O PONTO EM LUGAR DA NOSSA VIRGULA, E VICE-VERSA. JUSTIFICATIVA: GUARDAR HARMONIA ENTRE O QUE ESCREVEMOS E AS REPRODUCOES DE PROGRAMAS OU RELATORIOS OBTIDOS EM COMPUTADORES.

0.2-NOMENCLATURA

PROPOSITADAMENTE, AS VEZES CONSERVAMOS A NOMENCLATURA USUALMENTE ENCONTRADA NA LINGUA INGLESA, AS VEZES A TRADUZIMOS PARA A FORMA MAIS APROXIMADA EM PORTUGUES, E POR FIM, OUTRAS VEZES, NAO HESITAMOS EM ADOTAR NEOLOGISMOS, USANDO NOVAS PALAVRAS QUE ESTAO SENDO CRIADAS OU APORTUGUESADAS, CUJO USO JA ESTA SE CONSAGRANDO NO NOSSO MEIO PROFISSIONAL, E QUE SEM DUVIDA BREVEMENTE FIGURARAO NOS MELHORES DICIONARIOS DA NOSSA LINGUA.

APRESENTAMOS A SEGUIR UMA RELACAO PARCIAL DAS PALAVRAS INGLESAS MAIS COMUMENTE USADAS NO TEXTO, COM A TRADUCAO OU SIGNIFICADO EQUIVALENTE E QUE PODERIAM SER ADOTADOS, MESMO QUE COM ALGUMAS RESTRICOES. VER TAMBEM 1.2.

INDEX	:	INDEX OU INDICE.
HASH	:	TRANSFORMACAO CHAVE-ENDERECO.
HASHING	:	ATO DE TRANSFORMAR UMA CHAVE NUM ENDERECO.
SCATTER TABLE	:	TABELA (OU ARQUIVO) EM QUE O METODO DE ACESSO AOS REGISTROS E O "HASH".
BUCKET	:	REGISTRO FISICO (MAS NEM SEMPRE. VER 1.2).
SLOT	:	REGISTRO LOGICO.
HOME ADDRESS	:	ENDERECO DIRETAMENTE CORRESPONDENTE A UMA CHAVE EM QUE FIZEMOS UMA TRANSFORMACAO CHAVE-ENDERECO.
TO DELETE	:	SUPRIMIR, ELIMINAR, EXCLUIR, CONSIDERAR JA NAO MAIS VALIDO UM REGISTRO (OU ARQUIVO, OU PROGRAMA, ETC.), TORNANDO SEU ESPACO NOVAMEN-

TE DISPONIVEL E UTILIZAVEL PARA NOVAS APLICACOES. QUASE SEMPRE PREFERIMOS APORTUGUESAR, USANDO O VERBO "DELETAR", POIS TRATA-SE DE NEOLOGISMO AMPLAMENTE DIFUNDIDO EM NOSSOS MEIOS.

KEY	:	CHAVE OU IDENTIFICADOR
IDENTIFIER	:	CHAVE OU IDENTIFICADOR.
TO OVERFLOW	:	TRANSBORDAR, ULTRAPASSAR UM LIMITE SUPERIOR.
DATA	:	DADOS.
FILE	:	ARQUIVO, COLECAO RELATIVAMENTE GRANDE DE REGISTROS DO MESMO TIPO, USUALMENTE NA MEMORIA AUXILIAR.
TABLE	:	TABELA, COLECAO RELATIVAMENTE PEQUENA DE REGISTROS DO MESMO TIPO, USUALMENTE NA MEMORIA PRINCIPAL.

0.3-ADVERTENCIA AO LEITOR

EMBORA TENHAMOS PROCURADO TRATAR DE TODOS OS PROBLEMAS RELACIONADOS COM O "HASH", MESMO QUE NAO EXAUSTIVAMENTE, VISANDO QUE ESTA OBRA POSSA OFERECER EM PORTUGUES, COMO UM PEQUENO MANUAL, TUDO QUE UM ANALISTA PRECISA SABERSOBRE O ASSUNTO, O MODO COMO ESTE TRABALHO FOI ESCRITO TALVEZNAO O FACA IDEAL PARA UM PRIMEIRO CONTATO COM A MATERIA. ESPERAMOS, ASSIM, QUE QUEM O LEIA JA TENHA TIDO UM CONTATO INICIAL COM O TEMA, COMO ERA DE SE ESPERAR EM UMA TESE DE MESTRADO, A FIM DE SUPORTAR ALGUNS OCASIONAIS DESVIOS DA SEQUENCIA LOGICA IDEAL QUE UMA OBRA DIDATICA PARA NAO INICIADOS DEVERIA TER.

0.4-PALAVRAS CHAVE ("KEY-WORDS")

PESQUISA; "HASH", "HASHING", "SCATTER TABLE", TECNICAS DE RANDOMIZACAO, TRANSFORMACAO CHAVE-ENDERECO, FUNCAO DE "HASH"; CHAVE ("KEY"), IDENTIFICADOR ("IDENTIFIER"); SINONIMO, COLISAO, "HOME ADDRESS", "BUCKET OVERFLOW"; "INDEX", "SCATTER INDEX TABLE"; "MULTIKEY FILE"; "VIRTUAL SCATER TABLE", "VIRTUAL HASH ADDRESS"; PAGINA, MEMORIA PAGINADA, MEMORIA PAGINADA POR DEMANDA, MEMORIA VIRTUAL, CIRCULARIDADE DE ENDERECOS DENTRO DE PAGINAS.

0.5-ACENTOS, SINAIS, EXPRESSOES ARITMETICAS E FINS DE LINHAS.

ESTE TRABALHO FOI INTEIRAMENTE PERFURADO EM CARTOES, RAZAO DA FALTA DE ACENTOS E CERTOS SINAIS. NA INTERPRETACAO DE EXPRESSOES ARITMETICAS, LEVAR EM CONTA AS REGRAS DE FORTRAN. DESCULPAR TAMBEM AS INTERRUPCOES DE LINHAS PARA CONTINUACAO EM OUTRAS, ONDE FREQUENTEMENTE SEGUIMOS NEM AS BOAS REGRAS DA GRAMATICA, MAS SIM O USUAL EM PROGRAMACAO EM FORTRAN.

1-I N T R O D U C A O

ESTE TRABALHO PRETENDE FAZER UM ESTUDO GERAL DOS METODOS DE "HASH" EXISTENTES, PARA EM SEGUIDA FAZER UM ESTUDO MAIS APROFUNDADO DAS IMPLICACOES DO USO DE METODOS DE "HASH" EM MAQUINAS COM MEMORIA VIRTUAL PAGINADA.

1.1-AREAS DE APLICACAO DOS METODOS DE "HASH"

1.1.1-QUANDO PREFERIMOS UM METODO DE "HASH":

NAO HA RESPOSTA CATEGORICA E DEFINITIVA PARA ESTA QUESTAO. A ESCOLHA DE UMA TECNICA DE PESQUISA DEPENDE DUMA SERIE ENORME DE FATORES, O QUE PODE SER PERCEBIDO PELA LEITURA DE MARTIN (27), CAPITULO 21, ENTRE OUTRAS OBRAS.

NO ENTANTO, SENDO N O NUMERO DE REGISTROS DE UM ARQUIVO, COM UM METODO DE "HASH" NOS PODEMOS PESQUISAR O ARQUIVO NUM TEMPO QUE, PARA TODOS FINS PRATICOS, E INDEPENDENTE DO VALOR DE N; COM UM METODO DE PESQUISA BINARIA O TEMPO E PROPORCIONAL A LOG2(N); E COM UM METODO DE PESQUISA LINEAR O TEMPO E PROPORCIONAL A N.

ASSIM, GENERICAMENTE FALANDO, PODERIAMOS DIZER QUE, FREQUENTEMENTE, OS METODOS DE "HASH" VAO SETORNANDO CADA VEZ MAIS ATRAENTES, A MEDIDA QUE N CRESCE. "HASHING" PRESTA-SE MELHOR PARA PROCESSAMENTO EM TEMPO REAL OU "ON-LINE" DO QUE PARA PROCESSAMENTO EM LOTES ("BATCH"), ONDE ALGUMA ORDENACAO E QUASE SEMPRE REQUERIDA. A GRANDE DESVANTAGEM DO "HASHING" E QUE NAO HA NENHUMA ORDENACAO EM UMA TABELA-"HASH".

NOTEMOS QUE NEM TODA PESQUISA LINEAR OU BINARIA PODE SER SUBSTITUIDA POR UMA PESQUISA POR "HASH". USANDO UM EXEMPLO DE MAURER & LEWIS (30), SE TIVERMOS UM ARQUIVO DE CLIENTES E DESEJARMOS IMPRIMIR, PARA CADA UM DOS MESES DO ANO ANTERIOR, O NOME DO CLIENTE QUE GEROU A MAIOR ATIVIDADE DURANTE AQUELE MES, NOS DEVEREMOS ENTAO USAR UMA PESQUISA LINEAR, MUITO EMBORA NAO NECESSITEMOS FAZER DOZE PESQUISAS LINEARES SEPARADA-

MENTE. HA ALGUNS OUTROS CASOS EM QUE O USO DE UM METODO DE "HASH" NAO PODERIA (OU NAO DEVERIA) SER ADOTADO. NO ENTANTO, OS METODOS DE "HASH" SEMPRE PODEM SER USADOS PARA O MAIS COMUM DOS TIPOS DE PESQUISA, QUE E AQUELE EM QUE TENTAMOS ENCONTRAR, EM UM ARQUIVO, O REGISTRO QUE TEM O CONTEUDO DE UM PRE-FIXADO CAMPO IGUAL A UM VALOR DADO. EM 1.3 VEREMOS AS ALTERNATIVAS POSSIVEIS PARA O "HASHING", APONTANDO AS VANTAGENS E DESVANTAGENS DE CADA UMA DELAS EM RELACAO AO "HASH", DE MODO QUE TENHAMOS UMA ORIENTACAO PARA PONDERARMOS E DECIDIRMOS SE O "HASHING" E OU NAO A MELHOR ALTERNATIVA, EM VARIAS SITUACOES.

1.1.2-AS DUAS PRINCIPAIS AREAS DE APLICACAO DOS METODOS DE "HASH":

TAL COMO ALGUMAS DAS OUTRAS TECNICAS DE PESQUISA, AS TECNICAS DE "HASHING" TEM DUAS NOTAVEIS AREAS DE APLICACAO:

1. A AREA DE CONSTRUCAO DE COMPILADORES;
2. A AREA DE ADMINISTRACAO DE BANCO DE DADOS ("DATA BASE MANAGEMENT").

EXPLIQUEMOS MELHOR COMO AS COISAS SE PASSAM NESSAS DUAS AREAS:

1. CONSIDEREMOS A TABELA DE SIMBOLOS NUM MONTADOR, COMPILADOR, INTERPRETADOR OU TRADUTOR MAIS GERAL. TEMOS UM REGISTRO EM CORRESPONDENCIA A CADA IDENTIFICADOR NO PROGRAMA FONTE. DIGAMOS QUE O IDENTIFICADOR FIQUE ARMAZENADO NO PRIMEIRO CAMPO DO REGISTRO, E QUE AS OUTRAS INFORMACOES QUE DEVEM SER ASSOCIADAS A ESSE IDENTIFICADOR (ENDERECO, TIPO, DIMENSOES, ETC.), FIQUEM EM CAMPOS SUBSEQUENTES. CADA VEZ QUE UM IDENTIFICADOR E ENCONTRADO DURANTE O PROCESSAMENTO DO PROGRAMA FONTE PELO COMPILADOR, NOS DEVEMOS ENCONTRAR O REGISTRO CUJO CONTEUDO DO PRIMEIRO CAMPO SEJA IGUAL AO IDENTIFICADOR. SE UM TAL REGISTRO AINDA NAO EXISTE NA TABELA DE SIMBOLOS, UM NOVO REGISTRO CORRESPONDENTE SERA INSERIDO NESSA TABELA. USUALMENTE, TODOS OS REGISTROS FICARAO NA MEMORIA PRINCIPAL.
2. CONSIDEREMOS AGORA UM "DATA BASE", ONDE OS REGISTROS FICAM NORMALMENTE EM DISCOS, OU ALGUMA OUTRA FORMA DE MEMORIA AUXILIAR DIRETAMENTE ENDERECAVEL. EM CADA ARQUIVO, OS REGISTROS TEM UM CAMPO QUE IDENTIFICA-OS UNIVOCAMENTE, E QUE SERA CHAMADO 'CHAVE' OU 'IDENTIFICADOR' ("KEY" OU "IDENTIFIER"). ESTE CAMPO PODERA SER, POR EXEMPLO, O NUMERO DO EMPREGADO, O NUMERO DE UMA PECA, O NUMERO E A DATA DE VOO DE UMA GRANDE COMPANHIA DE AVIACAO, O NUMERO DE CARTEIRA DE IDENTIDADE, O NUMERO DE LICENCA DO VEICULO, E ASSIM POR DIANTE. SEMPRE QUE UMA TRANSACAO REALIZAR-SE NOS DEVEREMOS ENCONTRAR, NA MEMORIA SECUNDARIA, AQUELE REGISTRO

QUE TEM, COMO CONTEUDO DO CAMPO IDENTIFICADOR, UM VALOR FORNECIDO.

VEJAMOS AGORA AS PECULIARIDADES DAS DUAS PRINCIPAIS AREAS DE APLICACAO DOS METODOS DE "HASH":

1. NA AREA DE CONSTRUCAO DE COMPILADORES, AS TABELAS DE SIMBOLOS SAO RELATIVAMENTE PEQUENAS, CONTENDO TIPICAMENTE NA ORDEM DE 100 A 1,000 ENTRADAS, CADA UMA DELAS COM DE 10 A 50 CARACTERES DE COMPRIMENTO. ESSAS TABELAS SAO GERALMENTE ARMAZENADAS E MANIPULADAS INTERNAMENTE, NA MEMORIA PRINCIPAL, ONDE O TEMPO DE ACESSO E MEDIDO EM TERMOS DE NANO SEGUNDOS (1NS = 1.0E-09 SEGUNDO). DEVIDO A ESTA VELOCIDADE NO ACESSO DE DADOS, UM DOS OBJETIVOS PRIMORDIAIS NA SELECAO DE UMA ESTRATEGIA DE PESQUISA PARA UMA TABELA DE SIMBOLOS SERA MINIMIZAR O NUMERO MEDIO DE INSTRUCOES DE MAQUINA EXECUTADAS NO PROCESSO DE ARMAZENAMENTO OU RECUPERACAO DE UMA ENTRADA NA TABELA.
2. NA AREA DE ADMINISTRACAO DE "DATA BASES", TIPICAMENTE ENCONTRAMOS DEZENAS OU CENTENAS DE ARQUIVOS FORMANDO UM "DATA BASE", CADA ARQUIVO COM 1,000 A 1,000,000 DE REGISTROS, CADA REGISTRO COM DE 10 A 10,000 CARACTERES DE COMPRIMENTO. OBVIAMENTE, UM TAL CONJUNTO DE DADOS E POR DEMAIS GRANDE PARA RESIDIR NA MEMORIA PRINCIPAL, DEVENDO SER PARTICIONADO EM 'BLOCOS' DE MEMORIA SECUNDARIA (POR EXEMPLO : PAGINAS E TRILHAS) E RECUPERADO EM SEGMENTOS, A MEDIDA QUE FOREM REQUERIDOS. O ACESSO A DISPOSITIVOS DE MEMORIA SECUNDARIA ENVOLVE TIPICAMENTE O MOVIMENTO DE SENSORES FISICOS, OU SEJA, A UTILIZACAO DE MECANISMOS ELETRO-MECANICOS. O TEMPO DE RECUPERACAO E MEDIDO EM TERMOS DE MILISEGUNDOS (1MS = 0.001 SEGUNDO). PORISSO, UM DOS OBJETIVOS PRIMORDIAIS NA SELECAO DE UMA TECNICA DE PESQUISA NUM "DATA BASE" E MINIMIZAR O NUMERO MEDIO DE ACESSOS A MEMORIA SECUNDARIA. OUTRA PECULIARIDADE QUE DIFERENCIA UM "DATA BASE" DUMA TABELA DE SIMBOLOS E QUE UM "DATA BASE" MANTEM NORMALMENTE CONJUNTOS DE INTERCONEXOES FISICAS ENTRE OS REGISTROS, AS QUAIS DEVEM COEXISTIR COM QUALQUER QUE SEJA A TECNICA USADA PARA LOCALIZAR REGISTROS INDIVIDUAIS.

1.2-CONCEITOS GERAIS. TERMINOLOGIA BASICA. HISTORICO.

1.2.1-REGISTRO LOGICO ("LOGICAL RECORD, NODE OR SEGMENT"):

E UMA COLECAO ROTULADA DE CAMPOS ("DATA ITEMS" OU "FIELDS") RELATIVOS A UMA SO ENTIDADE. E A QUANTIDADE BASICA DE DADOS QUE PASSA 'DE' E 'PARA' UM PROGRAMA DE APLICACAO, SOB O CONTROLE DO "SOFTWARE" DE ADMINISTRACAO DUM "DATA BASE".

1.2.2-REGISTRO FISICO ("PHYSICAL RECORD", BLOCO OU "BLOCK"):

E UMA UNIDADE ISOLADA DE DADOS TRANSMITIDA DE OU PARA UM VOLUME ("DISK", "DISK PACK", TAMBOR, ETC.). EM OUTRAS PALAVRAS, E A UNIDADE BASICA DE DADOS QUE E LIDA OU ESCRITA POR UM UNICO COMANDO DE ENTRADA/SAIDA. EXEMPLO: DADOS ENTRE OS VAZIOS DEMARCADORES DE ENDERECOS ("INTER-RECORD GAPS") NUM DISCO. UM REGISTRO FISICO FREQUENTEMENTE CONTEM VARIOS REGISTROS LOGICOS.

1.2.3-"BUCKET" (BALDE, DEPOSITO, "POCKET"):

E UMA AREA CONTENDO UM GRUPO DE REGISTROS LOGICOS, NESTE CASO CHAMADOS "SLOTS", QUE SAO ENDERECADOS COMO UM SO CONJUNTO. O "BUCKET" PODE SER UM REGISTRO FISICO, UMA TRILHA, OU UMA CELULA, MAS FREQUENTEMENTE E UM AGRUPAMENTO DETERMINADO POR UMA TECNICA DE ENDERECAMENTO TAL COMO "HASHING", NAO NECESSARIAMENTE RELACIONADO COM O "HARDWARE". USUALMENTE, DENTRO DE UM "BUCKET", A PESQUISA DE UM "SLOT" E FEITA SEQUENCIMENTE; ADEMAIS, O "BUCKET" CONFUNDE-SE COM O REGISTRO FISICO.

1.2.4-"SLOT" (RANHURA, FENDA, REGISTRO LOGICO):

E QUALQUER UM DOS REGISTROS LOGICOS QUE COMPOEM UM "BUCKET".

1.2.5-TAMANHO (CAPACIDADE) DE "BUCKET" ("BUCKET SIZE"):

E O NUMERO DE REGISTROS LOGICOS ("SLOTS") QUE COMPOEM UM REGISTRO FISICO. COMO UM CASO ESPECIAL, PODEMOS TER O "BUCKET SIZE" IGUAL A 1, QUANDO UM REGISTRO FISICO COINCIDE COM UM REGISTRO LOGICO.

1.2.6-CHAVE OU IDENTIFICADOR ("KEY" OU "IDENTIFIER"):

E O CAMPO DE UM REGISTRO QUE O IDENTIFICA COMPLETA E UNI - VOCAMENTE. POR EXEMPLO, O IDENTIFICADOR DOS REGISTROS DE UM ARQUIVO QUE DESCREVE OS EMPREGADOS DE UMA EMPRESA PODE SER TANTO O CAMPO QUE CONTEM O NUMERO DE MATRICULA, COMO O CAMPO QUE CONTEM O NOME DO EMPREGADO. OBSERVEMOS QUE, POR HIPOTESE, EM TODO UM ARQUIVO NAO PODEM EXISTIR DUAS ENTIDADES DIFEREN - TES COM A MESMA CHAVE. ASSIM, NO EXEMPLO DADO, NAO DEVEMOS TER DOIS EMPREGADOS DISTINTOS COM O MESMO NUMERO DE MATRICU - LA, CASO A CHAVE ESCOLHIDA TENHA SIDO ESTE CAMPO.

1.2.7- DENSIDADE DE IDENTIFICADORES:

FISICAMENTE UM IDENTIFICADOR E REPRESENTADO POR UMA SE - QUENCIA DE CARACTERES DE UM VOCABULARIO DE CODIFICACAO. PARA UM CONJUNTO DE IDENTIFICADORES DE COMPRIMENTO FIXO LI ("LENG- TH OF IDENTIFIERS") E VOCABULARIO (UM CONJUNTO) DE TAMANHO V ("VOCABULARY"), EXISTEM PI = V**LI "POSSIBLE IDENTIFIERS". E- XEMPLO: USANDO-SE AS V = 26 LETRAS DO ALFABETO PARA FORMARMOS IDENTIFICADORES COM LI = 2 LETRAS, TEMOS PI = 26**2 IDENTIFI- DORES POSSIVEIS, OU SEJA, AA, AB, AC, ..., ZZ. SE APENAS N DESSES IDENTIFICADORES POSSIVEIS FOREM REALMENTE USADOS, DE - FINIMOS A DENSIDADE DE IDENTIFICADORES ("DENSITY OF IDENTI - FIERS") COMO

$$DI = N/PI = N/V^{**}LI \ .$$

ASSIM, NO "DATA BASE" DA FIGURA 3.1, TEMOS:

- COMPRIMENTO DO IDENTIFICADOR:
  LI = 2, PORQUE O NUMERO DO EMPREGADO TEM 2 DIGITOS;
- TAMANHO DO VOCABULARIO:
  V = 10, PORQUE O VOCABULARIO E 0,1,2,...,9;
- NUMERO DE IDENTIFICADORES REALMENTE USADOS:
  N = 15, PORQUE TEMOS 15 IDENTIFICADORES ATIVOS;
- DENSIDADE DE IDENTIFICADORES:
  DI = N/PI = N/V**LI = 15/10**2 = 0.15.

1.2.8-FATOR DE CARGA (FATOR DE CARREGAMENTO, "PACKING" OU "LOADING FACTOR") DUM ARQUIVO:

E A RELACAO ENTRE O NUMERO DE IDENTIFICADORES ATIVOS E O NUMERO TOTAL DE REGISTROS LOGICOS DO ARQUIVO:

$$P = \frac{N}{M} = \frac{\text{NUMERO DE REG. LOG. ATIVOS NO ARQUIVO}}{\text{NUMERO TOTAL DE REG. LOG. DO ARQUIVO}}$$

ASSIM, NA FIGURA 3.1, TEMOS P = 15/20 = 0.75.

COMO VEREMOS NO CAPITULO 3, TEMOS VARIOS METODOS PARA RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOW", MAS PODEMOS CLASSIFICAR ESSES METODOS EM DOIS TIPOS: METODOS QUE USAM, E METODOS QUE NAO USAM UMA AREA DE "OVERFLOW" EM SEPARADO. QUANDO USAMOS UMA AREA EM SEPARADO, ESPECIFICAMENTE PARA "OVERFLOWS", TEMOS NECESSIDADE DE DISTINGUIR ENTRE DOIS FATORES DE CARGA:

FATOR DE CARGA PRIMARIO: NA FORMULA ACIMA, M = NUMERO TOTAL DE REGISTROS LOGICOS DO ARQUIVO, EXCLUSIVE AQUELES DA AREA DE "OVERFLOW";

FATOR DE CARGA GLOBAL, OU, SIMPLESMENTE, FATOR DE CARGA : NA FORMULA ACIMA, M = NUMERO TOTAL DE REGISTROS LOGICOS DO ARQUIVO, INCLUSIVE AQUELES DA AREA DE "OVERFLOW".

1.2.9-IDEIA CENTRAL DO "HASH" ("HASHING", TRANSFORMACAO CHAVE-ENDERECO, "KEY-TO-ADDRESS TRANSFORMATION", TECNICA DE RANDOMIZACAO, "HASH CODE TECHNIQUE", "SCATTER STORAGE TABLE", "HASH TABLE"):

DADO UM IDENTIFICADOR ARBITRARIO, I, APLICAMOS UMA TRANSFORMACAO T (UNIVOCA) SOBRE O MESMO, OBTENDO-SE O ENDERECO T(I) QUE SERA USADO PARA ARMAZENAR O REGISTRO DE IDENTIFICADOR I NUM DASD ("DIRECT ACCESS STORAGE DEVICE"). SE POSTERIORMENTE QUISERMOS PESQUISAR O REGISTRO DE CHAVE I, BASTA CALCULARMOS T(I) NOVAMENTE. ISTO E O QUE TORNA O "HASHING" TAO CONVIDATIVO: NA MAIORIA DAS VEZES PODEMOS ENCONTRAR O REGISTRO DESEJADO IMEDIATAMENTE, SEM QUAISQUER COMPARACAO REPETIDA COM OUTROS ITENS.

1.2.10-FUNCAO DE "HASHING" ("HASHING FUNCTION", TRANSFORMACAO DE CHAVE OU IDENTIFICADOR EM ENDERECO, "KEY-TO-ADDRESS TRANSFORMATION"):

REALIZA UMA APLICACAO (MAPEAMENTO) DO ESPACO DOS IDENTIFICADORES POSSIVEIS NO ESPACO DOS ENDERECOS DE "BUCKETS".

1.2.11-FUNCAO DE "HASHING" UNIFORME:

SE UMA CHAVE SELECIONADA ALEATORIAMENTE TEM UMA IGUAL PROBABILIDADE DE SER APLICADA (MAPEADA) EM QUALQUER ENDERECO DE "BUCKET", ENTAO ESSA TRANSFORMACAO E DITA SER UNIFORME, POR QUE A DISTRIBUICAO DAS CHAVES PELA TRANSFORMACAO E UNIFORME .

1.2.12-SINONIMOS:

SENDO T A TRANSFORMACAO (FUNCAO) DE "HASH" ADOTADA, DIZEMOS QUE DOIS IDENTIFICADORES DISTINTOS I1 'DIFERENTE DE' I2 SAO SINONIMOS SE T(I1) = T(I2), ISTO E, SE SAO IGUAIS OS VALORES DA FUNCAO DE "HASH" APLICADA SOBRE I1 E I2.

1.2.13-COLISAO; ENDERECO INICIAL OU "HOME ADDRESS":

OS SINONIMOS ACIMA SAO DITOS COLIDENTES NO "HOME ADDRESS" OU ENDERECO INICIAL T(I1) = T(I2).

1.2.14-FUNCAO DE ENDERECAMENTO DIRETO:

AS VEZES E POSSIVEL ATRIBUIRMOS OS VALORES DAS CHAVES DE MODO QUE CADA VALOR DE CHAVE POSSA TAMBEM SER TOMADO DIRETAMENTE COMO ENDERECO DO REGISTRO CORRESPONDENTE. DIZEMOS, ENTAO, QUE TEMOS UMA FUNCAO DE ENDERECAMENTO DIRETO, ONDE NAO OCORREM COLISOES. UMA FUNCAO DE ENDERECAMENTO DIRETO PODERA SER SEMPRE CONSTRUIDA SE O ARQUIVO TEM UM "SLOT" (REGISTRO LOGICO) PARA CADA UM DOS IDENTIFICADORES POSSIVEIS. O FATOR DE CARGA, NESTE CASO, SERA IGUAL A DENSIDADE DE IDENTIFICADORES. ESSA SITUACAO PODE OU NAO SER UMA SITUACAO QUE OCORRA NA PRATICA, SENDO MUITAS VEZES UTOPICA. POR EXEMPLO, SE TEMOS APENAS 100 ALUNOS POSSIVEIS NUMA DETERMINADA DISCIPLINA E PODEMOS DAR A ESSES ALUNOS NUMEROS DE MATRICULA DE 00 A 99, ENTAO PODEMOS TOMAR COMO ENDERECO DE UM REGISTRO REFERENTE A UM ALUNO, SIMPLESMENTE A SUA PROPRIA CHAVE, OBTENDO-SE UM DESEMPENHO ESPETACULAR PARA UM NUMERO REAL DE ALUNOS NA FAIXA, DIGAMOS, DE 70 A 100; SE TEMOS APENAS 100 ALUNOS POSSIVEIS NA DISCIPLINA, MAS SEUS NUMEROS DE MATRICULA 'TEM' QUE VARIAR DE 0 A 9,999, NAO PODEMOS USAR ESTE METODO, QUE EQUIVALERIA A TOMARMOS 10,000 REGISTROS NA MEMORIA, PARA REALMENTE SO UTILIZARMOS 100. VER 1.3.6.

1.2.15-"HASHING" SEM COLISOES:

TEORICAMENTE, SE O ARQUIVO FOR ESTATICO (NAO SUJEITO A INSERCOES E DELECOES), COM AS CHAVES PERTINENTES TODAS A UM DOMINIO FIXO, PRE-CONHECIDO E RELATIVAMENTE PEQUENO, PODERIAMOS ENCONTRAR UMA FUNCAO DE "HASH" QUE GARANTISSE A NAO EXISTENCIA DE COLISOES PARA QUAISQUER DESSAS CHAVES. NO ENTANTO,

ALEM DA DETERMINACAO DESSA FUNCAO CONSTITUIR-SE NUM TREMENDO E TRABALHOSO QUEBRA-CABECA, A SITUACAO ACIMA CONSIDERADA E QUASE IMPOSSIVEL DE OCORRER NA VIDA REAL. NA PRATICA, AS CO - LISOES SAO INEVITAVEIS, PORQUE NORMALMENTE O NUMERO DE CHA - VES POSSIVEIS E DUMA ORDEM DE MAGNITUDE MUITAS VEZES MAIOR DO QUE O NUMERO DE "HOME ADDRESSES" POSSIVEIS. DE UM MODO GERAL, TENTAR PROCURAR FUNCOES DE "HASH" QUE NAO GEREM COLISOES E UM QUEBRA-CABECA QUE NAO COMPENSA O ESFORCO, PRINCIPALMENTE SE O ARQUIVO E DINAMICO, SITUACAO QUASE QUE UNIVERSAL. E PREFERI - VEL USARMOS UMA FUNCAO JA CONSAGRADA E UM DOS METODOS DE RE - SOLUCAO DE COLISOES TAMBEM JA CONSAGRADOS, E QUE MELHOR SE A- DAPTEM AO CASO PARTICULAR.

1.2.16-VANTAGENS DO "HASHING":

O "HASHING" PRESTA-SE MUITO BEM PARA PESQUISAS EM "DIRECT ACCESS STORAGE DEVICES" TAIS COMO DISCOS OU TAMBORES.

QUANDO O NUMERO DE ""BUCKETS" (E ASSIM O NUMERO DE REGIS - TROS) E REALMENTE MUITO GRANDE, O "HASHING", BEM UTILIZADO , SUPERA A TODO E QUALQUER OUTRO METODO DE PESQUISA, AO MENOS QUANTO A VELOCIDADE MEDIA DE ACESSO A UM REGISTRO QUALQUER , ISOLADAMENTE. ASSIM, O "HASHING" E BASTANTE ACONSELHAVEL QUANDO NOSSO PROCESSAMENTO E EXCLUSIVA (OU PREPONDERANTEMEN - TE) EM TEMPO REAL OU "ON LINE".

EMBORA O "HASHING" SO SEJA IDEAL PARA CERTAS SITUACOES DE PESQUISA BASTANTE SIMPLES, ONDE AS CHAVES A SEREM PESQUISADAS SURGEM NUMA SEQUENCIA ALEATORIA, ESTAS SITUACOES SAO BASTAN - TE COMUMENTE OCORRENTES PARA DAREM AO "HASH" UMA GRANDE IM - PORTANCIA.

1.2.17-DESVANTAGENS DO "HASHING":

O PRINCIPAL PROBLEMA COM O "HASHING" E QUE NAO HA NENHUMA ORDEM NUMA TABELA "HASH". APOS UMA PESQUISA MAL SUCEDIDA NU- MA "SCATTER TABLE", A UNICA COISA QUE SABEMOS E QUE A CHAVE I PESQUISADA NAO ESTA NA TABELA. ENQUANTO ISSO, METODOS DE PES- QUISA BASEADOS EM COMPARACOES SEMPRE DAO MAIS INFORMACOES, TAIS COMO QUAL E A MAIOR CHAVE <= I, QUAL E A MENOR CHAVE >= I, OU QUAIS SAO E ONDE ESTAO AS CHAVES ENTRE DOIS LIMITES DA- DOS, I1 E I2; METODOS BASEADOS EM COMPARACOES, COMO OS DE PESQUISA EM ARVORE BINARIA, TORNAM MAIS FACIL O PROCESSAMEN - TO OU SAIDA DOS REGISTROS DA TABELA EM ORDEM CRESCENTE (OU DECRESCENTE), SEM NECESSIDADE DE FAZERMOS UM "SORTING" SEPA- RADAMENTE, SENDO MELHORES PARA A MAIORIA DOS PROCESSAMENTOS EM LOTES (EM "BATCH").

AS VEZES E DIFICIL PREVERMOS O TAMANHO DE UMA "SCATTER TA-

BLE". UMA VEZ ESCOLHIDO ESTE VALOR ELE SERA FIXO. SE TIVER SIDO MUITO GRANDE, ESTAREMOS DESPERDICANDO MEMORIA, E SE FOR NECESSARIO AUMENTAR O TAMANHO DUMA "SCATTER TABLE" DURANTE UM "RUN", TODAS AS ENTIDADES COM CHAVES ATIVAS NA TABELA DEVERAO TER SEUS "HASH CODES" RECALCULADOS, SENDO DEPOIS ARMAZENADOS EM NOVAS LOCALIZACOES DE UMA NOVA "SCATTER TABLE", MAIOR QUE A ANTIGA. F. R. HOPGOOD, SEGUNDO KNUTH (20), SUGERIU NO .... "COMPUTER BULLETIN" 11, EM 1968, FAZERMOS UM NOVO "HASH" NA TABELA QUANDO A MESMA CHEGASSE A UM CERTO FATOR DE CARGA PO , SUBSTITUINDO O TAMANHO M DA TABELA POR DO*M. ESCOLHAS ADEQUADAS DESSES PARAMETROS PODEM SER FEITAS COM BASE EM 3.6 E NAS CARACTERISTICAS DOS DADOS, DE MODO QUE O PONTO CRITICO, NO QUAL SERA ECONOMICO FAZER O NOVO "HASH", PODE SER DETERMINADO.

EMBORA PROVEMOS QUE UM CERTO "HASHING" TEM UM DESEMPENHO MEDIO AGRADAVELMENTE BOM, O PIOR DESEMPENHO POSSIVEL ("WORST CASE") E AS VEZES TERRIVELMENTE INACEITAVEL EM CERTAS APLICACOES EM TEMPO REAL, TAIS COMO CONTROLE DE TRAFEGO AEREO, ONDE VIDAS DE PESSOAS ESTAO EM JOGO. NESTE CASO, PODERA SER PREFERIVEL ADOTARMOS UM METODO DE DESEMPENHO MEDIO UM POUCO PIOR, MAS QUE TENHA UM LIMITE ACEITAVEL PARA O "WORST POSSI - BLE CASE".

EM 1.3 VEREMOS AS ALTERNATIVAS POSSIVEIS PARA O "HASHING", BEM COMO ORIENTACOES PARA DECIDIRMOS SE O "HASH" E OU NAO APLICAVEL, EM VARIAS SITUACOES.

1.2.18-PEQUENO HISTORICO:

EMBORA TENHAMOS PRINCIPIADO A USAR "HASH" EM MEADOS DA DECADA DE 50, SO APOS 1968 COMECAMOS A USAR ESTE TERMO, HOJE CONSAGRADO.

QUEM PRIMEIRO ESTUDOU O "HASHING" FOI UM GRUPO DA IBM, POREM QUE NADA PUBLICOU (VER KNUTH (20), PAGINAS 540 A 541). AS PRIMEIRAS PUBLICACOES DEVEM-SE, INDEPENDENTEMENTE, A DUMEY (11) E PETERSON(37).APARENTEMENTE, O TERMO "SINONIMO" DEVE-SE A SCHAY & RAVER (41). DAS VARIAS "SURVEYS" EXISTENTES SOBRE METODOS DE "HASH", AS MELHORES SAO AS DE BUCHHOLZ (7) (EMBORA A MAIS ANTIGA E RESTRITA A METODOS DE ENDERECAMENTO DE ARQUIVOS) E A DE MAURER & LEWIS (30). VER TAMBEM MORRIS (33) , KNUTH (20) E SEVERANCE (43). UM METODO BASTANTE INTERESSANTE DE SE RECALCULAR OS "HASH CODES" QUANDO O TAMANHO DUMA TABELA DE "HASH" E EXPANDIDO E DADO POR BAYS (3). UM OTIMO HISTORICO SOBRE "HASH" E ENCONTRADO EM KNUTH (20).

1.3-ALTERNATIVAS POSSIVEIS PARA O "HASHING"

NATURALMENTE, EXISTEM MUITAS SITUACOES DE PESQUISA EM QUE O "HASH" NAO PODE OU NAO DEVE SER USADO, POR SER DESVANTAJOSO. APRESENTAREMOS AQUI, SUPERFICIALMENTE, UMA RELACAO PARCIAL DE ALGUNS DOS MAIS IMPORTANTES PROCEDIMENTOS DE PESQUISA QUE PODEM SER ADOTADOS EM SUBSTITUICAO AO "HASH". NOSSA PREOCUPACAO SERA TAO SOMENTE COMPARAR CADA UM DESSES METODOS ALTERNATIVOS COM O "HASHING", E NAO COMPARA-LOS ENTRE SI, POIS ISTO JA FOI MUITO BEM FEITO EM "SURVEYS" COMO AS DE SEVERANCE (43) E NIEVERGELT (35). LEMBREMOS : A GRANDE DESVANTAGEM DO "HASHING" E QUE NAO EXISTE ORDEM NUMA TABELA-"HASH"; A GRANDE VANTAGEM E A FACILIDADE PARA UM ACESSO ALEATORIO.

1.3.1-PESQUISA BINARIA:

CONSERVAMOS O ARQUIVO SEQUENCIAL E ORDENADO, PESQUISANDO UMA CHAVE PELA DIVISAO SUCESSIVA DO ARQUIVO EM METADES E DETERMINACAO DE EM QUAL METADE A CHAVE DADA ESTA. ESTE METODO E BOM PARA ARQUIVOS ESTATICOS OU QUASE ESTATICOS, ISTO E, QUE NAO MUDAM, OU SO O FAZEM MUITO RARAMENTE.

UMA PESQUISA BINARIA, NUM ARQUIVO COM N REGISTROS, NECESSITA EM MEDIA LOG2(N)-1 E NO MAXIMO LOG2(N)+1 COMPARACOES, SENDO PORISSO CHAMADA TAMBEM DE PESQUISA LOGARITMICA. OS CICLOS DE COMPARACAO SAO CURTOS, DE MODO QUE ESTE METODO E COMPARAVEL COM MUITOS DOS METODOS DE "HASHING" (EMBORA NAO COM OS MAIS RAPIDOS DELES), MESMO PARA UMA TABELA RAZOAVELMENTE GRANDE, PELO MENOS ENQUANTO ELA POSSA PERMANECER NA MEMORIA PRINCIPAL.

VANTAGEM SOBRE O "HASH": PERMITIR DE UMA MANEIRA SIMPLES E IMEDIATA IMPRESSOES DO ARQUIVO ORDENADAMENTE, CAPACIDADE MUITAS VEZES FREQUENTEMENTE REQUERIDA EM CERTOS ARQUIVOS CUJAS APLICACOES SO REQUEREM PROCESSAMENTO EM LOTES (EM "BATCH"), E IMPOSSIVEL DE SER ATENDIDA SE ADOTARMOS UMA FUNCAO "HASH" (SALVO SE, SEPARADAMENTE, USARMOS UM PROCESSO DE "SORTING").

DESVANTAGEM: A PESQUISA BINARIA SO E IDEAL PARA ARQUIVOS ESTATICOS OU QUASE ESTATICOS. SEJA N O NUMERO DE REGISTROS DO ARQUIVO INICIAL E N' O NUMERO DE REGISTROS A SEREM INSERIDOS (DELETADOS). UMA VEZ QUE O ARQUIVO DEVE PERMANECER ORDENADO, UMA INSERCAO (DELECAO) TOMA EM MEDIA N/2 TRANSFERENCIAS DE DADOS PARA A POSICAO SUCESSORA (ANTECESSORA) DE CADA REGISTRO A SER REMANEJADO. NO ENTANTO, UM GRUPO DE REGISTROS

QUE JA ESTEJA ORDENADO PODE SER INSERIDO (DELETADO) POR UM PROCESSO DE UNIAO ("MERGE") QUE NECESSITA N+N' (N-N') TRANSFERENCIAS DE REGISTROS PARA UM ARQUIVO CÓPIA. ASSIM, SE O ARQUIVO E RAZOAVELMENTE DINAMICO, O TEMPO GASTO EM MANTER-SE O ARQUIVO SEQUENCIAL E ORDENADO PODE TORNAR-SE COMPLETAMENTE PROIBITIVO. ADEMAIS, ESTE METODO NAO SE ADAPTA BEM PARA GRANDES ARQUIVOS NA MEMORIA SECUNDARIA, POIS UM ARQUIVO COM 2**10 =1,024 REGISTROS JA NECESSITA UMA MEDIA DE 9 ACESSOS POR PESQUISA.

PARA MAIORES DETALHES, VER KNUTH (20).

1.3.2-PESQUISA EM ARVORE BINARIA:

UMA ARVORE BINARIA E CONSTRUIDA COMO NA FIGURA 1.1. CADA REGISTRO TEM UM 'APONTADOR ESQUERDO' PARA UMA 'SUB-ARVORE ESQUERDA' E UM 'APONTADOR DIREITO' PARA UMA 'SUB-ARVORE DIREITA'. CADA CHAVE NOS REGISTROS DA SUB-ARVORE ESQUERDA (DIREITA) E MENOR (MAIOR) QUE A CHAVE DO REGISTRO QUE APONTA PA-

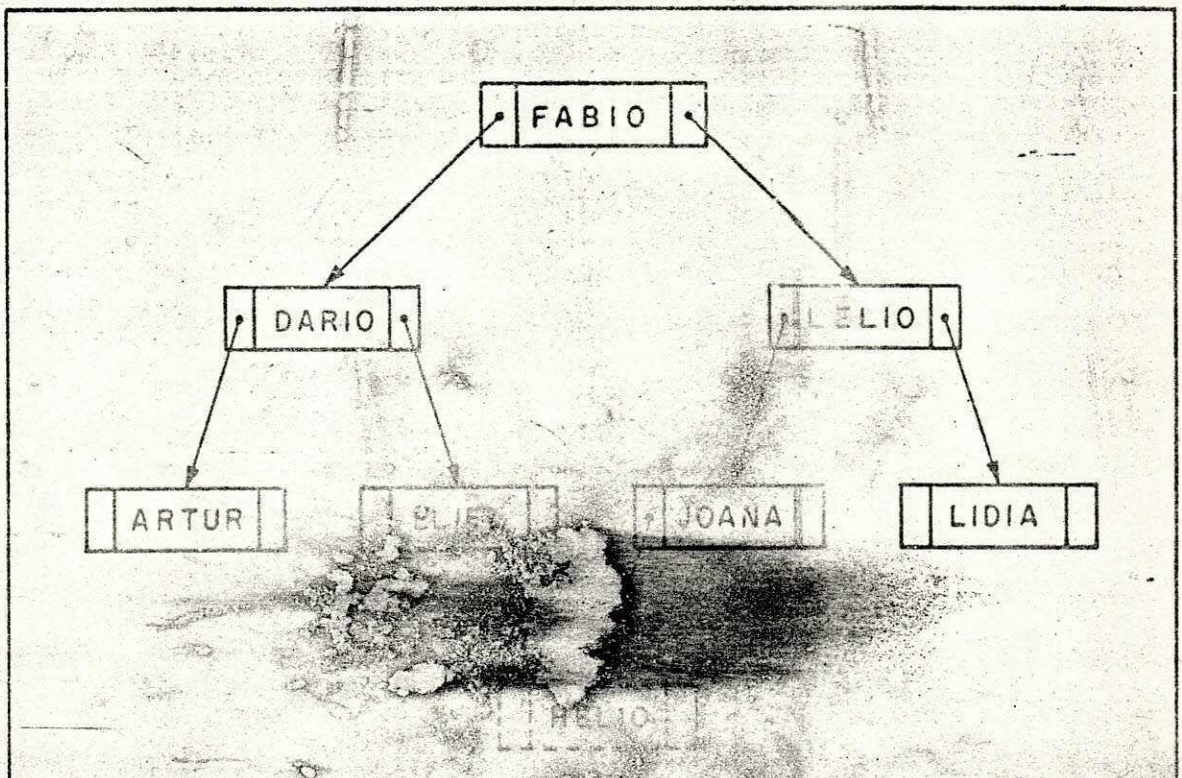


Fig.1.1 - Pesquisa em Árvore Binária

RA ESSA SUB-ARVORE. A PESQUISA DESENVOLVE-SE NO SENTIDO DO TOPO PARA A BASE DA FIGURA, NECESSITANDO NO MAXIMO LOG2(N) +1 COMPARACOES, DESDE QUE A ARVORE DE N REGISTROS ESTEJA BALANCEADA (ISTO E, O COMPRIMENTO DE CADA SUB-ARVORE ESQUERDA E EQUIVALHA AO DA SUB-ARVORE DIREITA CORRESPONDENTE).

ESTE METODO, COMPARADO COM O ANTERIOR, CONTINUA PERMITINDO PROCESSAMENTO SEQUENCIAL COM RELATIVA FACILIDADE, PERCORRENDO-SE ORDENADAMENTE A ARVORE PELO USO DE ALGORITMOS DE CAMINHAMENTO CENTRAL, TENDO POREM A VANTAGEM DE TER INSERCOES (E DELECOES) ENORMEMENTE FACILITADAS, COMO INDICADO EM LINHAS PONTILHADAS NA FIGURA 1.1, POIS ENVOLVEM APENAS POUQUISSIMAS ATUALIZACOES DE APONTADORES, SEM A NECESSIDADE DE COPIAR OU DESLOCAR PARTES ENORMES DO ARQUIVO. TEM POREM A DESVANTAGEM DE NECESSITAR, ADICIONALMENTE, ESPACO PARA OS 2*N APONTADORES.

VANTAGEM SOBRE O "HASH": PERMITIR, DE UMA MANEIRA AINDA RAZOAVELMENTE SIMPLES, A IMPRESSAO DO ARQUIVO ORDENADO.

DESVANTAGEM: MESMO QUE USEMOS ALGORITMOS SOFISTICADOS VOLTADOS PARA A CONSTRUCAO DE ARVORES BINARIAS NA MEMORIA EXTERNA, ESTE METODO, COMO O ANTERIOR, NAO SE PRESTA MUITO BEM PARA APLICACOES NA MEMORIA SECUNDARIA, PRINCIPALMENTE SE O ARQUIVO FOR RAZOAVELMENTE GRANDE. ADEMAIS, TEMOS O PERIGO DE INSERCOES FREQUENTES DESBALANCEAREM COMPLETAMENTE A ARVORE , NECESSITANDO PORTANTO DE ALGORITMOS QUE CUIDEM DE MANTER A ARVORE BALANCEADA.

PARA MAIORES DETALHES, VER KNUTH (20), NIEVERGEL (35) E FOSTER (12), QUE TRATAM TAMBEM DE IMPORTANTISSIMAS VARIACOES EM TORNO DO TEMA DE ARVORES.

1.3.3-ORGANIZACAO INDEXADA SEQUENCIAL:

A IDEIA BASICA POR TRAS DESTE METODO E UMA VARIACAO DA PESQUISA EM ARVORE BINARIA, NA QUAL TEMOS POUCOS NIVEIS (QUASE NUNCA MAIS QUE 4), COM CADA NODO QUE NAO SEJA FOLHA PODENDO TER UM NUMERO K (MUITISSIMO MAIOR QUE 2) DE APONTADORES , TAIS QUE, PARA 1=<I<J=<K, CADA CHAVE NA I(ESIMA) SUB-ARVORE , EM UM NIVEL QUALQUER DADO, SEJA MENOR QUE CADA CHAVE NA J(ESIMA) SUB-ARVORE DAQUELE NIVEL. NA IBM, AS FOLHAS SAO OS REGISTROS DO ARQUIVO E FICAM SEQUENCIAIS E ORDENADAMENTE, SENDO APONTADAS PELOS INDICES DE TRILHA, QUE POR SUA VEZ SAO APONTADOS PELOS INDICES DE CILINDRO E ESTES PELO INDICE MESTRE (OU DE UNIDADES), SE EXISTIR.

VANTAGEM SOBRE O "HASH": ALIAMOS AS VANTAGENS (SEM AS DESVANTAGENS) DOS METODOS DE PESQUISA BINARIA E POR "HASH". COM EFEITO, UM ARQUIVO ORGANIZADO NA FORMA INDEXADA SEQUENCIAL ASSEMELHA-SE A ORGANIZACAO SEQUENCIAL ORDENADA, PELA POSSIBILIDADE DE RAPIDO PROCESSAMENTO SEQUENCIAL, E ASSEMELHA-SE AO METODO DE "HASH" PELA POSSIBILIDADE DE UMA RAPIDA LOCALIZACAO

DE REGISTROS INDIVIDUAIS, PARA PROCESSAMENTO NAO SEQUENCIAL, ATRAVES DE REFERENCIAS A INDICES ASSOCIADOS AO ARQUIVO, QUASE NUNCA NECESSITANDO MAIS QUE 4 ACESSOS A MEMORIA SECUNDARIA. ALEM DISSO, UMA AREA SEPARADA DO ARQUIVO PRINCIPAL E MANTIDA PARA INSERCOES, ELIMINANDO ASSIM A NECESSIDADE DE COPIARMOS O ARQUIVO, COMO FARIAMOS NO CASO DE O ARQUIVO SER ORGANIZADO SEQUENCIAL E ORDENADAMENTE.

DESVANTAGEM: EM ALGUMAS APLICACOES, MESMO O PEQUENO NUMERO DE 3 OU 4 ACESSOS A MEMORIA SECUNDARIA NAO PODE SER TOLERADO, E, ADEMAIS, O PROCESSAMENTO SEQUENCIAL NAO E IMPORTANTE. NESTES CASOS, PREFERIMOS USAR ALGUM DOS MELHORES METODOS DE "HASH" EXISTENTES, QUE NECESSITAM, EM MEDIA, APENAS POUCO MAIS QUE 1 ACESSO A MEMORIA SECUNDARIA, POR PESQUISA DE CHAVE. O TRATAMENTO DE INSERCOES E DELECOES NUMA ORGANIZACAO INDEXADA SEQUENCIAL E MAIS COMPLICADO DO QUE AQUELE QUE TEMOS COM O "HASHING".

VER MAIORES DETALHES EM IBM (17), POR EXEMPLO.

1.3.4-METODO DA CHAVE DISTRIBUIDA:

TAMBEM ESTE METODO PODE SER CONSIDERADO COMO UMA VARIANTE EM TORNO DO TEMA DA PESQUISA BINARIA, SENDO USADO MAIS COMUMENTE QUANDO AS CHAVES SAO CADEIAS DE CARACTERES.

A FIGURA 1.2 MOSTRA UMA ESTRUTURA TIPICA DE ARVORE CORRESPONDENTE A ESTE METODO. EM CADA NIVEL, O NUMERO DE APONTADORES EM CADA REGISTRO E VARIAVEL, E CADA REGISTRO CONTEM NAO A CHAVE, MAS APENAS UM CARACTER DA MESMA. A PESQUISA E FEITA INICIANDO-SE NA RAIZ E CAMINHANDO PELO RAMO CORRESPONDENTE AO PRIMEIRO CARACTER DA CHAVE, DEPOIS PELO RAMO CORRESPONDENTE AO SEGUNDO CARACTER, E ASSIM POR DIANTE.

VANTAGEM SOBRE O "HASH": PERMITIR PROCESSAMENTO SEQUENCIAL. ESTE METODO TEM ALGUNS ASPECTOS POSITIVOS QUASE QUE APENAS NO CASO PARTICULAR DAS CHAVES SEREM CADEIAS DE CARACTERES NAO MUITO GRANDES, PODENDO PERMANECEREM, TODAS ELAS, NA MEMORIA PRINCIPAL. DEVERA HAVER, NO NODO DA TABELA ASSOCIADO AO ULTIMO CARACTER DUMA CHAVE, UM APONTADOR PARA O REGISTRO, NA MEMORIA SECUNDARIA, QUE CONTENHA OUTRAS INFORMACOES CORRESPONDENTES A ESSA CHAVE.

DESVANTAGEM: SE AS CHAVES FOREM CADEIAS RELATIVAMENTE GRANDES DE CARACTERES, ESTE METODO, PRINCIPALMENTE SE AS CHAVES ESTIVEREM NA MEMORIA SECUNDARIA, REQUER UM NUMERO DE ACESSOS MUITAS VEZES PROIBITIVO, ALEM DE UM PONDERAVEL USO DE MEMORIA PARA OS MUITOS APONTADORES NECESSARIOS.

1.3.5-PESQUISA LINEAR:

NESTE TIPO DE PESQUISA, CADA UM DOS REGISTROS TEM QUE SER RECUPERADO E VERIFICADO, ATE QUE ENCONTREMOS UM VALOR DESEJADO, OU TENHAMOS CERTEZA DA SUA NAO EXISTENCIA. EVIDENTEMENTE, TRATA-SE DE UM METODO MUITO POBRE, MAS QUE AINDA ASSIM E USADO PARA TABELAS MUITO PEQUENAS NA MEMORIA PRINCIPAL (QUANDO NAO COMPENSARIA USARMOS ALGORITMOS MUITO SOFISTICADOS), OU PARA QUALQUER TIPO DE PESQUISA QUE NAO SEJA ESPECIFICAMENTE PROVIDA POR UMA OUTRA ESTRUTURA DE ARQUIVO QUE TENHA SIDO ADOTADA. COMO ILUSTRACAO, ISTO OCORRE FREQUENTEMENTE QUANDO NECESSITAMOS SABER QUAL O REGISTRO R TAL QUE UMA DADA FUNCAO F(R) TEM SEU VALOR MAXIMO (OU MINIMO) EM TODO O ARQUIVO. NUM SISTEMA DE "DATA BASE", USANDO UM EXEMPLO DE MAURER & LEWIS (30), PODEMOS DESEJAR SABER QUAL CLIENTE MAIS SOLICITOU DE UM DETERMINADO PRODUTO X NO ULTIMO ANO, QUAL DEPARTAMENTO GASTOU MAIOR QUANTIDADE DE PAPEL TIMBRADO, QUAL FORNECEDOR TEVE A

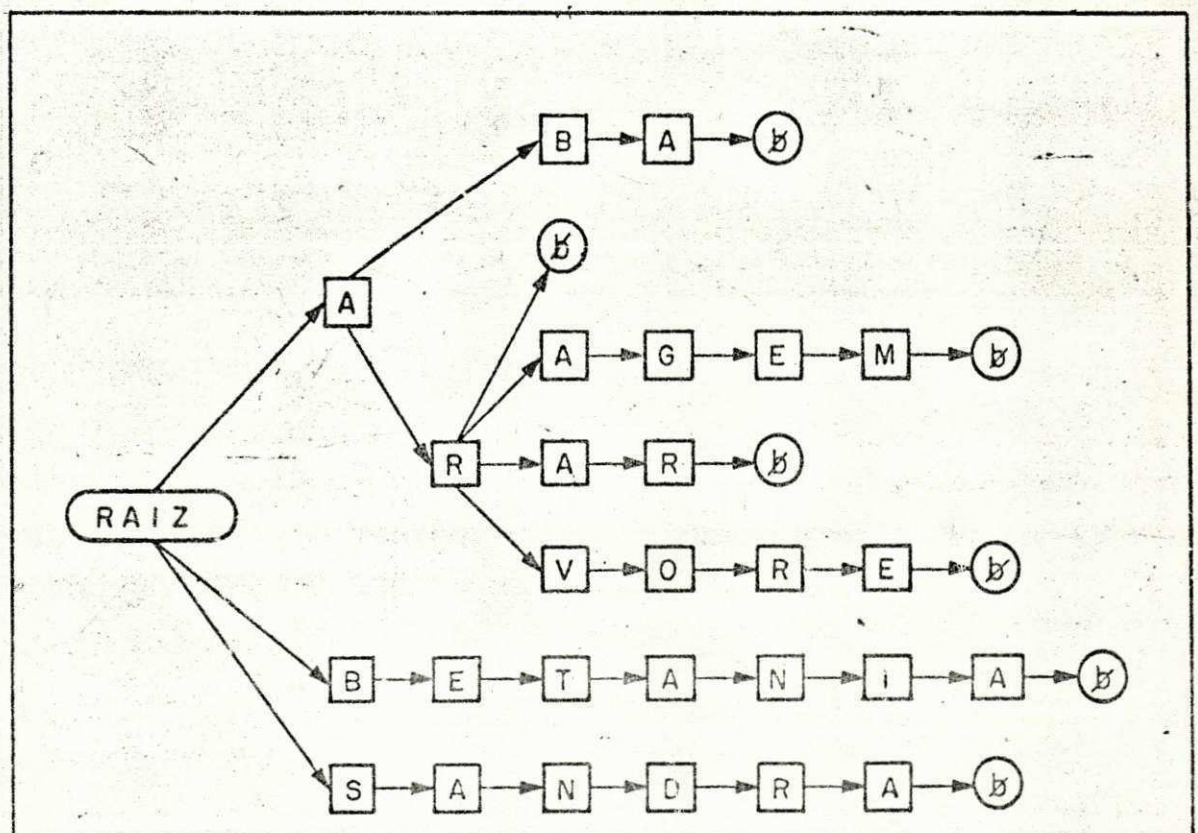


Fig. I.2 - O método da chave distribuída, para as chaves: ABA, AR, ARAGEM, ARAR, ARVORE, BETANIA e SANDRA.

MAIOR FALHA ("SLIPPAGE") TOTAL, OU QUAL VENDEDOR TEM O MELHOR REGISTRO EM ALGUM ITEM ESPECIFICO. ESTE TIPO DE PESQUISA SEMPRE NECESSITA DE TANTOS ACESSOS QUANTOS SAO OS REGISTROS DO ARQUIVO, A NAO SER QUE, SABENDO-SE DE ANTEMAO QUE NECESSITARIAMOS DE CERTAS INFORMACOES DESSE TIPO, TIVESSEMOS PREVISTO FACILIDADES PARA ISSO NO PROJETO DO ARQUIVO.

VANTAGENS SOBRE O "HASH": FACILIDADE DE PROGRAMACAO; CAPACIDADE DE RESPONDER PERGUNTAS TAIS COMO AS DO EXEMPLO ACIMA.

DESVANTAGEM: COMO O NUMERO DE ACESSOS E PROPORCIONAL AO NUMERO N DE REGISTROS NO ARQUIVO, A NAO SER PARA TABELAS MUITO PEQUENAS NA MEMORIA PRINCIPAL, ESTE METODO E TERRIVELMENTE INEFICIENTE, DEVENDO SEU USO SER NORMALMENTE EVITADO O MAIS POSSIVEL. DEVEMOS PREVER O MELHOR POSSIVEL A OCORRENCIA DE PERGUNTAS COMO AS DO EXEMPLO ACIMA, CRIANDO FACILIDADES NO PROGRAMA QUE EVITEM A PESQUISA LINEAR, SALVO EM SITUACOES NAO PREVISIVEIS OU DE OCORRENCIA EXTREMAMENTE RARA.

VER MAIORES DETALHES EM KNUTH (20).

1.3.6-METODOS DIRETOS:

ESTES METODOS SAO GERALMENTE USADOS, COM GRANDE VANTAGEM, QUANDO A ESCOLHA DOS VALORES DAS CHAVES ESTA SOB O NOSSO CONTROLE, PODENDO-SE ESCOLHER ESSAS CHAVES DE MODO QUE NOS DEEM DIRETAMENTE O ENDERECO PARA O REGISTRO CORRESPONDENTE, BIUNIVOCAMENTE, DE TAL FORMA QUE O "HASHING" TORNA-SE DESNECESSARIO. COMO EXEMPLO 1, CONSIDEREMOS UM ARQUIVO DE CLIENTES COM REGISTROS ARMAZENADOS TENDO POR CHAVE O NUMERO DO CLIENTE. SE O NUMERO DE INSCRICAO NO CADASTRO GERAL DE CONTRIBUINTES DO MINISTERIO DA FAZENDA (CGC) E ESCOLHIDO COMO CHAVE, EVIDENTEMENTE ALGUMA FORMA DE "HASH" SERA NECESSARIA. NO ENTANTO, A NOSSA EMPRESA PODE TER SEU PROPRIO SISTEMA DE NUMERACAO DE CLIENTES. NESTE CASO, POR QUE NAO ESCOLHER AS CHAVES DE MODO QUE SEJAM OS PROPRIOS ENDERECOS DOS REGISTROS? ISTO, NATURALMENTE, PERMITIR-NOS-IA O ACESSO DIRETO AOS REGISTROS, SEM "HASHING" NEM NENHUMA OUTRA COMPLICACAO, E DEVERIA SER ADOTADO COM MAIOR FREQUENCIA, NA VIDA PRATICA.

MESMO QUE NAO POSSAMOS ESCOLHER OS VALORES PARA AS CHAVES, ELAS PODEM ESTAR AINDA TAO NAO RANDOMICAMENTE DISTRIBUIDAS, QUE UM METODO DE ACESSO DIRETO DE MELHORES RESULTADOS QUE O EMPREGO DO "HASH". COMO EXEMPLO 2, SUPONHAMOS QUE UMA CIDADE TEM 7,000 TAXIS, TODOS ELES, POREM, TENDO A PLACA COMECANDO PELAS LETRAS SP, SEGUIDAS DE 4 ALGARISMOS. DEVEREMOS, AO INVES DE "HASH", TOMAR OS ULTIMOS 4 ALGARISMOS E USA-LOS COMO INDICE DUMA TABELA DE 10,000 REGISTROS, CONTENDO, CADA UM, UM APONTADOR PARA UM DOS 7,000 REGISTROS DO ARQUIVO REFERENTE AOS TAXIS. E EMBORA ESTE PROCEDIMENTO TENHA ALGUMA SEMELHANCA COM O "HASHING" PELA ANALISE DOS DIGITOS, NOTEMOS QUE ESTE METODO TEM A VANTAGEM ADICIONAL DE QUE OS REGISTROS DO ARQUI-

VO TANTO PODEM SER PROCESSADOS SEQUENCIALMENTE COMO TAMBEM PELO METODO DE ACESSO DIRETO AQUI DESCRITO.

EMBORA A SITUACAO NEM SEMPRE SEJA TAO SIMPLES COMO NOS DOIS EXEMPLOS ACIMA, COM ALGUMA ENGENHOSIDADE O METODO DE A - CESSO DIRETO PODE (E DEVE) SER USADO EM MUITOS CASOS, E DEVE- MOS NOS ESFORCAR POR ISSO.

COMO EXEMPLO 3, SUPONHAMOS QUE TEMOS 10,000 CHAVES ASSIM DISTRIBUIDAS:

1,000 NO INTERVALO 00,000 A 00,000 + 1,199 ,
3,000 NO INTERVALO 20,000 A 20,000 + 3,999 E
6,000 NO INTERVALO 60,000 A 60,000 + 7,999 ;

NESTE CASO, PODEMOS TESTAR A CHAVE PARA DETERMINARMOS EM QUE FAIXA ELA ESTA, SUBTRAIRMOS UMA CONSTANTE APROPRIADA (A FAI - XA) E OBTERMOS DIRETAMENTE O INDICE DUMA TABELA COM 1,199 + 3,999 + 7,999 + 3 = 13,200 ENTRADAS, CONTENDO, CADA UMA DAS ENTRADAS VALIDAS DESSA TABELA, UM APONTADOR PARA AQUELE, DOS 10,000 REGISTROS DO ARQUIVO, QUE TENHA A CHAVE DADA.

VANTAGEM SOBRE O "HASH": ACESSO AOS REGISTROS TANTO SE - QUENCIAL COMO ALEATORIAMENTE, PELA MELHOR (MAIS FACIL, DIRETA E RAPIDA) MANEIRA POSSIVEL.

DESVANTAGEM: ESTE METODO SO PODE SER APLICADO QUANDO PODE- MOS ESCOLHER APROPRIADAMENTE AS CHAVES, COM TOTAL LIBERDADE , OU QUANDO ELAS, NATURALMENTE, JA TEM UM CONJUNTO DE VALORES QUE SE PRESTE PARA ESSE METODO. ALEM DISSO, UMA MUDANCA DE E- QUIPAMENTO, DO SISTEMA OPERACIONAL OU DO TAMANHO DOS REGIS - TROS, PROVAVELMENTE IMPLICARIA NA NECESSIDADE DE RENUMERAR - MOS TODOS OS CLIENTES DO EXEMPLO 1 ACIMA, EMBORA ISSO NEM SEMPRE SEJA NECESSARIAMENTE UM GRANDE IMPECILHO.

1.3.7-COMBINACOES E VARIANTES MAIS SOFISTICADAS DAS TECNICAS BASICAS ACIMA:

NAO ESQUECAMOS O FATO IMPORTANTE DE QUE TODAS ESSAS ALTER- TIVAS PARA O "HASHING", DE 1.3.1 A 1.3.6, SAO APENAS TECNICAS BASICAS. UMA SOLUCAO OTIMA PARA UM DADO PROBLEMA E QUASE SEM- PRE CONSEGUIDA SO PELA COMBINACAO (OU VARIANTES SOFISTICADAS) DESSAS TECNICAS. COMO EXEMPLO, PODEMOS COMBINAR METODOS DE "HASH" COM METODOS DE PESQUISA EM ARVORES BINARIAS, OU COM METODOS DIRETOS, E ASSIM POR DIANTE. PARTICULARMENTE, ASSINA- LEMOS QUE E BASTANTE FREQUENTE UMA GRANDE PROPORCAO DUM AR - QUIVO PODER SER ENDERECADA DIRETAMENTE, USANDO-SE "HASHING" APENAS PARA OS CASOS EXCEPCIONAIS. POR EXEMPLO, SUPONHAMOS QUE, DOS 8,000 TAXIS DUMA CIDADE, 7,000 TENHAM PLACAS COME - CANDO PELAS LETRAS SP SEGUIDAS DE 4 ALGARISMOS, E OS 1,000 TAXIS RESTANTES, POR ALGUM MOTIVO, TENHAM AS PLACAS SEM OBE - DECER NENHUMA REGRA. PODERIAMOS ENTAO FAZER UM TESTE: SE A PLACA COMECASSE POR SP USARIAMOS ENDERECAMENTO DIRETO, E, SE NAO, USARIAMOS "HASH" COM UMA FUNCAO ADEQUADA, MAPEANDO, DI-

GAMOS, DE 10,000 A 11,499.

1.4-FATORES DETERMINANTES DA EFICIENCIA DO "HASHING"

COMO VIMOS, O PRINCIPAL PROBLEMA DO "HASH" E O DAS COLI - SOES. PROCURAMOS SEMPRE CONTORNAR ESSA DIFICULDADE, TRABA - LHANDO CONCOMITANTEMENTE SOBRE OS QUATRO FATORES QUE AFETAM A EFICIENCIA DO "HASHING":

1. O "BUCKET SIZE" ;
2. O FATOR DE CARGA ;
3. A FUNCAO DE "HASHING" ;
4. O METODO DE RESOLUCAO DE "OVERFLOWS".

AS ESCOLHAS DAS POLITICAS A SEREM FIXADAS PARA CADA UM DESTES FATORES ACIMA SAO FEITAS POR NOS MESMOS. ATE CERTO PONTO PODERIAM SER FEITAS INDEPENDENTEMENTE, MAS, HAVENDO CERTAS INTERDEPENDENCIAS ENTRE ESSES FATORES, O MELHOR E QUE LEVEMOS ISTO EM CONTA, CHEGANDO RAPIDAMENTE A UMA SOLUCAO O - TIMIZADA, POR UM PROCESSO QUE PODERIAMOS CHAMAR "ITERATIVO " (POR "PASSO TOTAL" OU POR "PASSO PARCIAL"), QUE CONVERGE RA- PIDAMENTE, USUALMENTE EM 2 OU 3 ITERACOES NO PROCESSO POR "PASSO TOTAL", E 1 OU 2 "ITERACOES", NO PROCESSO POR "PASSO PARCIAL".

1.4.1-"BUCKET SIZE":

SE O ARQUIVO ESTA NA MEMORIA SECUNDARIA, A ADOCAO DE "BUCKETS", LOGICAMENTE COM O "BUCKET SIZE" > 1, E CLARAMENTE- VANTAJOSA (ISTO SO E DISCUTIVEL EM MEMORIA VIRTUAL PAGI- NADA, CONFORME VEREMOS EM 4.3). A VANTAGEM E QUE, SE N REGIS- TROS LOGICOS COMPOEM UM "BUCKET", ENTAO QUALQUER REGISTRO LO- GICO QUE E ARMAZENADO NAQUELE "BUCKET" PODE TER (N-1) SINONI- MOS SEM NENHUM PROBLEMA. NATURALMENTE, PODE HAVER MAIS SINO - NIMOS QUE ISSO, E NESTE CASO NOS DEVEREMOS ENCONTRAR UM OUTRO "BUCKET". ISTO E CHAMADO "BUCKET OVERFLOW" E E TRATADO NO CA- PITULO 3.

E CLARO QUE, SE ESCOLHERMOS UM "BUCKET SIZE" PEQUENO, TE - REMOS UMA PROPORCAO RELATIVAMENTE ALTA DE "OVERFLOWS", NECES- SITANDO DE LEITURAS ADICIONAIS DE "BUCKETS", E POSSIVELMENTE DE "SEEKS" ADICIONAIS. POR OUTRO LADO, SE ESCOLHERMOS UM

"BUCKET SIZE" MAIOR, TEREMOS MENOS "OVERFLOWS", POREM TEREMOS QUE FAZER TANTO UM "INPUT" MAIS DISPENDIOSO DESSE "BUCKET" NA MEMORIA PRINCIPAL, COMO TAMBEM UMA PESQUISA MAIS LONGA PARA DETERMINACAO, DENTRO DO "BUCKET", DO "SLOT" QUE DESEJAMOS.

UMA CUIDADOSA PONDERACAO DESTES E DE OUTROS ASPECTOS DEVE SER FEITA PARA FIXARMOS O "BUCKET SIZE" NUMA APLICACAO, E A FIGURA 1.3 PODERA SER DE BOM VALOR NESTA DECISAO. NO ENTANTO, DE UM MODO GERAL, PODEMOS DIZER QUE, SE O DASD ("DIRECT ACCESS STORAGE DEVICE") NO QUAL O ARQUIVO ESTA, TEM UM TEMPO DE ACESSO RELATIVAMENTE GRANDE, E PREPONDERANTE QUE MINIMIZEMOS O NUMERO DE "OVERFLOWS", DE MODO QUE UM "BUCKET SIZE" DE 10 OU MAIS DEVERA SER ADOTADO. POR OUTRO LADO, SE O ARQUIVO ESTA NUMA MEMORIA DE NUCLEOS MAGNETICOS OU "SOLID-STATE", UM "BUCKET SIZE" = 1 E GERALMENTE MAIS ADEQUADO. NA SECCAO 4.3 VEREMOS QUE ISTO PODE ACONTECER TAMBEM PARA MAQUINAS COM MEMORIA VIRTUAL PAGINADA.

NA PRATICA, O "BUCKET SIZE" E FREQUENTEMENTE ADAPTADO A CERTAS CARACTERISTICAS DE "HARDWARE", DE MODO QUE GERALMENTE UM "BUCKET" COINCIDE COM UM REGISTRO FISICO, UMA TRILHA, MEIA TRILHA DE UM DISCO, UMA CELULA, ETC.

1.4.2-FATOR DE CARGA:

EVIDENTEMENTE, A MEDIDA QUE O FATOR DE CARGA PRIMARIO CRESCER EM DIRECAO A 1, A PROBABILIDADE DE OCORRENCIA DE "OVERFLOWS" CRESCERA TAMBEM. PORISSO NOS DEVEMOS FAZER UMA CUIDADOSA PONDERACAO ENTRE ECONOMIA DE MEMORIA E ECONOMIA DE TEMPO, AJUSTANDO CONVENIENTEMENTE O FATOR DE CARGA PRIMARIO.

NO CAPITULO 2 VEREMOS QUE, DAS VARIAS FUNCOES DE "HASH", ALGUMAS TEM DESEMPENHO PIOR QUE UMA FUNCAO RANDOMICA TEORICA, MUITAS TEM UM DESEMPENHO MAIS OU MENOS EQUIVALENTE, E PELO MENOS UMA TEM GERALMENTE UM DESEMPENHO ALGO MELHOR. NO ENTANTO, EMBORA VEJAMOS ISTO MELHOR NO CAPITULO 2, A FIGURA 1.3, ADAPTADA DE MARTIN (27), JA FORNECE UMA ORIENTACAO BEM UTIL PARA A PONDERACAO ENTRE O FATOR DE CARGA PRIMARIO, O "BUCKET SIZE" E A PROPORCAO DE "OVERFLOWS".

NO CASO DE O ARQUIVO ESTAR NUMA MEMORIA SECUNDARIA ELETROMECANICA COM UM GRANDE TEMPO DE ACESSO, USUALMENTE NOSSA PRINCIPAL PREOCUPACAO SERA BAIXAR O NUMERO DE ACESSOS, DE MODO QUE PODEREMOS VIR A DECIDIR TER A PERCENTAGEM DE "OVERFLOW" EM TORNO DE 1%. COMO VEMOS NA FIGURA 1.3, PODEREMOS CONSEGUIR ISTO TOMANDO UM FATOR DE CARGA PRIMARIO DE 70% E UM "BUCKET SIZE" DE 20 OU MAIS. POR OUTRO LADO, ALGUMAS VEZES O TEMPO DE ACESSO PODE SER DE MENOR IMPORTANCIA QUE O USO EFICIENTE DE MEMORIA, DE MODO QUE DECIDAMOS USAR UM FATOR DE CARGA PRIMARIO DE 95% E AINDA ASSIM ADOTARMOS UM "BUCKET SIZE" GRANDE (DIGAMOS, 50 OU 100) PARA MANTERMOS BAIXA A PERCENTAGEM DE "OVERFLOWS". AO INVES DISSO PODEREMOS QUERER

POUPAR O TEMPO DE CPU NECESSÁRIO PARA A PESQUISA DE UM "BUCKET" GRANDE, DE MODO QUE DEVEREMOS TER UM "BUCKET SIZE" PEQUENO E UM BAIXO FATOR DE CARGA PRIMÁRIO, COM USO DE BASTANTE MEMÓRIA, SE NÃO QUISERMOS QUE A PROPORÇÃO DE "OVERFLOWS" SEJA MUITO ALTA.

NO CASO DE O ARQUIVO ESTAR EM MEMÓRIA DE NÚCLEOS MAGNÉTICOS OU "SOLID-STATE", ONDE O TEMPO DE ACESSO É DESPREZÍVEL, PODEREMOS PREFERIR USAR UM "BUCKET SIZE" IGUAL A 1 E UM ALTO FATOR DE CARGA, PORQUE ESTE TIPO DE MEMÓRIA É BASTANTE CARO.

O CASO DE ARQUIVOS EM MÁQUINAS COM MEMÓRIA VIRTUAL PAGINADA SERÁ DISCUTIDO EM 4.3.

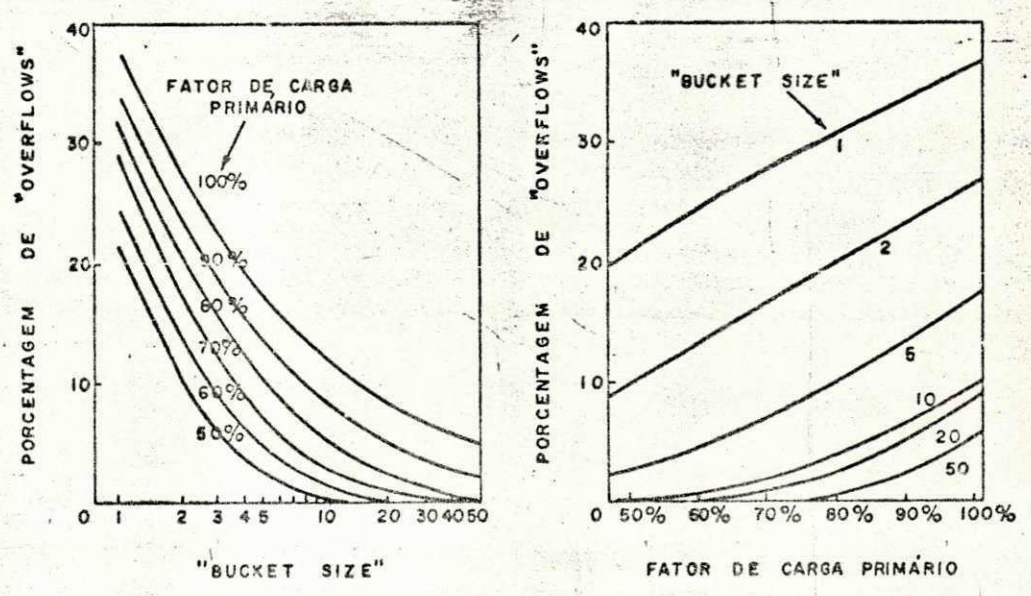


Fig.-1.3 - O Analista de sistemas deve fazer uma compensação entre o fator de carga primário, o "buckt size" e a percentagem de "overflows". As curvas acima correspondem a uma função de "hash" que randomiza perfeitamente o conjunto de chaves, como uma roleta ideal. Compare com a figura-2.1. A presente figura foi retirada de Martin [27].

1.4.3-FUNCAO DE "HASHING":

EMBORA SEJAM QUATRO OS FATORES DETERMINANTES DA EFICIENCIA DO "HASH", E TODOS ELES IMPORTANTES, OS DOIS PRIMEIROS FORMAM UM PROBLEMA MAIS FACILMENTE RESOLVIVEL, DE MODO QUE A PARTE MAIS DIFICIL CONSTITUI-SE DOS DOIS ULTIMOS: A) ESCOLHER A FUNCAO DE "HASHING" E B) ESCOLHER O METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS".

QUER O ARQUIVO ESTEJA NA MEMORIA PRINCIPAL, QUER NA SECUNDARIA, DEVEMOS PROCURAR FAZER O "HASH" DE MODO QUE SEJA MINIMIZADA A PROBABILIDADE DE COLISOES OU "BUCKET OVERFLOWS", ATRAVES DA ESCOLHA ADEQUADA DA FUNCAO DE "HASHING", T(I). TRATAREMOS DESTE ASSUNTO NO CAPITULO 2. NO ENTANTO, ASSINALAMOS QUE O METODO DA DIVISAO, CUIDADOSAMENTE USADO, E UM DOS QUE DA GERALMENTE MELHORES RESULTADOS, SENDO ASSIM, INCLUSIVE PELA SUA SIMPLICIDADE, O METODO ESCOLHIDO NA MAIORIA DAS APLICACOES PRATICAS.

1.4.4-METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS":

COMO VIMOS ACIMA, A ESCOLHA DO METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS" E UM DOS PROBLEMAS DE RESOLUCAO MAIS DIFICIL, NO "HASHING". ESTUDAREMOS ESTE PROBLEMA DETALHADAMENTE NOS CAPITULOS 3 (MAQUINAS SEM PAGINACAO POR DEMANDA) E 4 (MAQUINAS COM PAGINACAO POR DEMANDA).

QUANTO A AREA EM QUE AS COLISOES E/OU "BUCKET OVERFLOWS" SAO RESOLVIDOS, TEMOS DUAS CLASSES DE METODOS:

1.-METODOS SEM AREA DE "OVERFLOW" EM SEPARADO, ESTUDADOS EM 3.2, COM DESEMPENHO UM POUCO MELHOR (PIOR) EM ESPACO (TEMPO) QUE A ABAIXO. AS COLISOES E/OU "BUCKET OVERFLOWS" SAO RESOLVIDAS LIVREMENTE, ATRAVES DE TODO O ARQUIVO.

2.-METODOS COM AREA DE "OVERFLOW" EM SEPARADO, ESTUDADOS EM 3.3. AS COLISOES E/OU "BUCKET OVERFLOWS" SAO RESOLVIDAS EXCLUSIVAMENTE EM CERTAS AREAS ESPECIFICAMENTE DESTINADAS PARA ESTE FIM.

QUANTO AO USO DE APONTADORES, TEMOS TAMBEM DUAS CLASSES DE METODOS:

1.-METODOS QUE USAM APONTADORES, FORMANDO LISTAS ENCADEADAS, COMPOSTAS OU DE "BUCKETS" (EM MEMORIA SECUNDARIA) OU DE REGISTROS LOGICOS (EM MEMORIA PRINCIPAL), DE MODO QUE, PARA ENCONTRARMOS UM REGISTRO, DETERMINAMOS INICIALMENTE SEU "HOME ADDRESS" PARA DEPOIS IRMOS PESQUISANDO A LISTA ASSOCIADA AO MESMO, ATE ENCONTRARMOS A CHAVE PESQUISADA OU CHEGARMOS AO FINAL DA LISTA. VER 3.2.7 E 3.3.1. DE UM MODO GERAL, ESTES METODOS NECESSITAM DE UM POUCO MAIS DE MEMORIA (PARA OS APONTADORES), MAS SAO OS DE MAIOR EFICIENCIA EM TEMPO, PARTICULARMENTE SE O "BUCKET SIZE" E IGUAL A 1

(USUALMENTE EM MEMORIA PRINCIPAL OU VIRTUAL) OU E RELATIVAMENTE PEQUENO (DIGAMOS, ENTRE 1 E 4) E O FATOR DE CARGA PRIMARIO E RELATIVAMENTE GRANDE (DIGAMOS, MAIOR QUE 0.80).

2.-METODOS QUE NAO USAM APONTADORES, RESOLVENDO AS COLISOES E/OU "BUCKET OVERFLOWS" ATRAVES DA DETERMINACAO, POR CERTOS ALGORITMOS, DUMA SEQUENCIA DE ENDERECOS QUE DEVEREMOS PESQUISAR EM SUCESSAO O "HOME ADDRESS". VER 3.2.1 A 3.2.6, 3.3.2 E 3.3.3.

2 - FUNCOES DE "HASH"

2.1-CONSIDERACOES GERAIS SOBRE AS FUNCOES DE "HASH"

COMO MARTIN (27) EXPLANOU, UM ALGORITMO DE TRANSFORMACAO CHAVE-ENDERECO TEM GERALMENTE 3 PASSOS:

1.- SE A CHAVE E NUMERICA, ENTAO GERALMENTE NAO TEMOS PROBLEMAS E PODEMOS PASSAR DIRETAMENTE PARA O PASSO 2 (SALVO SE A ANALISE DOS DIGITOS DOS VALORES DA CHAVE MOSTRAR ALGUMA GRAVE ANOMALIA NA ALEATORIEDADE DESSES VALORES, DEVENDO SER PREVIAMENTE NORMALIZADA. VER 2.3.7).

SE A CHAVE E NAO NUMERICA, ELA DEVE SER PREVIAMENTE TRANSFORMADA (SEM PERDA DE INFORMACOES, MESMO QUE PARCIAL) NUMA FORMA NUMERICA PRONTA PARA MANIPULACAO (POR EXEMPLO, UM CARACTER ALFANUMERICO NAO DEVE SER TRANSFORMADO EM 1 DIGITO, E SIM EM 2 DIGITOS). ORA, EVIDENTEMENTE HA MUITAS SOLUCOES PARA ISTO:

A) QUANDO POSSIVEL, GERALMENTE ADOTAMOS A SOLUCAO DE MANIPULAR A CHAVE ALFANUMERICA COMO UMA CADEIA BINARIA;

B) QUANDO ISTO NAO FOR POSSIVEL, PODEMOS ATRIBUIR VALORES AOS CARACTERES, TAIS COMO A=11, B=12, C=13,..., Z=36, ..., E, USANDO UMA DAS MANEIRAS DE FAZER O "SHIFTING" OU O "FOLDING" VISTAS EM 2.3.8 E 2.3.9 (POR EXEMPLO, SOMANDO OS VALORES NUMERICOS ATRIBUIDOS A CADA UM DOS CARACTERES DA CHAVE), OBTER UM VALOR NUMERICO ASSOCIADO A CHAVE; COMO ASSINALADO POR LUM, YUEN & DODD (24), DIVERSOS ESQUEMAS DIFERENTES PARA CODIFICACAO DOS CARACTERES ALFANUMERICOS FORAM TENTADOS, POREM SEM RESULTAREM EM VARIACAO SIGNIFICATIVA DO DESEMPENHO, PELO MENOS ENQUANTO OS ESQUEMAS DE CODIFICACAO PRESERVARAM A DISTINGUIBILIDADE DOS SIMBOLOS; POR ESTA RAZAO, DEVEMOS SEMPRE USAR A CODIFICACAO INTERNA NATURAL DOS CARACTERES NA MAQUINA (EXEMPLO: CODIGO EBCDIC), DIRETAMENTE, SEM COMPLICACOES DESNECESSARIAS. EM FORTRAN, GERALMENTE TOMAMOS O VALOR ABSOLUTO DA SOMA DOS CARACTERES DA CHAVE, A QUAL FOI USUALMENTE LIDA NO FORMATO A2,A4 OU A8.

2.- FAZEMOS OPERACOES SOBRE OS VALORES NUMERICOS CORRES -

PONDENTES AS CHAVES, UTILIZANDO UM ALGORITMO QUE AS MAPEIE NUMA FAIXA DE NUMEROS COM AMPLITUDE DA MESMA ORDEM DE GRANDEZA DO NUMERO DE "BUCKETS" REQUERIDOS. O CONJUNTO DE CHAVES DEVE SER DISTRIBUIDO TAO UNIFORMEMENTE QUANTO POSSIVEL ATRAVES DESSA FAIXA.

3.- FINALMENTE, OS NUMEROS RESULTANTES SAO MULTIPLICADOS POR UMA CONSTANTE E, SE NECESSARIO, SOMADOS COM OUTRA CONSTANTE, DE MODO QUE SEJAM "AJUSTADOS" A FAIXA ADEQUADA DE ENDERECOS. POR EXEMPLO, O PASSO 2 PODE DAR NUMEROS COM 4 DIGITOS (ISTO E, DE 0,000 A 9,999) QUANDO SO USAMOS 7000 "BUCKETS", PORISSO MULTIPLICAMOS POR 0.7 O QUE FOI OBTIDO NAQUELE PASSO, OBTENDO UM NUMERO (INTEIRO) NO INTERVALO DE 0,000 A 6,999 (*). ESTES NUMEROS DE ENDERECOS RELATIVOS DE "BUCKETS" SAO ENTAO TRANSFORMADOS, PELO SISTEMA OPERACIONAL, NOS ENDERECOS DE MAQUINA DOS "BOKETS".

2.2-QUESTOES A CONSIDERAR NA ESCOLHA DEUMA FUNCAO DE "HASH"

MAURER & LEWIS (30) DEIXARAM MAGNIFICAMENTE CLARO QUE A ESCOLHA DE UMA FUNCAO DE "HASHING" DEPENDE DAS RESPOSTAS AS 7 QUESTOES ABAIXO:

2.2.1-"PODEMOS ESCOLHER NOSSA FUNCAO DE 'HASH' AJUSTANDO-A AO CONJUNTO PARTICULAR DE CHAVES A SEREM PESQUISADAS?"

EM UM MONTADOR, COMPILADOR, INTERPRETADOR OU TRADUTOR MAIS GERAL, OBVIAMENTE A RESPOSTA E 'NAO'. NO PODEMOS SABER QUAIS IDENTIFICADORES SERAO ENCONTRADOS NO PROGRAMA FONTE, ATE QUE REALMENTE OS ENCONTREMOS.

NO CASO DE UM "DATA BASE", ENTRETANTO, A RESPOSTA PODE SER 'SIM', PARTICULARMENTE SE O NOSSO "DATABASE" E ESTATICO, OU QUASE ESTATICO. DISCUTIREMOS MELHOR ESSE ASSUNTO AO ESTUDARMOS O METODO DA ANALISE DOS DIGITOS, EM 2.3.7. PARA CERTOS CONJUNTOS PARTICULARES DE CHAVES, COM CERTAS CARACTERISTICAS,

---

* OBSERVACAO: GERALMENTE OS METODOS DE DIVISAO E MULTIPLICATIVO, VISTOS EM 2.3.1 E 2.3.2, NAO NECESSITAM DESSA OPERACAO.

PODEREMOS, COM CERTA ENGENHOSIDADE, ENCONTRAR FUNCOES DE "HASH" COM UM DESEMPENHO MUITO SUPERIOR AO USUALMENTE OBTIDO PELO EMPREGO SIMPLES E IMEDIATO DE METODOS DE APLICACAO GENERICA.

2.2.2-"E IMPORTANTE O TEMPO NECESSARIO PARA O CALCULO DA FUNCAO DE 'HASH'?"

NO CASO DE OS REGISTROS ESTAREM NA MEMORIA PRINCIPAL, COMO NUM MONTADOR, COMPILADOR, INTERPRETADOR OU TRADUTOR MAIS GERAL, A RESPOSTA PODE SER 'SIM', ISTO E, PODE SER PREFERIVEL ADOTARMOS UMA FUNCAO DE "HASH" QUE NECESSITE, PELA SUA SIMPLICIDADE, DO MINIMO DE TEMPO TOTAL PARA CALCULO (INCLUINDO-SE O TEMPO PARA OS CALCULOS PROPRIAMENTE DITOS E AQUELE PARA OS ACESSOS A MEMORIA PRINCIPAL REQUERIDOS PARA EFETIVACAO DOS CALCULOS). A JUSTIFICATIVA E QUE NAO FARA SENTIDO ESCOLHERMOS UMA FUNCAO QUE SEJA MAIS "EFICIENTE" (NO SENTIDO DE REQUERER, DIGAMOS, 2 A 5% A MENOS DE ACESSOS A MEMORIA PRINCIPAL PARA RESOLUCAO DAS COLISOES), SE ESSA FUNCAO, SENDO DE CALCULO MATEMATICO MAIS COMPLEXO, NECESSITARA, DIGAMOS, DE 100% A MAIS NO TEMPO TOTAL PARA O SEU CALCULO.

NO CASO DE OS REGISTROS ESTAREM NUM DISCO OU TAMBOR, A RESPOSTA QUASE SEMPRE E 'NAO', PARTICULARMENTE SE O TEMPO DE ACESSO FOR GRANDE. ISTO SIGNIFICA QUE NOS DEVEREMOS USAR O METODO QUE MINIMIZE O NUMERO DE ACESSOS A MEMORIA SECUNDARIA, DESPREZANDO O TEMPO NECESSARIO PARA CALCULO DA FUNCAO DE "HASH".

2.2.3-"PODEMOS (E COMPENSA) PROJETAR 'HARDWARE' (OU MICROPROGRAMACAO) PARA IMPLEMENTAR A FUNCAO DE 'HASH'?"

INFELIZMENTE, A MAIORIA DOS COMPUTADORES NAO TEM INSTRUCOES QUE PERMITAM A REALIZACAO DO "HASHING" DIRETAMENTE, A NIVEL DE "HARDWARE", O QUE TALVEZ SEJA UMA DAS CARACTERISTICAS MAIS DESEJAVEIS PARA AS PROXIMAS MAQUINAS. PRESENTEMENTE, DEVEMOS USAR ALGUMA COMBINACAO DAS INSTRUCOES EXISTENTES, E ISTO E BASTANTE INOPORTUNO, UMA VEZ QUE O "HASHING" E UMA OPERACAO TAO COMUM NUMA LARGA VARIEDADE DE PROGRAMAS. UMA DAS FUNCOES DE "HASH" (METODO DA DIVISAO POLINIMIAL OU DA CODIFICACAO ALGEBRICA, QUE VEREMOS EM 2.3.10) FOI DESENVOLVIDA ESPECIFICAMENTE PARA IMPLEMENTACAO A NIVEL DE "HARDWARE".

OUTRA POSSIBILIDADE SERIA PROJETAR UMA INSTRUCAO DE "HASHING" QUE INCORPORASSE UM OU ALGUNS DOS METODOS DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS", VISTOS NO CAPITULO 3.

HOJE, O MAIS VIAVEL TALVEZ SEJA EMPREGAR MICROPROGRAMACAO.

2.2.4-"QUAL O COMPRIMENTO DAS CHAVES?"

SE ESTIVERMOS PREOCUPADOS COM O TEMPO DE CALCULO DA FUN - CAO, PODEREMOS NAO DESEJAR USAR DIRETAMENTE UM METODO COMO O DA DIVISAO PARA CHAVES QUE SEJAM MAIORES QUE UMA PALAVRA DO COMPUTADOR. PARA CHAVES QUE SEJAM MAIORES, PODEREMOS COMBINAR O METODO DE "FOLDING" OU "SHIFTING", APRESENTADOS EM 2.3.9 E 2.3.8, COM QUALQUER UM DOS OUTROS METODOS QUE VEREMOS DE 2.3.1 A 2.3.10.

2.2.5-"OS ENDERECOS SAO BINARIOS OU DECIMAIS?"

NECESSITAMOS TER ESTA RESPOSTA PORQUE, DENTRE OS VARIOS METODOS VISTOS EM 2.3, ALGUNS APOIAM-SE FORTEMENTE EM INSTRU- COES ESPECIFICAS QUE NORMALMENTE SAO DISPONIVEIS SO NA FORMA BINARIA OU SO NA FORMA DECIMAL DE ARITMETICA E ENDERECAMENTO.

2.2.6-"PODEMOS ESCOLHER LIVREMENTE OS ENDERECOS OU INDICES QUE SERAO USADOS NO NOSSO ARQUIVO?"

SE 'SIM', PODEMOS ESCOLHER O TAMANHO DO ARQUIVO COMO UMA POTENCIA DE 2 (OU DE 10), SE OS ENDERECOS SAO BINARIOS (OU DECIMAIS). ISTO OU E REQUERIDO POR ALGUNS DOS METODOS APRE - SENTADOS EM 2.3, OU E BASTANTE CONVENIENTE.

SE 'NAO', O QUE SIGNIFICA QUE OS ENDERECOS JA FORAM PRE - VIAMENTE FIXADOS POR UMA OUTRA MANEIRA QUALQUER, NOSSAS ROTI- NAS DE "HASHING" DEVERAO LEVAR ISTO EM CONTA, ADOTANDO-SE AS CONDICOES E RESTRICOES QUE JA NOS FORAM IMPOSTAS.

2.2.7-"AS CHAVES FORAM ESCOLHIDAS MAIS OU MENOS ALEATORIAMEN- TE?"

EM MUITOS CASOS AS CHAVES ESCOLHIDAS ESTAO TAO PROXIMAS DE SEREM PERFEITAMENTE ALEATORIAS QUE QUASE TODOS OS METODOS DE 2.3 DARAO OTIMOS RESULTADOS.

EM MUITOS OUTROS CASOS AS CHAVES SE AFASTAM BASTANTE DA A- LEATORIEDADE PERFEITA, DE MODO QUE, PARA UMA DETERMINADA FUNCAO DE "HASH" ESCOLHIDA, UMA AMPLA CLASSE DE CHAVES PLAU - SIVEIS TERA O MESMO "HOME ADDRESS", OBRIGANDO-NOS A ESTUDAR CUIDADOSAMENTE A FUNCAO DE "HASH", PARA EVITAR ISSO O MAIS POSSIVEL. EXISTEM MESMO CASOS EXTREMOS EM QUE AS CHAVES SAO TAO NAO ALEATORIAMENTE DISTRIBUIDAS, QUE DEVEMOS DECIDIR DE-

FINITIVAMENTE NAO USAR "HASH", E SIM UMA DAS VARIANTES DOS METODOS DE ENDERECAMENTO DIRETO (1.3.6) OU OUTRA QUALQUER DAS ALTERNATIVAS PARA O "HASHING", VISTAS EM 1.3.

SOB CERTAS CONDICOES, TODAVIA, PODEMOS PROVAR QUE CHAVES CONSTRUIDAS DE CERTAS MANEIRAS ESPECIAIS 'NAO TEM' O MESMO "HOME ADDRESS", OU SEJA, CERTOS DESVIOS DA ALEATORIEDADE PERFEITA PODEM SER TRANSFORMADOS EM VANTAGEM. EM PARTICULAR, SUPONHAMOS QUE TEMOS IDENTIFICADORES TAIS COMO I, (I+J), (I+ 2*J), ... (O QUE E COMUM NA AREA DE COMPILADORES, ETC., POIS MUITOS DOS PROGRAMAS FONTES TRARAO IDENTIFICADORES TAIS COMO FILA1, FILA2, FILA3, ETC.). SE O METODO DA DIVISAO QUE VEREMOS EM 2.3.1 FOR USADO, E SE T(I)=A, ENTAO T(I+J)=A+J, T(I+2*J)=A+2*J E ASSIM POR DIANTE (MODULO O TAMANHO DO ARQUIVO, M). ASSIM, SOB ESSAS CONDICOES, ASSEGURAMO-NOS DE QUE ESTAS CHAVES NAO COLIDIRAO ENTRE SI, O QUE E UMA GRANDE VANTAGEM. ESTE ASSUNTO SERA MELHOR DISCUTIDO EM 2.3.1 E 2.4.2.

2.3-METODOS PARA CALCULO DA FUNCAO DE "HASH"

DE UM MODO GERAL, BUCHHOLZ(07), DIPEL & HOUSE(10), HELLERMAN (01), KNUTH (20), MAURER (29), WEGNER (50), MARTIN (27) E MAURER & LEWIS (30) TRAZEM BOAS "SURVEYS" SOBRE FUNCOES DE "HASH", PODENDO SEREM CONSULTADOS, PARA MELHOR COMPREENSAO DO ASSUNTO.

2.3.1-METODO DA DIVISAO:

A CHAVE I E DIVIDIDA POR UM CERTO NUMERO M, IGUAL A (OU POUCO MENOR QUE) O NUMERO DE ENDERECOS DE "BUCKETS" POSSIVEIS, TOMANDO-SE O RESTO DA DIVISAO COMO O VALOR PARA T(I), OU SEJA, COMO O ENDERECO RELATIVO DO "BUCKET", CONSIDERANDO-SE QUE ESSES ENDERECOS VARIAM DESDE 0 ATE M-1, (OU DE 0 ATE MAIS QUE M-1). OBVIAMENTE, SE QUISERMOS QUE ESSA VARIACAO SEJA A PARTIR DE 1 (POR EXEMPLO, DE 1 A M), BASTARA SOMARMOS +1 AO RESTO DESSA DIVISAO.

EM LINGUAGEM MATEMATICA:

$$T(I) = I \bmod M \qquad (2.1)$$

ONDE $0 \leq T(I) < M$

EM LINGUAGEM FORTRAN, POR EXEMPLO:

T(I) = MOD (I,M) = I - I/M*M (ARITMETICA INTEIRA)

EM CERTOS CASOS, ALGUNS VALORES DE M SAO BEM MELHORES QUE OUTROS (VER KNUTH (20), PAGINAS 508 E 509). SENDO R A BASE DO CONJUNTO DE CARACTERES USADO, (USUALMENTE R=64, 256 OU 100), O IDEAL SERIA QUE M FOSSE UM NUMERO PRIMO TAL QUE (R**K+A) MOD. M E (R**K-A) MOD. M FOSSEM NAO NULOS, PARA PEQUENOS VALORES DE A E K. NO ENTANTO ESTUDOS COMO OS DE LUM, YUEN & DODD (24) MOSTRAM QUE, NA PRATICA, JA E OTIMA A ESCOLHA DE M SIMPLESMENTE COMO O MAIOR NUMERO PRIMO MENOR OU IGUAL QUE O NUMERO DE ENDERECOS DE "BUCKETS" POSSIVEIS, SENDO TAMBEM PLENAMENTE ACEITAVEL, NA MAIORIA DOS CASOS, UM VALOR DE M NAO PRIMO, DESDE QUE NAO SEJA FAVORAVEL POR NUMEROS MUITO PEQUENOS (DIGAMOS, DE 2 A 17).

NAO OBSTANTE ASSINALARMOS QUE E SEMPRE PREFERIVEL SEGUIRMOS AS RECOMENDACOES ACIMA, ASSINALAMOS TAMBEM QUE ALGUMAS TECNICAS, COMO AS DESCRITAS EM 4.2.2 (2A. APLICACAO), BASEIAM-SE EM TOMARMOS M COMO UMA POTENCIA 2**K (10**K), DESDE QUE O ENDERECAMENTO SEJA BINARIO (DECIMAL). ESSA ESCOLHA DE M NAO TERA GRAVES INCONVENIENTES, DESDE QUE AS CHAVES TENHAM UMA ALEATORIEDADE BEM SATISFATORIA, SEM LEIS DE FORMACAO PARA CERTOS GRUPOS DE CHAVES. O METODO SERA ENTAO BASTANTE SIMPLIFICADO, CONSISTINDO APENAS EM TOMARMOS OS K "BITS" (DIGITOS) MENOS SIGNIFICATIVOS DA CHAVE (O QUE PODE SER FEITO POR UMA RAPIDA OPERACAO DE "SHIFT"), RAZAO PELA QUAL O METODO, NESTE CASO PARTICULAR, E TAMBEM CHAMADO DE METODO DO TRUNCAMENTO.

COMO UM EXEMPLO, CONSIDEREMOS QUE TEMOS O IDENTIFICADOR I=123,456 E QUE TEMOS 7,000 "BUCKETS". PODEREMOS TOMAR M=6,997, OBTENDO:

T(I)=123,456 MOD. 6,997=4,507 .

UM OUTRO BOM EXEMPLO E ENCONTRADO NA FIGURA 3.1.

O METODO DA DIVISAO, ALEM DE SER BASTANTE SIMPLES, DA GERALMENTE RESULTADOS MELHORES QUE OUTROS METODOS QUE, TEORICAMENTE, TEM UMA UNIFORMIDADE MAIS PERFEITA NO MAPEAMENTO DAS CHAVES. UMA RAZAO PARA ISTO E QUE MUITOS CONJUNTOS DE CHAVES TEM SEQUENCIAS DE VALORES CONSECUTIVOS; NUM METODO "PERFEITAMENTE" RANDOMICO, ALGUMAS DESSAS CHAVES PODERIAM COLIDIR; MAS NAO NO METODO DA DIVISAO, POIS OS RESTOS DA DIVISAO DE NUMEROS CONSECUTIVOS POR UM MESMO DIVISOR M, SAO NUMEROS TAMBEM CONSECUTIVOS, E PORTANTO DISTINTOS. VER 2.4.2.

AO QUE SAIBAMOS, QUEM PRIMEIRO ESCREVEU SOBRE ESTE METODO FOI DUMEY (11).

2.3.2-METODO MULTIPLICATIVO:

SENDO W O MAIOR INTEIRO PARA O COMPUTADOR (CORRESPONDENTE A UMA PALAVRA DO MESMO), ESCOLHEMOS UM VALOR, A, DA MESMA ORDEM DE GRANDEZA DE W, E PRIMO RELATIVO COM W, CALCULANDO - SE

DE UMA VEZ POR TODAS UMA CONSTANTE (FRACIONARIA) :

$$C = A/W \qquad (2.2)$$

DADO UM IDENTIFICADOR I, ADOTAMOS ENTAO

$$T(I) = \lfloor M*((C*I) \text{ MOD. } 1) \rfloor \qquad (2.3)$$

ONDE M E O NUMERO DE VALORES POSSIVEIS PARA T(I), A NOTACAO $\lfloor$ E $\rfloor$ SIGNIFICA " O MAIOR INTEIRO CONTIDO POR E" OU " E ARREDONDADO PARA MENOS" E A OPERACAO (C*I)MOD. 1 E FEITA PRIMEIRAMENTE, PARA DEPOIS M SER MULTIPLICADO POR ESTE RESULTADO.

USUALMENTE, TOMAMOS M COMO UMA POTENCIA DE 2 (10) NUM COMPUTADOR BINARIO (DECIMAL), DE MODO QUE T(I) CONSISTE DOS "BITS" (DIGITOS) INICIAIS DA METADE MENOS SIGNIFICATIVA DO PRODUTO A*I, O QUE FACILITA MUITO AS COISAS.

EXEMPLO: O COMPUTADOR E DECIMAL, A PALAVRA TEM 3 DIGITOS (DONDE W=10**3=1,000) E I=3,333. ESCOLHEMOS A=711 E M=10**2 = 100, DE MODO QUE C=A/W=711/1,000=0.711. USANDO O QUE FOI DITO NO PARAGRAFO IMEDIATAMENTE ANTERIOR, TEMOS:

$$A*I = 711*3{,}333 = 2{,}369{,}763$$

DONDE: $T(I) = 76$, OBTIDO SO COM UMA MULTIPLICACAO.
SE USARMOS DIRETAMENTE (2.3), OBTEREMOS O MESMO VALOR, MAS POR UM CUSTO MAIOR:

$$T(I) = \lfloor 100*((0.711*3{,}333) \text{ MOD. } 1) \rfloor =$$
$$= \lfloor 100*0.763 \rfloor = 76$$

O METODO MULTIPLICATIVO TEM DESEMPENHO COMPARAVEL AO METODO DA DIVISAO, E E AS VEZES ESCOLHIDO EM LUGAR DESTE ULTIMO, QUANDO O TEMPO DE CALCULO E IMPORTANTE E, A OPERACAO DE MULTIPLICACAO SENDO MAIS RAPIDA QUE A DE DIVISAO, PODEMOS TOMAR M COMO POTENCIA DE 2 (OU DE 10).

EM UM CERTO SENTIDO, O METODO MULTIPLICATIVO PODE SER ENCARADO COMO UMA GENERALIZACAO DO METODO DA DIVISAO, POIS, DADO M (MESMO QUE PRIMO), PODEMOS TOMAR A COMO UMA APROXIMACAO DE W/M, DE MODO QUE C=1/M E AS FORMULAS 2.2 E 2.3 EQUIVALERAO A MULTIPLICARMOS I POR 1/M, TOMAR DO RESULTADO SOMENTE A PARTE FRACIONARIA (O QUE PODE SER FEITO SEM DIVISOES) E MULTIPLICA-LA POR M, OBTENDO O MESMO RESULTADO QUE TERIAMOS PELA FORMULA 2.1. EM ALGUNS COMPUTADORES, MESMO TUDO ISTO AINDA SERA MAIS RAPIDO QUE UMA DIVISAO, E ASSIM PREFERIR-SE-A O PRESENTE METODO.

EMBORA O METODO MULTIPLICATIVO TENHA ALGUMA SEMELHANCA COM O ABAIXO EXPOSTO ("MID-SQUARE"), FOI VERIFICADO QUE, PARA CONSTANTES ADEQUADAS, O PRESENTE METODO APRESENTA MELHOR DESEMPENHO.

SE AS CHAVES SAO APROXIMADAMENTE ALEATORIAS, OBTEMOS OTIMOS RESULTADOS TOMANDO "A" COMO O INTEIRO MAIS PROXIMO DA "RAZAO AUREA" = (SQRT(5.)-1)/2*W = 0.618,033,988,7...*W E QUE SEJA PRIMO RELATIVO COM W. ESTE "HASHING" E CONHECIDO POR ALGUNS COMO "HASHING" DE FIBONACCI.

SE O COMPUTADOR E DECIMAL, COM PALAVRA COMPOSTA DE 5 " BYTES", CADA UM PODENDO REPRESENTAR UMA CENTENA OU UM CARACTER

ALFANUMERICO, PODEREMOS TOMAR:

A = 61 80 33 98 87.

SUPONHAMOS R=100 (BASE DA CODIFICACAO) E 1 PALAVRA = 5 CARACTERES. SE HA CERTA PROBABILIDADE DE OCORREREM CHAVES COM 1 PALAVRA DE COMPRIMENTO, COM O SEGUNDO CARACTER EM PROGRESSAO ARITMETICA (POR EXEMPLO: E1___, E2___, E3___, ...) E SE AS CHAVES TEM REPRESENTACAO AJUSTADA A ESQUERDA ("LEFT JUSTIFIED"), TOMARMOS PARA "A" O VALOR ACIMA INDICADO TERA O MESMO EFEITO DE USARMOS UM MULTIPLICADOR EFETIVO C=(100**3*A/W) = 0.9887, DEMASIADAMENTE DIFERENTE DA RAZAO AUREA. SERA MELHOR TOMARMOS UM MULTIPLICADOR TAL COMO:

A = 61 61 61 61 61

NO ENTANTO, A SOLUCAO ACIMA TEM O INCONVENIENTE DE QUE CHAVES COMO AB E BA TENDERAO A TER O MESMO VALOR T(I).

ESTUDOS FEITOS MOSTRAM QUE OS MELHORES VALORES PARA C FICAM NUMA DAS FAIXAS:
(1/4<C<3/10), (1/3<C<3/7), (4/7<C<2/3) OU (7/10<C<3/4).

PARA O COMPUTADOR DO EXEMPLO ACIMA, PODEREMOS ESCOLHER "A" DE MODO QUE O VALOR DE CADA "BYTE" RECAIA NUMA DAS FAIXAS REFERIDAS E NAO SEJA DEMASIADAMENTE PROXIMO DOS VALORES DOS OUTROS "BYTES" E DOS SEUS COMPLEMENTOS. ASSIM, RECOMENDAMOS, POR EXEMPLO:

A = 61 25 42 33 71

PARA MAIORES DETALHES, VER KNUTH (20), PAGINAS 509 A 511 .

2.3.3-METODO DO QUADRADO DO MEIO ("MID-SQUARE"):

O IDENTIFICADOR I E MULTIPLICADO POR SI MESMO, E OS "BITS" (DIGITOS) NECESSARIOS DO CENTRO DO QUADRADO SAO TOMADOS E AJUSTADOS A FAIXA DE ENDERECOS RELATIVOS DOS "BUCKETS".

ASSIM, SE TIVERMOS UMA MAQUINA DECIMAL, O ARQUIVO USAR M = 7,000 "BUCKETS" E AS CHAVES DOS REGISTROS TIVEREM 6 DIGITOS, ELEVAMOS AO QUADRADO CADA CHAVE, OBTENDO UM CAMPO DE 12 DIGITOS, E USAMOS APENAS OS 4 DIGITOS COM NUMERO DE ORDEM 5, 6, 7 E 8. POR EXEMPLO, SE I=123,456, O QUADRADO SERA ..... 015,241,383,936, OS 4 DIGITOS CENTRAIS FORMAM O NUMERO 4,138 QUE, COMO PODERIAM FORMAR NUMEROS DE 0 A 9,999, DEVE SER MULTIPLICADO POR 7,000/10**4=0.7, OBTENDO-SE T(I)=0.7*4,138= 2,896 (EM ARITMETICA INTEIRA), QUE SERA USADO COMO ENDERECO RELATIVO DO "BUCKET".

ESTE METODO E UM DOS QUE DA RESULTADOS MAIS PROXIMOS DAQUELES RESULTADOS TEORICOS MOSTRADOS NA FIGURA 1.3 (OU SEJA, E UM DOS METODOS QUE MAIS SE APROXIMA DO DESEMPENHO DE UM ALGORITMO DE RANDOMIZACAO TEORICAMENTE "PERFEITO"), PELO MENOS ENQUANTO AS CHAVES NAO TEM PROPENSAO A COMECAR OU TERMINAR POR UMA GRANDE QUANTIDADE DE ZEROS.

2.3.4-METODO RANDOMICO:

QUASE TODOS OS CENTROS DE COMPUTACAO TEM SUBROTINAS QUE SAO USADAS QUANDO UMA SEQUENCIA DE NUMEROS ALEATORIOS E DESEJADA, ESPECIALMENTE EM TRABALHOS DE ESTATISTICA OU DE SIMULACAO. ESSSAS SUBROTINAS SAO CONHECIDAS COMO "GERADORES DE NUMEROS RANDOMICOS". NA VERDADE, SENDO GERADOS SEGUNDO UM ALGORITMO, ESSES NUMEROS NAO SAO CONCEITUALMENTE RANDOMICOS E, A RIGOR, DEVERIAMOS CHAMAR AS SUBROTINAS DE "GERADORES DE NUMEROS PSEUDO -RANDOMICOS". NO ENTANTO, TOLERAMOS A PRIMEIRA DESIGNACAO, DESDE QUE A SEQUENCIA DE NUMEROS GERADOS PASSE POR TODOS OS TESTES CONHECIDOS DE ALEATORIEDADE, TAO BEM COMO UMA SEQUENCIA DE NUMEROS "SORTEADOS" POR UMA "ROLETA PERFEITA" . TIPICAMENTE FORNECEMOS A UMA DESSAS SUBROTINAS UM VALOR INICIAL, CHAMADO DE SEMENTE ("SEED"), E A SUBROTINA NOS FORNECE, QUANDO CHAMADA REPETITIVAMENTE, UMA SEQUENCIA DE NUMEROS PSEUDO-RANDOMICOS.

O METODO RANDOMICO CONSISTE EM UTILIZARMOS O IDENTIFICADOR I COMO A SEMENTE, ESCOLHER O PRIMEIRO DOS NUMEROS RANDOMICOS PARA SER O "HOME ADDRESS" E USAR OS OUTROS, SE NECESSARIO , PARA RESOLUCAO DE "BUCKET OVERFLOWS". NATURALMENTE, COMO NUMEROS RANDOMICOS PRODUZIDOS POR TAIS SUBROTINAS SAO USUALMENTE NUMEROS REAIS ENTRE 0 E 1, SENDO M O NUMERO DE ENDERECOS DE "BUCKETS" POSSIVEIS, NOS DEVEMOS PREVIAMENTE MULTIPLICAR O NUMERO RANDOMICO INICIAL POR M E DEPOIS TRUNCAR ESTE NUMERO REAL PARA INTEIRO, OBTENDO UM NUMERO RANDOMICO INTEIRO ENTRE 0 E M-1 INCLUSIVE.

AO QUE SAIBAMOS, QUEM PRIMEIRO MENCIONOU O METODO RANDOMICO FOI MCILROY (32). DETALHES SOBRE A GERACAO DE NUMEROS RANDOMICOS PODEM SER ENCONTRADOS EM NAYLOR & OUTROS (34), BEM COMO EM KNUTH (19).

2.3.5-METODO DA MUDANCA DE BASE ("RADIX"):

HA MUITAS VARIANTES PARA ESTE METODO E UMA DAS MAIS USADAS E A SEGUINTE: SEJA I=123,456 UM IDENTIFICADOR E M=7,000 O NUMERO DE ENDERECOS RELATIVOS DE "BUCKETS" POSSIVEIS; INTERNAMENTE, NUM COMPUTADOR DE ARITMETICA DECIMAL EM BCD ("BINARY - CODED DECIMAL"), TEREMOS I=0001 0010 0011 0100 0101 0110; TOMAMOS AGORA A CHAVE COMO SE FOSSE UMA CADEIA DE DIGITOS OCTAIS I'=000 100 100 011 010 001 010 110 = 04432126 (NA BASE 8); CONSIDERAMOS AGORA QUE ESTA ULTIMA CADEIA DE DIGITOS ESTA NUMA OUTRA BASE, DIGAMOS 11, OBTENDO I'' = 0*11**7 + 4*11**6+ 4*11**5 + 3*11**4 + 2*11**3 + 1*11**2 + 2*11**1 + 6*11**0 = 7,777,182 (NA BASE 10); COMO NO METODO "MID-SQUARE", TOMAMOS OS 4 DIGITOS CENTRAIS I'''=7,771; MULTIPLICAMOS FINALMENTE ESTE ULTIMO VALOR POR 7.000/10**4=0.7, OBTENDO ENTAO T(I) =

0.7*7,771 = 5,439.

UMA OUTRA VARIANTE MUITO USADA CONSISTE SIMPLESMENTE EM CONSIDERAR OS DIGITOS DO IDENTIFICADOR COMO RELATIVOS A UMA OUTRA BASE, DIGAMOS, 11. PARA OS MESMOS VALORES I E M DO EXEMPLO ACIMA, TEMOS I'=1*11**5 + 2*11**4 + 3*11**3 + 4*11**2+ 5*11**1 + 6*11**0 = 194,871 (NA BASE 10); TOMANDO OS 4 DIGITOS CENTRAIS OBTEMOS I''=9,487; MULTIPLICANDO POR 0.7, TEMOS FINALMENTE T(I)=0.7*9,487=6,640. MUDANCA DE BASE PARA 11, NUM COMPUTADOR DECIMAL, PODE SER FEITA MAIS RAPIDAMENTE POR UMA SERIE APROPRIADA DE "SHIFTS" E ADICOES.

NA REALIDADE O METODO DE MUDANCA DE BASE E UMA PARTICULARIZACAO DO ABAIXO.

2.3.6-METODO DE LIN:

ESTE METODO CONSISTE EM TOMARMOS OS DIGITOS DO IDENTIFICADOR I COMO SE ESTIVESSEM NUMA OUTRA BASE P, CALCULARMOS O VALOR CORRESPONDENTE AO IDENTIFICADOR ASSIM CONSIDERADO E O TOMARMOS EM MODULO Q**N, ONDE P E Q SAO NUMEROS PRIMOS (OU SEM PEQUENOS FATORES PRIMOS) E N E UM INTEIRO POSITIVO. GERALMENTE, POR SIMPLICIDADE, TOMAMOS P E Q APENAS RELATIVAMENTE PRIMOS ENTRE SI, OU SEJA, USUALMENTE P=Q+1, ESCOLHENDO Q E M DE MODO QUE Q**N SEJA IGUAL A (OU POUCO MENOR QUE) O NUMERO DE ENDERECOS RELATIVOS DE "BUCKETS" POSSIVEIS, M.

VAMOS ILUSTRAR ESTE METODO COM O EXEMPLO DADO POR LUM , YUEN & DODD (24). SEJA I=975, P=8, Q=7, N=2 E M=50. CODIFICANDO A CHAVE EM BCD, DE MANEIRA QUE A CADA DIGITO CORRESPONDEM 4 "BITS", TEMOS A CADEIA BINARIA 1001 0111 0101. TOMEMOS ESTA CADEIA E REAGRUPEMO-LA EM GRUPOS DE 3 "BITS" (POIS 2**3 =8=P), OBTENDO 100 101 110 101 = 4565 (NA BASE 8) = 2,421 (NA BASE 10). TEMOS , FINALMENTE T(I)=2,421 MOD. (7**2) = 20 .

NOTEMOS QUE SE P FOSSE IGUAL A 10, ENTAO O ENDERECO RELATIVO DE "BUCKET" TERIA SIDO OBTIDO SIMPLESMENTE TOMANDO O IDENTIFICADOR EM MODULO Q**N, COMO NO METODO DA DIVISAO.

NOTEMOS TAMBEM QUE EXISTEM DIFERENTES MANEIRAS DE EXPRESSAR UMA CHAVE COMO UMA CADEIA BINARIA, ALEM DOS CODIGOS BCD, BINARIO, ETC., E OS ESTUDOS DE LUM, YUEN E DODD (24) MOSTRARAM QUE NAO EXISTE DIFERENCA SIGNIFICATIVA NO DESEMPENHO DESSAS VARIAS MANEIRAS.

PARA MAIOR APROFUNDAMENTO, VER LIN (22). O QUE AQUI MOSTRAMOS DIFERE EM DETALHES DA EXPOSICAO INICIAL DE LIN, MAS OS PRINCIPIOS SAO OS MESMOS.

2.3.7-METODO DA ANALISE DOS DIGITOS:

ESTE E O UNICO DOS METODOS DE "HASH" AQUI APRECIADOS QUE

DEPENDE DO CONJUNTO DE CHAVES A SEREM USADAS.

A IDEIA E ANALISARMOS A DISTRIBUICAO DE VALORES DE CADA UM DOS DIGITOS OU CARACTERES DA CHAVE, USANDO CERTOS INDICES ESTATISTICOS OBTIDOS DA ANALISE DUMA AMOSTRA A MAIOR POSSIVEL DAS CHAVES. ASSIM, SE O NUMERO DE ENDERECOS RELATIVOS DE "BUCKETS" POSSIVEIS E M=1,000=10**3, SE AS CHAVES TEM TODAS ELAS 6 DIGITOS, SE O PRIMEIRO DIGITO E SEMPRE 1 E O SEXTO DIGITO QUASE SEMPRE E PAR, ENQUANTO OS DEMAIS TEM UMA DISTRIBUICAO QUASE PERFEITAMENTE UNIFORME, ENTAO, DADO O IDENTIFICADOR I=123,456, UMA SOLUCAO SERA ABANDONARMOS O PRIMEIRO E O SEXTO DIGITO, COMO TAMBEM UM OUTRO QUALQUER COM BOA DISTRIBUICAO, DIGAMOS, O SEGUNDO, OBTENDO ASSIM, DIRETAMENTE, T(I)=345; UMA OUTRA SOLUCAO SERA ABANDONARMOS SO O PRIMEIRO E O ULTIMO DIGITO, APLICANDO QUALQUER OUTRA FUNCAO DE "HASHING" SOBRE O NUMERO RESTANTE 2,345, NA ESPERANCA DE QUE, EM AMBAS AS SOLUCOES, TEREMOS UM "HASHING" MAIS UNIFORME QUE SE PARTISSEMOS DA "MA" CHAVE INICIAL, COM TODOS OS SEUS DIGITOS.

ESTE METODO E USADO MAIS PARA ARQUIVOS ESTATICOS OU QUASE ESTATICOS, OU QUANDO, PARA ARQUIVOS DINAMICOS, AS CHAVES TEM, SE NAO UMA REGRA (PARCIAL) DE FORMACAO, PELO MENOS UMA FORTE TENDENCIA (PARCIAL) NESSA FORMACAO.

NA PRATICA, TOMADAS TODAS AS PRECAUCOES POSSIVEIS PARA A APLICACAO DE UM OUTRO METODO QUALQUER DA MANEIRA MAIS EFICIENTE POSSIVEL, VERIFICOU-SE QUE E PREFERIVEL QUE A FUNCAO DE "HASH" DEPENDA DE TODOS OS "BITS" DAS CHAVES.

PARA MAIORES DETALHES, VER MAURER & LEWIS (30) E PRICE.... (39).

2.3.8-METODO DO DESLOCAMENTO ("SHIFTING"):

A CHAVE E DIVIDIDA EM UM NUMERO DE PARTES CADA UMA DAS QUAIS, EXCETO A ULTIMA, TEM UM COMPRIMENTO DA MESMA ORDEM DE GRANDEZA DAQUELES DOS ENDERECOS RELATIVOS DOS "BUCKETS" POSSIVEIS. DAMOS UM DESLOCAMENTO ("SHIFTING") EM CADA UMA DESSAS PARTES, DE MODO QUE FIQUEM TODAS ALINHADAS PELA DIREITA (OU ESQUERDA), E EM SEGUIDA FAZEMOS A SOMA (OU "EXCLUSIVE OR" NUMA MAQUINA BINARIA) DE TODAS ELAS, AJUSTANDO FINALMENTE O RESULTADO A FAIXA DE VARIACAO DOS ENDERECOS DE "BUCKETS" POSSIVEIS.

COMO EXEMPLO, CONSIDEREMOS O IDENTIFICADOR ..... I=1,234,567,890,123, QUE O NUMERO DE "BUCKETS" POSSIVEIS E M=700 E QUE O COMPUTADOR E DECIMAL. CADA SECCAO DEVERA TER 3 DIGITOS, PORQUE 10**3>M. ASSIM, TEREMOS:

I = | 123 | 456 | 789 | 012 | 3 |

1A. SECCAO: 123

2A. SECCAO: 456

3A. SECCAO: 789

4A. SECCAO: 012

5A. SECCAO: 3

SOMA: 1,383 MOD. 1,000 = 383

T(I) = 0.7*383 = 268

ESTE METODO E FREQUENTEMENTE USADO EM COMBINACAO COM OUTROS METODOS, QUANDO AS CHAVES SAO LONGAS. POR EXEMPLO, SE TIVERMOS UMA CHAVE DE 32 "BYTES" NUM IBM 370, PODEMOS CONSIDERA-LA COMO 8 PALAVRAS SIMPLES, FAZER A ADICAO (OU "EXCLUSIVE-OR") SOBRE AS MESMAS E USAR O METODO DA DIVISAO (OU ALGUM OUTRO) SOBRE A PALAVRA SIMPLES RESULTANTE.

NOTEMOS QUE SE TIVESSEMOS, NO EXEMPLO , QUALQUER CHAVE COMO I' = 123 789 456 012 3, TERMINARIAMOS POR TER T(I')=T(I) . G. D. KNOTT SUGERIU EVITARMOS ESTE PROBLEMA FAZENDO UM DESLOCAMENTO ("SHIFT") CICLICO (O PRIMEIRO "BIT" PASSANDO A SER O ULTIMO) JUSTAMENTE ANTES DE FAZERMOS CADA ADICAO OU "EXCLUSIVE-OR". A VANTAGEM DESSA IDEIA, NA PRATICA, E DISCUTIVEL.

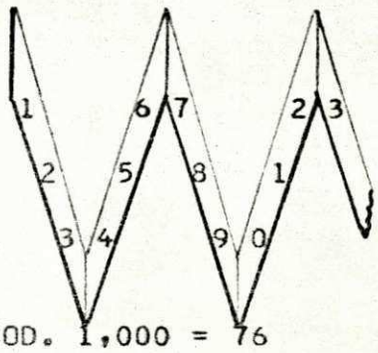
PARA MAIORES DETALHES, VER PRICE (39).

2.3.9-METODO DO DOBRAMENTO ("FOLDING"):

A CHAVE E DIVIDIDA EM UM CERTO NUMERO DE SECCOES, CADA UMA DAS QUAIS, EXCETO A ULTIMA, TEM UM COMPRIMENTO DA MESMA ORDEM DE GRANDEZA DAQUELES DOS ENDERECOS RELATIVOS DOS "BUCKETS" POSSIVEIS. PROCEDEMOS ENTAO COMO QUEM ESTA DOBRANDO UMA FOLHA DE PAPEL DE MODO A DEIXA-LA SANFONADA: TOMAMOS A 1A. SECCAO NORMALMENTE, A 2A. EM ORDEM INVERSA, A 3A. NORMALMENTE, A 4A. EM ORDEM INVERSA, E ASSIM POR DIANTE. EM SEGUIDA, FAZEMOS A SOMA (OU "EXCLUSIVE-OR", NUMA MAQUINA BINARIA) DE TODAS AS SECCOES, AJUSTANDO POR FIM O RESULTADO A FAIXA DE VARIACAO DOS ENDERECOS DE "BUCKETS".

COM OS MESMOS DADOS DO EXEMPLO ACIMA, TEREMOS:

I = | 123 | 456 | 789 | 012 | 3 |



1A. SECCAO: 123
2A. SECCAO: 654
3A. SECCAO: 789
4A. SECCAO: 210
5A. SECCAO: 300

SOMA: 2,076 MOD. 1,000 = 76

T(I) = 0.7*76 = 53

COMO O ANTERIOR, ESTE METODO E TAMBEM FREQUENTEMENTE USADO EM COMBINACAO COM OUTROS, QUANDO AS CHAVES TEM GRANDE COMPRIMENTO.
VER MAIORES DETALHES EM PRICE (39).

2.3.10-METODO DA DIVISAO POLINOMIAL OU DA CODIFICACAO ALGEBRICA:

CADA UM DOS N "BITS" (OU DIGITOS) DA CHAVE I, NUMA MAQUINA BINARIA (OU DECIMAL), E CONSIDERADO COMO UM COEFICIENTE DE UM POLINOMIO P(X), DE GRAU N-1. O POLINOMIO ASSIM OBTIDO E DIVIDIDO POR UM OUTRO POLINOMIO INVARIAVEL G(X) (DE GRAU K TAL QUE LOG2 (OU LOG10)M + 1 =< K =< N-1), PREVIA E ADEQUADAMENTE ESCOLHIDO SEGUNDO REGRAS ENCONTRAVEIS EM PETERSON (38). TODAS AS OPERACOES ARITMETICAS NESTA DIVISAO SAO FEITAS EM MODULO 2 (OU 10), SUPONDO-SE A MEMORIA BINARIA (OU DECIMAL). MATEMATICAMENTE, ISTO EQUIVALE A EFETUARMOS A DIVISAO POLINOMIAL SOBRE O CAMPO DE GALOIS ("GALOIS FIELD") COM 2 (OU 10) ELEMENTOS, COMUMENTE DESIGNADO, EM INGLES, POR GF(2) (OU GF(10)) . TOMAMOS OS K COEFICIENTES DO POLINOMIO-RESTO DESSA DIVISAO POLINOMIAL COMO SE FOSSEM OS "BITS" (OU DIGITOS) DE UM NUMERO E AJUSTAMOS ESTE NUMERO A FAIXA DE VARIACAO DOS ENDERECOS RELATIVOS DE "BUCKETS" POSSIVEIS, OBTENDO ASSIM T(I).
COMO EXEMPLO, SUPONHAMOS QUE TEMOS UM COMPUTADOR DECIMAL , O IDENTIFICADOR E I=123,456 (COM N=6), O NUMERO DE "BUCKETS " POSSIVEIS E M=700, ESCOLHEMOS K=3 E G(X) = X**3 + 2*X**2+ 3*X + 1. TEREMOS:

P(X) = 1*X**5 + 2*X**4 + 3*X**3 + 4*X**2 + 5*X + 6
P(X) / G(X) = X**2, COM RESTO 3*X**2 + 5*X + 6

I' = 356
T(I) = 700/10**3*356 = 0.7*356 = 249

PODEMOS NOTAR QUE ESTE METODO E, DE LONGE, O MAIS COMPLICADO AQUI APRESENTADO. NOTEMOS, NO ENTANTO, QUE ELE FOI PRO -

JETADO PARA IMPLEMENTACAO A NIVEL DE "HARDWARE" OU DE MICRO - PROGRAMACAO, E QUE SE ISSO FOR FEITO, ELE PODERA TER UMA PERFORMANCE BASTANTE RAZOAVEL, INCLUSIVE GARANTINDO, PARA G(X) ADEQUADAMENTE ESCOLHIDO, QUE SOMENTE CHAVES QUE DIFIRAM EM MAIS QUE UM CERTO NUMERO DE "BITS" PODERAO EVENTUALMENTE COLIDIR.

UMA DISCUSSAO BEM MAIS DETALHADA DESTE METODO PODE SER ENCONTRADA EM PETERSON (37 E 38), SCHAY & RAVER (41) E HANAN & PALERMO (14).

2.3.11-COMBINACAO DOS METODOS APRESENTADOS:

E BASTANTE COMUM QUE COMBINEMOS, NA PRATICA, DOIS OU MAIS DOS METODOS ACIMA.

COMO EXEMPLO 1, PODE SER QUE UM CONJUNTO DE CHAVES COM 6 DIGITOS TENHA UMA FORTE TENDENCIA DE TER O ULTIMO DIGITO IGUAL SO A 0 OU 5. NESTE CASO, APLICADO O METODO DE ANALISE DOS DIGITOS, DESPREZAMOS O ULTIMO DIGITO DE CADA CHAVE E DEPOIS APLICAMOS UM OUTRO METODO PARA OS 5 DIGITOS RESTANTES. OBSERVACAO: SE USARMOS SO O METODO DA DIVISAO, COM M PRIMO, NA PRATICA VERIFICAREMOS QUE A ANALISE DOS DIGITOS NAO MELHORARA A PERFORMANCE.

COMO EXEMPLO 2, SE TIVERMOS AS CHAVES COM UM COMPRIMENTO MUITO GRANDE, ANTES DE USARMOS OUTRO METODO DEVEREMOS USAR OU O METODO DO "SHIFTING" OU O DO "FOLDING".

2.4-COMPARACAO ENTRE OS METODOS DE "HASH"; ESCOLHA DE UM DELES.

2.4.1-COMPARACAO DO TEMPO DE CALCULO DOS METODOS:

INFELIZMENTE ESTE TIPO DE COMPARACAO DEPENDE INTEIRAMENTE DE CADA MAQUINA. EM MUITAS MAQUINAS ESTA COMPARACAO E IMPOSSIVEL, PORQUE OU O TEMPO NECESSARIO PARA A EXECUCAO DE CADA INSTRUCAO NAO FOI PUBLICADO, OU PORQUE, NA REALIDADE, ESSE TEMPO VARIA DE UMA EXECUCAO PARA OUTRA.

NAO OBSTANTE TUDO ISSO, PODEMOS NOS BASEAR NOS ESTUDOS JA FEITOS PARA ALGUMAS DAS MAQUINAS MAIS REPRESENTATIVAS. EM BOA

PARTE DOS CASOS, PODEMOS DIZER QUE, PROVAVELMENTE:

1.-O METODO DE "SHIFTING" E O MAIS RAPIDO (EMBORA DEVA SER QUASE SEMPRE USADO EM COMBINACAO COM OUTROS);

2.-OS METODOS DE DIVISAO E DE MULTIPLICACAO SAO TAMBEM BEM RAPIDOS (SUPONDO-SE QUE EXISTAM INSTRUCOES APROPRIADAS PARA ESSAS OPERACOES);

3.-O METODO "MID-SQUARE" E TAMBEM RAZOAVELMENTE RAPIDO, TOMANDO SO POUCO TEMPO A MAIS QUE OS ACIMA;

4.-O METODO DA ANALISE DOS DIGITOS, ALEM DE TOMAR BASTANTE TEMPO NA ANALISE PREVIA DAS CHAVES (E QUASE NUNCA SER USADO ISOLADAMENTE), NAO TEM O CALCULO TAO RAPIDO QUANTO OS ACIMA, EMBORA NAO SEJA CLASSIFICAVEL COMO UM METODO LENTO;

5.-O METODO DE "FOLDING" E CONSIDERADO COMO DE CALCULO JA RAZOAVELMENTE LENTO (ALEM DISSO, QUASE SEMPRE DEVE SER USADO EM COMBINACAO COM OUTROS, QUANDO O METODO DE "SHIFTING" PRODUZIRIA OS MESMOS RESULTADOS COM MAIOR RAPIDEZ);

6.-OS METODOS MAIS COMPLICADOS, TAIS COMO O DA MUDANCA DE BASE ("RADIX"), O DE LIN E O DA DIVISAO POLINOMIAL (OU DA CODIFICACAO ALGEBRICA), SAO OS DE CALCULO MAIS LENTO PARA AS CONDICOES NORMAIS, DEVENDO SEREM USADOS APENAS SE TIVERMOS CONSTRUIDO "HARDWARE" OU MICROPROGRAMACAO COM A FINALIDADE ESPECIFICA DE EFETUA-LOS.

2.4.2-COMPARACAO DO NUMERO DE COLISOES E/OU "BUCKET OVERFLOWS" DOS METODOS:

SUPONDO O ARQUIVO NA MEMORIA EXTERNA, ONDE O TEMPO DE CALCULO DA FUNCAO DE "HASH" E DE SOMENOS IMPORTANCIA, DIVERSOS PESQUISADORES REALIZARAM ESTUDOS TEORICOS E TAMBEM ESTUDOS EXPERIMENTAIS SOBRE GRANDES E DIVERSIFICADOS CONJUNTOS TIPICOS DE CHAVES, PROCURANDO DETERMINAR QUAL O METODO IDEAL. NUMA NOTAVEL SERIE DE ARTIGOS, LUM, YUEN, DODD & GHOSH (24, 25, 13 E 26) FIZERAM UMA PROFUNDA ANALISE DOS METODOS BASICOS DE CALCULO DE "HASH", DONDE PODEMOS EXTRAIR AS PRINCIPAIS CONCLUSOES:

1.-DOS METODOS COMUMENTE USADOS, OS MELHORES, NO SENTIDO DE MINIMIZAREM OS "BUCKET OVERFLOWS" (E CONSEQUENTEMENTE OS TEMPOS DE ACESSO AOS DISCOS), SAO OS METODOS DA DIVISAO E O RANDOMICO.

2.-O METODO DA DIVISAO E O MELHOR DE TODOS, SUPERANDO MESMO UM METODO RANDOMICO TEORICAMENTE PERFEITO. A EXPLICACAO PARA ISTO E QUE OS CONJUNTOS DE CHAVES FREQUENTEMENTE CONTEM GRUPOS DE CHAVES EM SEQUENCIA (COMO NUMA, NUMB, NUMC, ...). QUALQUER QUE SEJA O CODIGO USADO, QUER AS CHAVES SEJAM "LEFT" OU "RIGHT-JUSTIFIED", QUER AS CHAVES TENHAM OU NAO UM COMPRIMENTO TAL QUE TODOS OS SEUS CARACTERES CAIBAM EXATAMENTE NUMA PALAVRA DO COMPUTADOR, OS VALORES CORRESPONDENTES AS CHAVES DA SEQUENCIA FORMARAO UMA PROGRESSAO

ARITMETICA I, I + J, I + 2*J, ETC. SUPONDO-SE AGORA QUE O METODO DA DIVISAO SEJA USADO SEGUINDO TODAS AS RECOMEN - DACOES DE 2.3.1 (SENDO M PRIMO), ENTAO, SE T(I) = A, TEREMOS T(I + J) = A + J, T(I + 2*J) = A + 2*J,..., EM MODULO M. ASSIM, TEREMOS CERTEZA MATEMATICA DE QUE CHAVES EM PROGRESSAO ARITMETICA SERAO MAPEADAS EM ENDERECOS TAMBEM EM PROGRESSAO ARITMETICA E PORTANTO DISTINTOS (PARA O QUE BASTARIA QUE M E J FOSSEM PRIMOS ENTRE SI). DAI A VANTAGEM DO METODO DA DIVISAO SOBRE O RANDOMICO (MESMO QUE TEORICAMENTE PERFEITO), POIS NESTE AS CHAVES EM SEQUENCIA OU PROGRESSAO ARITMETICA PODEM COLIDIR TANTO QUANTO AS OUTRAS.

A FIGURA 2.1, ADAPTADA DE MARTIN (27), MOSTRA ALGUNS RE - SULTADOS TIPICOS EXTRAIDOS DO ESTUDO DE LUM, YUEN & DODD (24). AS LINHAS PONTILHADAS MOSTRAM O COMPORTAMENTO TEORICO DE UMA FUNCAO DE PERFEITA RANDOMIZACAO. OS PONTOS (TRIANGU -

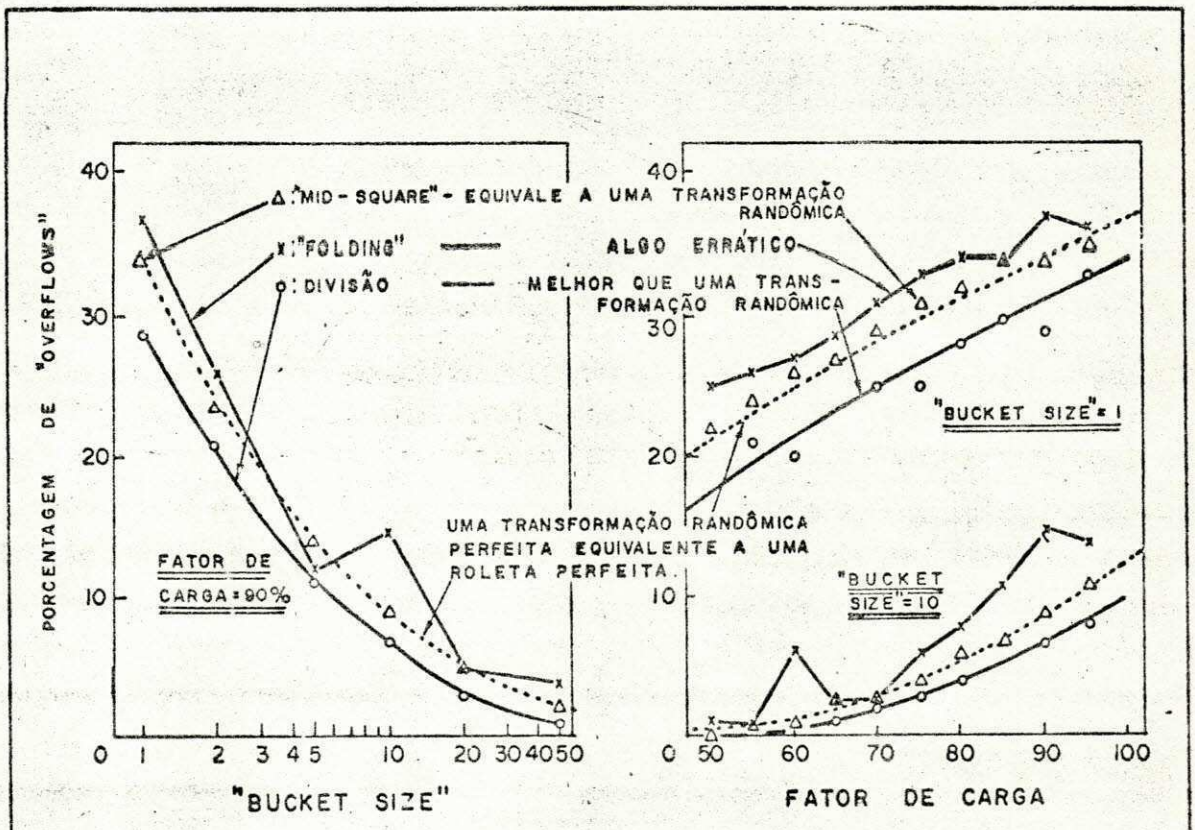


Fig - 2.1 - Comparação de três algorítmos de "hashing" bastante populares, com uma transformação perfeitamente randômica (figura retirada de Martin [27] e baseada nos experimentos de Lum, Yuen & Dodd [24]).

LOS, CIRCULOS E "X") MARCADOS CORRESPONDEM AS PORCENTAGENS MEDIAS DE "OVERFLOWS" OCORRENTES EM 3 METODOS COMUNS APLICA - DOS EM 8 GRANDES CONJUNTOS DE CHAVES BASTANTE TIPICOS E VA - RIADOS.

OBSERVAMOS QUE:

1.-O METODO "MID-SQUARE" E MUITO PROXIMO DE UMA FUNCAO TEORI- CA DE PERFEITA RANDOMIZACAO;

2.-O METODO DA DIVISAO, COMO ERA DE SE ESPERAR, MANTEM SEMPRE UM DESEMPENHO MELHOR QUE OS OUTROS;

3.-O METODO DO "FOLDING" (COMO TAMBEM O DO "SHIFTING") E DE PERFORMANCE IRREGULAR, IMPREVISIVEL (PROVAVELMENTE DEVIDO A DISTRIBUICAO NAO UNIFORME DE CARACTERES NOS CONJUNTOS DE CHAVES), POREM QUASE SEMPRE PIOR QUE O DA DIVISAO;

4.-OS METODOS MAIS COMPLICADOS, TAIS COMO O DE MUDANCA DE BA- SE ("RADIX"), O DE LIN E O DA DIVISAO POLINOMIAL (OU CODI- FICACAO ALGEBRICA), TAMBEM TEM UMA PERFORMANCE GERALMENTE INFERIOR AQUELA DO METODO DA DIVISAO, PORQUE O COMPORTA - MENTO DAQUELES METODOS APROXIMA-SE DO COMPORTAMENTO DE UM GERADOR IMPERFEITO DE NUMEROS PSEUDO-ALEATORIOS;

5.-COMO MARTIN (27) OBSERVOU, O METODO IDEAL NAO E AQUELE QUE DISTRIBUI O CONJUNTO 'IDEAL' DE CHAVES RANDOMICAMENTE, MAS SIM AQUELE QUE DISTRIBUI O CONJUNTO 'REAL' DE CHAVES UNIFORMEMENTE, ATRAVES DE TODO O ARQUIVO.

- EM RESUMO, QUANDO O TEMPO DE ACESSO E RELATIVAMENTE GRANDE E E FATOR MAIS IMPORTANTE QUE O TEMPO DE CALCULO, OS ESTUDOS DE LUM, YUEN, DODD & GHOSH NOS INDICAM QUE QUASE SEMPRE DEVE- MOS USAR O METODO DA DIVISAO, MESMO QUE O TEMPO DE CALCULO SEJA UM POUCO MAIOR, MESMO QUE TENHAMOS QUE REPETIR VARIAS VEZES UMA MACRO (POR EXEMPLO, PARA FAZER DIVISAO DECIMAL NUMA MAQUINA COMO O IBM 370, QUE TEM OS ENDERECOS DE DISCO DECI - MAIS MAS SO TEM INSTRUCAO DE DIVISAO BINARIA) OU MESMO QUE TENHAMOS QUE CONTORNAR OUTROS PEQUENOS PROBLEMAS DESSE TIPO.

OBVIAMENTE, O METODO MULTIPLICATIVO, SENDO UMA GENERALIZA- CAO DO METODO DA DIVISAO, EQUIPARA-SE A ESTE, DESDE QUE ADE - QUADAMENTE EMPREGADO.

VER MAIORES DETALHES NOS ARTIGOS AQUI CITADOS (13,24,25 E 26).

2.4.3-COMPARACAO DOS CUSTOS GLOBAIS DOS METODOS:

EVIDENTEMENTE ESTA COMPARACAO DOS CUSTOS GLOBAIS E DIFICI- LIMA, DE GENERALIZACAO QUASE IMPOSSIVEL, RESTANDO MUITO A FA- ZER NESTA AREA (AO MENOS PARA PROVAR AQUILO DE QUE JA TEMOS QUASE CERTEZA, POREM NAO TEMOS PROVAS).

TRABALHOS RELATIVAMENTE RECENTES, DE VAN DER POOL (47 E 48) E WEBB (49) INCLUIRAM PARAMETROS ADICIONAIS AOS QUE JA TINHAMOS CONSIDERADO, NA DETERMINACAO DA PERFORMANCE DOS ME -

TODOS.

VAN DER POOL CONSIDEROU:

1.-CUSTO DE MEMORIA PARA O ARQUIVO TOTAL;
2.-CUSTO DE MEMORIA PARA 1 REGISTRO POR UNIDADE DE TEMPO;
3.-CUSTO DE ACESSO A AREA PRIMARIA DO ARQUIVO;
4.-CUSTO DOS ACESSOS ADICIONAIS.

DESSE MODO DERIVOU VARIAS FORMULAS PARA ESTIMAR O CUSTO GLOBAL, COM E SEM INSERCOES E DELECOES, MAS RESTRINGINDO-SE SEMPRE AO USO DE AREA EM SEPARADO PARA "OVERFLOWS". DE ACORDO COM ESSES ESTUDOS, FATORES DE CARGA PRIMARIOS MAIORES QUE 1 PODEM TER UM CUSTO GLOBAL MENOR QUE FATORES DE CARGA PRIMARIOS MENORES QUE 1, PARTICULARMENTE ENQUANTO TOMARMOS EM CONTA A MEMORIA DESPERDICADA E ENQUANTO O CUSTO DA MEMORIA TIPO DASD AINDA FOR RELATIVAMENTE ALTO.

WEBB INCLUIU O TEMPO DE CPU NA SUA AVALIACAO DOS ESQUEMAS DE "HASH", COMBINANDO OS VARIOS METODOS DE CALCULO DA FUNCAO DE "HASHING" COM OS VARIOS METODOS DE RESOLUCAO DE "BUCKET OVERFLOWS", DETERMINANDO UM CUSTO GLOBAL. E IMPORTANTE ASSINALARMOS QUE SEUS ESTUDOS CONCORDARAM BASICAMENTE COM OS RESULTADOS OBTIDOS EXPERIMENTALMENTE POR LUM, YUEN & DODD (24).

2.4.4-A ESCOLHA DA FUNCAO DE "HASH" NUMA APLICACAO PRATICA:

OBVIAMENTE, A "MELHOR" MANEIRA DE ESCOLHERMOS UMA FUNCAO DE "HASH" SERIA TOMARMOS UMA AMOSTRA A MAIOR POSSIVEL DAS CHAVES QUE REALMENTE TEREMOS NO ARQUIVO EM QUESTAO, SIMULANDO EM SEGUIDA (SE NECESSARIO PARA VARIOS FATORES DE CARGA E "BUCKET SIZES") O COMPORTAMENTO CORRESPONDENTE AS VARIAS FUNCOES E PARAMETROS (TAIS COMO M NO METODO DA DIVISAO) POSSIVEIS. PARA CADA CONJUNTO FUNCAO-PARAMETRO TOMADO, FARIAMOS UMA SIMULACAO TAO PROXIMA DA REALIDADE QUANTO POSSIVEL, E QUE RESULTASSE EM INDICES DE "OVERFLOWS" (POR "BUCKET", NO TOTAL E EM MEDIA) E DO NUMERO DE PESQUISAS DE "SLOTS" NECESSARIAS (POR "SLOT", POR "BUCKET", NO TOTAL E EM MEDIA).

ESSA MANEIRA DE ESCOLHER A FUNCAO VIA EXPERIMENTOS PRATICOS SOBRE O CONJUNTO PARTICULAR DE CHAVES E INDUBITAVELMENTE A MAIS SEGURA. NO ENTANTO E EXAUSTIVA E SO E USADA EM CASOS ESPECIALISSIMOS, POIS QUASE SEMPRE RESPONDIDAS AS 7 QUESTOES DE 2.2 E APROVEITADOS OS MUITOS ESTUDOS E GENERALIZACOES TEORICO-EXPERIMENTAIS EXISTENTES, PODEMOS FIXAR NOSSA ATENCAO EM 1 OU NO MAXIMO 2 METODOS DE CALCULO DA FUNCAO DE "HASH" (QUASE SEMPRE ENVOLVENDO O METODO DA DIVISAO) SOBRE OS QUAIS FAREMOS APENAS DE 1 A 5 EXPERIMENTOS SIMPLIFICADOS, COM CERTOS PARAMETROS ESCOLHIDOS SEGUNDO AS REGRAS PRAGMATICAS EXISTENTES. A FINALIDADE DESSE(S) EXPERIMENTO(S) E APENAS VERIFICAR SE O COMPORTAMENTO DO CONJUNTO FUNCAO-PARAMETROS-CHAVES E O ESPERADO, SEM QUALQUER "SURPRESA" DESAGRADAVEL.

NA SELECAO DA FUNCAO DE "HASH", DEVEREMOS USAR BASTANTE DO NOSSO BOM SENSO E PERSPICACIA. COMO EXEMPLO 1, SE USARMOS O METODO DA DIVISAO COM M = 2**K (K =< 16), SENDO AS CHAVES ALFANUMERICAS E TODAS ELAS AJUSTADAS A ESQUERDA ("LEFT-JUSTIFIED"), E COMPLETADAS COM BRANCOS (OU ZEROS) A DIREITA ("BLANK FILLED"), SUPOSTOS 8 "BITS" POR CARACTER E 8 CARACTERES POR PALAVRA, ENTAO TODAS AS CHAVES COM COMPRIMENTO MENOR OU IGUAL QUE 8 - 2 = 6 CARACTERES TERAO O MESMO "HOME ADDRESS", QUE SERA DADO SIMPLESMENTE PELOS K "BITS" MAIS A DIREITA... COMO EXEMPLO 2, A SERIE DE COMPUTADORES CDC-6000 NAO TEM INSTRUCAO DE DIVISAO EM PONTO FIXO, MAS TEM INSTRUCAO DE MULTIPLICACAO EM PONTO FIXO, COM OS 48 "BITS" MAIS A DIREITA RESERVADOS COMO MANTISSA, DE MODO QUE SE USARMOS O METODO "MID-SQUARE", TODOS OS IDENTIFICADORES DE 1 OU 2 CARACTERES (QUE SAO "LEFT-JUSTIFIED" E "BLANK-FILLED") TERAO OS MESMOS ULTIMOS 48 "BITS"; PROBLEMAS SEMELHANTES OCORRERIAM NESSE COMPUTADOR COM QUALQUER DOS METODOS MULTIPLICATIVOS, SE ANTES DA MULTIPLICACAO DE CADA CHAVE NAO FIZESSEMOS UM DESLOCAMENTO DA MESMA PARA A DIREITA ("SHIFT RIGHT"). EM QUALQUER SITUACAO DE FUNCAO-CHAVES-MAQUINA, PRECISAMOS ESTAR ALERTAS PARA QUE FATOS SEMELHANTES NAO NOS APANHEM DESPREVENIDOS.

TOMADAS TODAS AS PRECAUCOES DE QUE TEMOS FALADO, O METODO DA DIVISAO E O ESCOLHIDO NA MAIORIA DOS CASOS, COM O DIVISOR M PRIMO (OU NAO FATORAVEL POR NUMEROS MENORES QUE 20), MESMO QUE ESSA ESCOLHA IMPLIQUE NO USO DE OUTROS METODOS EM COMBINACAO COM O DA DIVISAO, NO USO DE MACROS OU OUTROS ARTIFICIOS ESPECIAIS .

3 - METODOS DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS"

DOS 4 FATORES DETERMINANTES DA EFICIENCIA DO "HASHING", APONTADOS EM 1.4, A ESCOLHA DO METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS" TALVEZ SEJA O MAIS PROBLEMATICO. NO PRESENTE CAPITULO ABORDAREMOS ESTE ASSUNTO SEM USO DE INDEX E PARA MEMORIAS NAO PAGINADAS, E, NO PROXIMO CAPITULO, COM USO DE INDEX E DEPOIS PARA MEMORIAS PAGINADAS.

COMO VIMOS EM 1.4.4, OS METODOS DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS" CLASSIFICAM-SE, DE UM MODO GENERICO, EM:

1.-QUANTO A AREA DE RESOLUCAO DAS COLISOES E/OU "BUCKET OVERFLOWS":
   A) METODOS SEM AREA DE "OVERFLOW" EM SEPARADO. AS COLISOES E/OU "BUCKET OVERFLOWS" SAO RESOLVIDOS LIVREMENTE, ATRAVES DE TODO O ARQUIVO. TEM DESEMPENHOS MELHORES EM ESPACO E PIORES EM TEMPO, QUANDO COMPARADOS COM OS ABAIXO.
   B) METODOS COM AREA DE "OVERFLOW" EM SEPARADO. AS COLISOES E/OU "BUCKET OVERFLOWS" SAO RESOLVIDOS EXCLUSIVAMENTE EM CERTAS AREAS ESPECIFICAMENTE DESTINADAS PARA ESTE FIM. TEMOS DUAS ALTERNATIVAS: USAR UMA SO AREA DE "OVERFLOW" PARA TODO O ARQUIVO, OU VARIAS (DIGAMOS, UMA POR CADA PAGINA, OU TRILHA, OU VOLUME...).

2.-QUANTO AO USO DE APONTADORES:
   A) METODOS QUE USAM APONTADORES. FORMAM LISTAS ENCADEADAS, COMPOSTAS OU DE "BUCKETS" (EM MEMORIA SECUNDARIA) OU DE REGISTROS LOGICOS (EM MEMORIA PRINCIPAL OU EM MEMORIA PAGINADA), DE MODO QUE, PARA ENCONTRARMOS UM REGISTRO, DETERMINAMOS INICIALMENTE SEU "HOME ADDRESS" PARA DEPOIS IRMOS PESQUISANDO A LISTA ASSOCIADA AO MESMO, ATE ENCONTRARMOS A CHAVE PESQUISADA OU CHEGARMOS AO FI-

NAL DA LISTA. NECESSITAM DE UM POUCO MAIS DE ESPACO (PARA OS APONTADORES), MAS TEM MELHORES DESEMPENHOS EM TEMPO, PRINCIPALMENTE SE O "BUCKET SIZE" E IGUAL A 1 (USUALMENTE EM MEMORIA PRINCIPAL OU PAGINADA), OU E RELATIVAMENTE PEQUENO (DIGAMOS, MENOR QUE 5, OU MESMO 10) E O FATOR DE CARGA PRIMARIO E RELATIVAMENTE GRANDE (DIGAMOS, MAIOR QUE 0.90, OU MESMO 0.80).

B) METODOS QUE NAO USAM APONTADORES. RESOLVEM AS COLISOES E/OU "BUCKET OVERFLOWS" ATRAVES DA DETERMINACAO (POR CERTOS ALGORITMOS) DUMA SEQUENCIA DE ENDERECOS QUE DEVEREMOS PESQUISAR EM SUCESSAO AO "HOME ADDRESS". ESTA CLASSE GERAL DE METODOS FOI CHAMADA DE METODOS DE ENDERECAMENTO ABERTO ("OPEN ADDRESSING") POR PETERSON (37).

3.-QUANTO AO USO DE INDICES:

A) METODOS QUE USAM UM INDEX. A FUNCAO DE "HASH" REFERENCIA UMA POSICAO DO INDEX, A QUAL CONTEM UM APONTADOR PARA UM REGISTRO NO ARQUIVO, LOCAL EM QUE A PESQUISA PROPRIAMENTE DITA COMECARA A SER FEITA. ESTUDAREMOS ESSES METODOS NO PROXIMO CAPITULO (CAP. 4).

B) METODOS QUE NAO USAM UM INDEX. A FUNCAO DE "HASH" REFERENCIA DIRETAMENTE UM REGISTRO OU "BUCKET" DO ARQUIVO, POR ONDE A PESQUISA COMECARA. ESTUDAREMOS ESSES METODOS NO PRESENTE CAPITULO (CAP. 3).

3.1 - INTRODUCAO. ALGORITMOS GENERICOS PARA RESOLUCAO DE COLISOES E "BUCKET OVERFLOWS".

MUITAS VEZES CONFUNDIMOS OS CONCEITOS DE COLISAO E DE "BUCKET OVERFLOW". ESTES FENOMENOS SURGEM DE MANEIRAS BASTANTE SEMELHANTES, E NA VERDADE TEM MUITAS ANALOGIAS, MAS A RIGOR SAO DIFERENTES. ESTABELECAMOS, SEGUNDO MAURER & LEWIS (42), UMA DIFERENCA BEM DELINEADA ENTRE OS DOIS CONCEITOS:

1.-O TERMO 'COLISAO' APLICA-SE EM SITUACOES EM QUE O "BUCKET SIZE" E IGUAL A 1, O QUE GERALMENTE ACONTECE PARA ARQUIVOS NA MEMORIA PRINCIPAL OU EM MEMORIA PAGINADA. ASSIM, UM PROBLEMA DE COLISAO OCORRE QUANDO ESTAMOS SUPONDO QUE, NA MAIORIA DAS VEZES, TODAS AS NOSSAS CHAVES TEM "HOME ADDRESSES" DISTINTOS. QUANDO DOIS IDENTIFICADORES DISTINTOS TEM O MESMO "HOME ADDRESS", ENTAO APELAMOS PARA UM METODO

DE RESOLUCAO DE COLISOES.

2.-O TERMO "BUCKET OVERFLOW" APLICA-SE EM SITUACOES EM QUE O "BUCKET SIZE", N, E MAIOR QUE 1, O QUE GERALMENTE ACONTECE PARA ARQUIVOS NA MEMORIA SECUNDARIA. ASSIM, UM PROBLEMA DE "BUCKET OVERFLOW" OCORRE QUANDO ESTAMOS SUPONDO QUE, NA MAIORIA DAS VEZES, O NUMERO DE CHAVES QUE TEM O MESMO "HOME ADDRESS" NAO EXCEDE O "BUCKET SIZE", N. NOS POUCOS CASOS EM QUE TEMOS UM "HOME ADDRESS" CORRESPONDENDO A MAIS QUE N IDENTIFICADORES, ENTAO APELAMOS PARA UM METODO DE RESOLUCAO DE "BUCKET OVERFLOWS". VER 3.1.2.

VIMOS QUE OS PROBLEMAS DE COLISAO E "BUCKET OVERFLOW" SAO, A RIGOR, DIFERENTES. NO ENTANTO, HAVENDO MUITAS SEMELHANCAS E ANALOGIAS ENTRE OS DOIS PROBLEMAS, POR BREVIDADE USAMOS , AS VEZES, APENAS UM QUALQUER DOS DOIS TERMOS, COM O SENTIDO GERAL ABRANGIDO POR AMBOS.

PODEMOS ENCARAR O PROBLEMA DE COLISAO COMO UM CASO PARTICULAR DO PROBLEMA DE "BUCKET OVERFLOW", ONDE O "BUCKET SIZE" E IGUAL A 1. PORISSO APRESENTAREMOS NESTE TRABALHO APENAS OS ALGORITMOS DOS METODOS DE RESOLUCAO DE "BUCKET OVERFLOWS" , DEIXANDO AO LEITOR O PEQUENO E IMEDIATO TRABALHO DE FAZER AS POUCAS ADAPTACOES NECESSARIAS PARA OS CASOS DE RESOLUCAO DE COLISOES. EXAMINAR, COMPARATIVAMENTE, OS ALGORITMOS DE 3.1.2 E 3.1.1, NESTA ORDEM.

3.1.1-ALGORITMO A:
PROCEDIMENTO GENERICO PARA PESQUISA E INSERCAO DE UM REGISTRO DE IDENTIFICADOR I, SEM USO DE "BUCKETS", USUALMENTE EM MEMORIA PRINCIPAL OU MEMORIA PAGINADA, ADMITINDO-SE COLISOES.

A.1 - CALCULE O "HOME ADDRESS" T(I) DO REGISTRO, POR QUALQUER QUE SEJA A FUNCAO DE "HASH" QUE TENHA SIDO ADOTADA.

A.2 - (ESTE ENDERECO CONTEM O REGISTRO PESQUISADO ?): SE SIM, OBTEVE-SE SUCESSO: O REGISTRO FOI ENCONTRADO E PODE SER RECUPERADO(MAS, SE SE ESPERAVA FAZER UMA INSERCAO, COMETEU-SE UM ERRO). FIM.

A.3 - SE NAO:

A.3.1-(EXISTE ALGUM REGISTRO VALIDO, DIFERENTE DO PESQUISADO, NESTE ENDERECO ?):
SE SIM, CALCULE O PROXIMO ENDERECO "SUBSEQUENTE" DESTE, SEGUINDO QUALQUER PROCEDIMENTO QUE TENHA SIDO ADOTADO PARA GERAR UMA SEQUENCIA DE ENDERECOS, COM O FIM DE RESOLVER AS COLISOES.

A.3.1.1-(ESTE NOVO ENDERECO "SUBSEQUENTE" COINCIDE COM O "HOME ADDRESS", OU COM OUTRO ENDERECO QUE JA PESQUISAMOS, OU SIMPLESMENTE E UMA CONVENCAO DE FIM

DE ESPACO DISPONIVEL ?):
SE SIM, OBTEVE-SE INSUCESSO E O AR - QUIVO ESTA CHEIO. FIM.

A.3.1.2-SE NAO, VA PARA A.2.

A.3.2-SE NAO, OBTEVE-SE INSUCESSO: O REGISTRO PESQUISADO NAO FOI ENCONTRADO. EM ALGUMAS APLICACOES ISTO E SIMPLESMENTE ASSINALADO, MAS AQUI SUPOEM-SE QUE A INSERCAO E DESEJADA EM TAL SITUACAO, PORISSO INSIRA O REGISTRO PESQUISADO NESTE ENDERECO ("SUBSEQUENTE"). FIM.

3.1.2-ALGORITMO B:

PROCEDIMENTO GENERICO PARA PESQUISA E INSERCAO DE UM REGISTRO DE IDENTIFICADOR I, COM USO DE "BUCKETS", USUALMENTE EM MEMORIA SECUNDARIA, ADMITINDO-SE "BUCKET OVERFLOWS".

B.1 - CALCULE O "HOME ADDRESS" T(I) DO REGISTRO, POR QUALQUER QUE SEJA A FUNCAO DE "HASH" QUE TENHA SIDO ADOTADA. AQUI O "HOME ADDRESS" REFERE-SE A UM "BUCKET".

B.2 - (ESTE ENDERECO DE "BUCKET" CONTEM O REGISTRO PESQUISADO ?
OBS: USUALMENTE ESSA PESQUISA DENTRO DO "BUCKET" E FEITA SEQUENCIALMENTE):
SE SIM, OBTEVE-SE SUCESSO: O REGISTRO FOI ENCONTRADO E PODE SER RECUPERADO (MAS, SE SE ESPERAVA FAZER UMA INSERCAO, COMETEU-SE UM ERRO). FIM.

B.3 - SE NAO:

B.3.1-(O "BUCKET" ESTA COMPLETAMENTE CHEIO ?):SE SIM, CALCULE O PROXIMO ENDERECO DE "BUCKET" "SUBSEQUENTE" A ESTE, SEGUINDO QUALQUER PROCEDIMENTO QUE TENHA SIDO ADOTADO PARA GERAR UMA SEQUENCIA DE ENDERECOS, COM O FIM DE RESOLVER OS "BUCKET OVERFLOWS".

B.3.1.1-(ESTE NOVO ENDERECO "SUBSEQUENTE" COINCIDE COM O "HOME ADDRESS", OU COM OUTRO ENDERECO QUE JA PESQUISAMOS, OU SIMPLESMENTE E UMA CONVENCAO DE FIM DE ESPACO DISPONIVEL ?):
SE SIM, OBTEVE-SE INSUCESSO E O ARQUIVO ESTA CHEIO. FIM.

B.3.1.2-SE NAO, VA PARA B.2

B.3.2-SE NAO, OBTEVE-SE INSUCESSO: O REGISTRO PESQUISADO NAO FOI ENCONTRADO. EM ALGUMAS APLICACOES ISTO E SIMPLESMENTE ASSINALADO, MAS AQUI SUPOE-SE QUE A INSERCAO E DESEJADA EM TAL SITUACAO, PORISSO INSIRA O REGISTRO PESQUISADO NO "SLOT" APROPRIADO QUE ESTIVER DISPONIVEL DENTRO DO "BUCKET".

(USUALMENTE O PRIMEIRO "SLOT" DISPONIVEL). FIM.

3.1.3-QUESTOES A CONSIDERAR NA ESCOLHA DE UM METODO DE RESOLUCAO DE "BUCKET OVERFLOWS" (REFERENCIA 42):

1.- SUPONHAMOS QUE ACABAMOS DE TER ACESSO AO REGISTRO DE ENDERECO ALFA. QUAL O TEMPO NECESSARIO PARA SE TER ACESSO A UM OUTRO REGISTRO DE ENDERECO BETA? E, MAIS IMPORTANTE, COMO ESTE TEMPO DEPENDE DA POSICAO DE BETA EM RELACAO A ALFA?

EXEMPLO 1: SE ALFA E BETA FOREM ENDERECOS DE UM DISCO DE CABECOTE MOVEL, ESSE TEMPO SERA BEM MENOR SE ELES ESTIVEREM NUM MESMO CILINDRO, PORQUE O BRACO DE ACESSO NAO TERA QUE SE MOVER. O MESMO NAO ACONTECERA SE O DISCO FOR DO TIPO "HEAD PER TRACK".

EXEMPLO 2: SE BETA FOR IMEDIADAMENTE SUBSEQUENTE A ALFA NUMA TRILHA DE UM DISCO, SERA ALGO MAIS RAPIDO CHEGAR A BETA PARTINDO DE ALFA DO QUE PARTIR DE BETA PARA CHEGAR A ALFA SOMENTE APOS UMA VOLTA QUASE QUE COMPLETA PELA TRILHA.

2.- E IMPORTANTE O TEMPO GASTO PARA CALCULARMOS O ENDERECO "SUBSEQUENTE" AQUELE EM QUE ESTAMOS ?

SE NOSSOS REGISTROS ESTIVEREM NA MEMORIA PRINCIPAL A RESPOSTA E 'SIM', PORQUE A VANTAGEM QUE UM METODO DE RESOLUCAO DE COLISOES POSSA TER SOBRE OUTRO (EM REDUZIR O NUMERO DE ACESSOS NECESSARIOS) PODE SER SUPLANTADA PELO AUMENTO DO TEMPO DE CALCULO DE CADA UM DOS ENDERECOS SUBSEQUENTES.

SE NOSSOS REGISTROS ESTIVEREM NA MEMORIA SECUNDARIA, A RESPOSTA QUASE QUE COM CERTEZA SERA 'NAO', POIS O TEMPO DE CALCULO SERA GERALMENTE DESPREZIVEL EM RELACAO AO TEMPO NECESSARIO PARA UM ACESSO.

3.- QUAO BOM E O DESEMPENHO DO NOSSO METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS" PARA AS CONDICOES PARTICULARES DA APLICACAO ?

UMA VEZ DEFINIDAS AS CONDICOES PARTICULARES DA APLICACAO (ENTRE AS QUAIS TEMOS O "BUCKET SIZE", O FATOR DE CARGA, A FUNCAO DE "HASHING", AS CARACTERISTICAS ESPECIAIS DO CONJUNTO DE CHAVES, DO MEIO DE ARMAZENAMENTO, ETC.), NOS PODEMOS MELHORAR A PERFORMANCE DO NOSSO PROCEDIMENTO GLOBAL DE ARMAZENAMENTO E RECUPERACAO, PELA ESCOLHA DE UM BOM METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS". EM PARTICULAR, SE EXISTE UM CERTO NUMERO DE SINONIMOS DE UM PARTICULAR REGISTRO, CERTOS METODOS NOS PERMITEM ENCONTRAR A MAIORIA DESSES SINONIMOS CALCULANDO APENAS UM ENDERECO SUBSEQUENTE EM CADA CASO. POR OUTRO LADO, UM FENOMENO CONTRAPOSTO E O 'AMONTOAMENTO' ("CLUSTERING"), NO QUAL CERTOS REGISTROS NAO SAO SINONIMOS MAS TEM OS "HASH CODES"

EM SEQUENCIA, DE MANEIRA QUE, SE ADOTARMOS A PESQUISA LI - NEAR PARA A RESOLUCAO DE COLISOES, ELES OCUPARAO AS MESMAS PORCOES DO ARQUIVO QUE OCUPARIAM SE FOSSEM SINONIMOS, DE - GRADANDO UM POUCO A EFICIENCIA DO METODO.

3.2 - METODOS SEM AREA DE "OVERFLOW" EM SEPARADO

3.2.1-PESQUISA LINEAR OU POR DERRAMAMENTO ("OPEN OVERFLOW", "LINEAR SEARCH" OU "CONSECUTIVE SPILL"):

CONSISTE EM ARMAZENARMOS UM REGISTRO, QUE TRANSBORDOU, NO PRIMEIRO "SLOT" DISPONIVEL (OU "OPEN") DO PROXIMO "BUC- KET" NAO TOTALMENTE CHEIO. "PROXIMO" SIGNIFICA O ENDERECO DE "BUCKET" QUE SE SEGUE SEQUENCIALMENTE NO ESPACO DE EN - DERECAMENTO, CONSIDERADO CICLICO (OU CIRCULAR), ONDE AO " ULTIMO" ENDERECO DE "BUCKET" SEGUE-SE O "PRIMEIRO" DELES. NOTEMOS QUE NUM IBM/370, POR EXEMPLO, A PROXIMA POSICAO DE UM REGISTRO (OU "BUCKET") EM NOSSO ARQUIVO, APOS A POSICAO DE ENDERECO END, PODE TER O ENDERECO END+64, AO INVES DE END+1, DESDE QUE O REGISTRO (OU "BUCKET") TENHA O COMPRI - MENTO DE 64 "BYTES". AS ADAPTACOES NECESSARIAS EM NOSSOS ALGORITMOS, PARA CUIDAREM DISSO, SAO TAO TRIVIAIS QUE AS DEIXAREMOS TOTALMENTE A CARGO DO LEITOR.

SENDO I O IDENTIFICADOR DO REGISTRO PESQUISADO, T(I) A TRANSFORMACAO CHAVE-ENDERECO, HAVENDO M "BUCKETS" NO AR - QUIVO, NUMERADOS COMO B(0), B(1), B(2), ..., B(M-1), ENTAO A SEQUENCIA DE ENDERECOS RELATIVOS DE "BUCKETS", GERADA PELA ETAPA 3.1 DO ALGORITMO GENERICO B (EM 3.1.2) SERA:
T(I), T(I)+1, T(I)+2, ..., B(M-1), B(0), B(1),..., T(I)-1

CHAMAMOS DE 'DESLOCAMENTO' DE UM REGISTRO O NUMERO DE " BUCKETS" DE "OVERFLOW" QUE PERCORREMOS ANTES DE LOCALIZAR OU ARMAZENAR UM REGISTRO.

COMO UM EXEMPLO DE DETALHAMENTO DO ALGORITMO B PARA UM QUALQUER METODO DE RESOLUCAO DE "BUCKET OVERFLOWS", APRE - SENTAMOS O ALGORITMO ABAIXO:

- ALGORITMO C:

PESQUISA E INSERCAO DE UM REGISTRO DE IDENTIFICADOR I, RESOLVENDO OS "BUCKET OVERFLOWS" POR PESQUISA LINEAR.

```
   ARGUMENTOS DE ENTRADA:
       I E O IDENTIFICADOR DO REGISTRO PESQUISADO;
       NRGBKT E O NUMERO DE REGISTROS POR "BUCKET" (2, NA FI-
              GURA 3.1);
       M E O NUMERO DE "BUCKETS" NO ARQUIVO (10, NA FIGURA 3.
              1), USUALMENTE PRIMO.
   OUTROS SIMBOLOS:
       T(I) E A FUNCAO DE "HASH" ADOTADA, TAL QUE 0<=T(I)
              <=M-1; USUALMENTE T(I)=I MOD. M;
       IBUCKT E O APONTADOR PARA O "BUCKET" SENDO EXAMINADO;
       INICBK E O APONTADOR PARA O REGISTRO INICIAL DO "BUC -
              KET" SENDO EXAMINADO;
       B(J), 0<=J<=M-1 SAO OS "BUCKETS";
       R(J), 0<=J<=M*NRGBKT-1 SAO OS REGISTROS.
   ARGUMENTOS DE RETORNO:
       IREGI E O APONTADOR PARA O REGISTRO SENDO EXAMINADO
              (VARIA DE 0 A NRGBKT*M-1);
       FOUND = ".TRUE." SE A CHAVE FOI ENCONTRADA;
             = ".FALSE." SE A CHAVE NAO FOI ENCONTRADA.

      CONDICAO INICIAL: AO INICIALIZARMOS O ARQUIVO, POR AL -
   GUMA CONVENCAO MARCAMOS TODOS OS REGISTROS COMO DESOCUPA -
   DOS.

C.1 - (FAZ O "HASH")
      IHASH=T(I)
      IBUCKT=IHASH
C.2 - (ESTE "BUCKET" CONTEM O REGISTRO PESQUISADO ?):
      INICBK=IBUCKT*NRGBKT
      IREGI=INICBK
      C.2.1-(O REGISTRO R(IREGI) ESTA DESOCUPADO ?):
            SE SIM, VA PARA C.3.2.
            SE NAO:
      C.2.2-(O REGISTRO R(IREGI) TEM A CHAVE IGUAL A I ?):
            SE SIM, OBTEVE-SE SUCESSO. FOUND = ".TRUE.". FIM.
            SE NAO:
      C.2.3-IREGI=IREGI+1;
            SE IREGI<=INICBK+(NRGBKT-1) VA PARA C.2.1.
            SE NAO:
C.3 - (O "BUCKET" NAO CONTEM O REGISTRO PESQUISADO):
      C.3.1-(O "BUCKET" ESTA CHEIO.     CALCULA   O   PROXIMO
            "BUCKET"):
            IBUCKT=IBUCKT+1. (SE IBUCKT>M-1 FACA IBUCKT=0).
            C.3.1.1-(ESTE NOVO "BUCKET" COINCIDE COM O "HOME
                  BUCKET" ?):
            SE IBUCKT=IHASH ENTAO O ARQUIVO ESTA CHEIO. ALEM
                            DE TERMOS INSUCESSO,    ISTO   E,
                            FOUND=".FALSE.". FIM.
            C.3.1.2-SE NAO, VA PARA C.2.
```

C.3.2-[O "BUCKET" TEM UM "SLOT" VAZIO, ONDE ARMAZENA-SE O REGISTRO]:
OBTEVE-SE INSUCESSO, ISTO E, FOUND=".FALSE." ;
MARQUE O REGISTRO R[IREGI] COMO OCUPADO E ARMAZENE O REGISTRO QUE ESTAVA PESQUISANDO. FIM.

COMO EXEMPLO, CONSIDERAMOS UM ARQUIVO DE EMPREGADOS CUJOS REGISTROS LOGICOS TEM 3 CAMPOS (NOME, NUMERO DE MATRICULA, HABILIDADE), ESCOLHENDO-SE COMO CHAVE O CAMPO "NUMERO DE MATRICULA". A FIGURA 3.1 ILUSTRA O FUNCIONAMENTO DE UMA FUNCAO DE "HASH" POR DIVISAO, EM CONJUNTO COM O METODO DE PESQUISA LINEAR PARA RESOLUCAO DE "BUCKET OVERFLOWS", SUPONDO-SE QUE OS REGISTROS APARECERAM COMO ARGUMENTOS DE ENTRADA DO ALGORITMO C NA SEGUINTE ORDEM: 12, 31, 61, 65, 09, 06, 20, 18, 26, 49, 44, 17, 15, 24, E 67. O ARQUIVO E COMPOSTO DE 10 "BUCKETS", CADA UM COM 2 "SLOTS". NOTEMOS QUE OS IDENTIFICADORES 24 E 44 SAO SINONIMOS, DESDE QUE AMBOS MAPEIAM NO MESMO "BUCKET" 4, MAS COMO EXISTIAM 2 "SLOTS" DISPONIVEIS NESSE "BUCKET", AMBOS OS REGISTROS PUDERAM SER ACOMODADOS SEM PROBLEMAS. SUPONHAMOS AGORA QUE UM NOVO REGISTRO DE IDENTIFICADOR 74 SEJA PESQUISADO. O "BUCKET" 4 JA ESTA CHEIO E IREMOS PESQUISAR OS "BUCKETS" 5, 6 E 7, PARA FINALMENTE ARMAZENARMOS O NOVO REGISTRO NO "BUCKET" 8 (COM DESLOCAMENTO IGUAL A 4), QUANDO NOS CERTIFICARMOS DE QUE AQUELE REGISTRO NAO ESTAVA PRESENTE NO ARQUIVO. UM REGISTRO DE IDENTIFICADOR 69 SERIA ARMAZENADO NO "BUCKET" 80, QUE SUCEDE AO "HOME BUCKET" CHEIO, 89.

PODEMOS OBSERVAR O AMONTOAMENTO OU ACUMULO ("CLUSTERING" OU "PILE UP") NA FIGURA 3.1, ATRIBUIVEL A UMA TRANSFORMACAO CHAVE-ENDERECO NAO UNIFORME. COMO OBSERVADO POR KNUTH(14), ESTE PROBLEMA TENDE A SER FORMADO E AGRAVADO POR DERRAMAMENTOS ("SPILLS") CONSECUTIVOS, POIS O PRESENTE METODO TENDE A REUNIR AMONTOAMENTOS CONSECUTIVOS. UM NOVO REGISTRO DE CHAVE I A SER ARMAZENADO NO ARQUIVO DA FIGURA 3.1 TERIA QUE SER INSERIDO NUM DOS 5 "SLOTS" DESOCUPADOS, MAS ELES NAO SAO IGUALMENTE PROVAVEIS; DE FATO, O REGISTRO SERA INSERIDO EM R17 SE 4<=T(I)<=8, ENQUANTO O SERA EM R06 APENAS SE T(I)=3, DONDE A POSICAO R17 TEM UMA PROBABILIDADE 5 VEZES MAIOR DO QUE R06 DE SER OCUPADA, OU SEJA: "AS SEQUENCIAS GRANDES TENDEM A CRESCER AINDA MAIS".

ESTA CARACTERISTICA NAO E POR SI SO SUFICIENTE PARA EXPLICAR O DESEMPENHO POBRE DO METODO DE PESQUISA LINEAR, POIS UM FATO SEMELHANTE OCORRE, POR EXEMPLO, COM O METODO DA PESQUISA ENCADEADA COM LISTAS COALESCENTES (ALGORITMO E), ONDE UMA LISTA DE COMPRIMENTO 5 TAMBEM TEM UMA PROBABILIDADE 5 VEZES MAIOR DE CRESCER QUE UMA LISTA DE COMPRIMENTO 1. O PROBLEMA REAL OCORRE QUANDO UM REGISTRO COMO R17, NA FIGURA 3.1, E OCUPADO, POIS DOIS AMONTOAMENTOS SE-

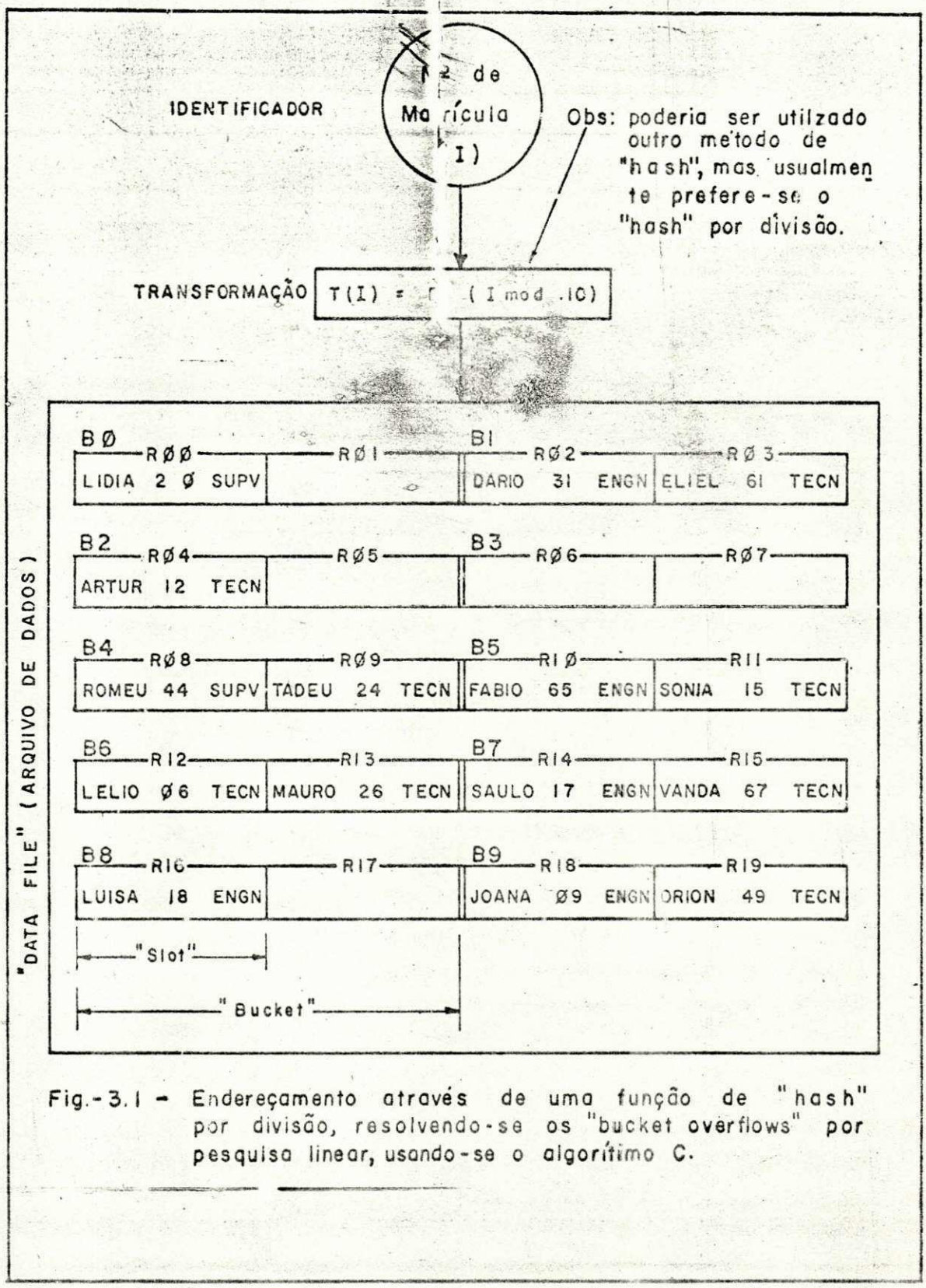


Fig.-3.1 - Endereçamento através de uma função de "hash" por divisão, resolvendo-se os "bucket overflows" por pesquisa linear, usando-se o algorítimo C.

PARADOS SAO FUNDIDOS NUM SO, CRESCENDO DE 5(9) PARA 7(13) "BUCKETS" (REGISTROS) DE UMA SO VEZ, ENQUANTO QUE AS LISTAS DO ALGORITMO "E" NUNCA AUMENTAM MAIS QUE +1, CADA VEZ. PORISSO, A PERFORMANCE DA PESQUISA LINEAR DEGRADA-SE RAPIDAMENTE A MEDIDA QUE O FATOR DE CARGA TENDE A 1.0.

NOTEMOS QUE, NA PRATICA, HA UMA CERTA TENDENCIA DE ENCONTRARMOS IDENTIFICADORES CONSECUTIVOS, I, I+1, I+2,..., O QUE TENDE A AGRAVAR O PROBLEMA DO AMONTOAMENTO ("CLUSTERING"), QUANDO O "BUCKET SIZE" E 1 (OU MESMO 2 OU 3) E USAMOS O METODO DA DIVISAO. O USO DE UM OUTRO BOM METODO DE CALCULO DO "HASH", COMO O METODO DA MULTIPLICACAO, PODERIA EVITAR (OU ATENUAR) ESTE PROBLEMA.

EXISTEM MUITAS VARIACOES EM TORNO DO METODO DA PESQUISA LINEAR, VISANDO CONTORNAR OU ATENUAR O PROBLEMA DO "CLUSTERING". UMA DAS MAIS SIMPLES E TOMARMOS IBUCKT=(IBUCKT+ISKIP)MOD. M AO INVES DE IBUCKT=IBUCKT+1, NO PASSO C.3.1 DO ALGORITMO C, PARA QUALQUER ISKIP>0 QUE SEJA PRIMO RELATIVO COM M. NA VERDADE ESTA SOLUCAO NAO RESOLVE O PROBLEMA DO AMONTOAMENTO, POIS GRUPOS DE REGISTROS DISTANCIADOS UM DOS OUTROS DE ISKIP SERAO FORMADOS, FENOMENO QUE PODERIA SER CHAMADO DE "AMONTOAMENTO SECUNDARIO", POR SER TAO REAL QUANTO O ANTERIOR, MAS NAO TAO APARENTE. A UNICA VANTAGEM E QUE, PARA "BUCKET SIZE" IGUAL A 1, A OCORRENCIA DE CHAVES EM SEQUENCIA (I, I+1, I+2,...) SERA VANTAJOSA AO INVES DE DESVANTAJOSA.

SE O ARQUIVO ESTIVER NA MEMORIA SECUNDARIA A PESQUISA LINEAR, E ESTA SUA VARIACAO ACIMA SUGERIDA, PROVAVELMENTE SERAO MELHORES AS DESCRITAS DE 3.2.2 A 3.2.6, PORQUE O INCREMENTO ISKIP PODE SER FREQUENTEMENTE ESCOLHIDO DE MODO QUE MINIMIZE O TEMPO DE LATENCIA ("LATENCY DELAY") ENTRE ACESSOS CONSECUTIVOS. NA VERDADE, PARA UM "BUCKET SIZE" RAZOAVELMENTE GRANDE (DIGAMOS, MAIOR QUE 20) E FATOR DE CARGA NAO MUITO ALTO (DIGAMOS, NAO MAIOR QUE 0.85), ESTE METODO, SENDO DOS MAIS SIMPLES DE TODOS, PODE ATE EQUIPARAR-SE (OU MESMO SOBREPUJAR) O METODO DA PESQUISA ENCADEADA, EM NUMERO DE ACESSOS A MEMORIA SECUNDARIA.

SEGUNDO KNUTH(14), A VARIACAO PROPOSTA POR KRUTAR PARECE SER UMAS DAS MAIS INTERESSANTES E EFICIENTES, E CONSISTE EM TOMARMOS ISKIP=0 NA ETAPA C.1 E TRANSFORMARMOS A ETAPA C.3.1 DO ALGORITMO C EM:

ISKIP=ISKIP+1
IBUCKT=(IBUCKT+ISKIP)MOD. M

ESTA SOLUCAO TEM ALGUMAS VANTAGENS MESMO SOBRE O METODO DE PESQUISA POR DUPLO "HASH", POIS ATENUA O PROBLEMA DO "CLUSTERING" DE MODO EQUIPARAVEL, E EVITA O CALCULO DE T1(I).

AO QUE SAIBAMOS, O METODO DE PESQUISA LINEAR FOI MENCIONADO PRIMEIRAMENTE POR PETERSON (27) E MODIFICADO POR SCHAY & SPRUTH (31), TENDO TAINITER (33) REALIZADO UMA ANALISE DESTE METODO MODIFICADO. VER SEVERANCE (43), PAGI -

NA 180.

3.2.2-PESQUISA POR DUPLO "HASH":

CONSISTE EM MODIFICARM S A ETAPA C.3.1 DO ALGORITMO C (VER 3.2.1) PARA:

IBUCKT=(IBUCKT+T1(I)) MOD.M

ONDE T1(I) E UMA 2A FUNCAO DE "HASH", POIS A 1A. DELAS, T(I), FOI ADOTADA EM C.1.

OS VALORES DA FUNCAO T1(I) DEVEM TANTO VARIAREM NO INTERVALO DE 1 A M-1 NO MAXIMO, COMO SEREM PRIMOS RELATIVOS COM M. VEJAMOS ALGUMAS SUGESTOES:

1.-TOMAR M PRIMO

T(I)=I MOD. M

T1(I)=1+ I MOD(M-2)

SERIA OTIMO QUE TANTO M COMO M-2 FOSSEM PRIMOS.

2.-TOMAR M PRIMO

T(I)=I MOD. M

T1(I)=1+⌊I/M⌋MOD. (M-2)

ONDE ⌊I/M⌋ E O MAIOR INTEIRO CONTIDO NO QUOCIENTE DA DIVISAO I/M, O QUE, SENDO UM SUBPRODUTO DO CALCULO DE T(I), TORNA ESTA SOLUCAO FACIL E EFICIENTE.

3.-TOMAR M=2**K (EM MAQUINA BINARIA). CALCULAR T(I) PELO METODO MULTIPLICATIVO. CALCULAR T1(I) DESLOCANDO-SE O PRODUTO, OBTIDO NO CALCULO DE T(I), MAIS K "BITS" PARA A ESQUERDA, TOMANDO-SE ESTE RESULTADO E FAZENDO-SE "OR" COM 00....01, DE MODO A OBTER UM VALOR IMPAR ENTRE 1 E 2**K-1, PORTANTO PRIMO COM M. ESTA SOLUCAO E DE CALCULO TAMBEM BASTANTE RAPIDO.

OBS: EM CADA UMA DAS 3 TECNICAS ACIMA SUGERIDAS, T(I) E T1(I) SAO "INDEPENDENTES", NO SENTIDO DE QUE IDENTIFICADORES DISTINTOS SO TERAO OS MESMOS VALORES, TANTO PARA UMA COMO PARA A OUTRA TRANSFORMACAO, COM UMA PROBABILIDADE O(1/M**2), AO INVES DE O(1/M). TESTES EMPIRICOS COM ESTE METODO, PARA T(I) E T1(I) INDEPENDENTES, MOSTRARAM QUE PRATICAMENTE NAO EXISTE AMONTOAMENTO (NEM SEQUER AMONTOAMENTO SECUNDARIO), O QUE E UMA BOA VANTAGEM SOBRE O METODO DE RESOLUCAO DE SINONIMOS POR PESQUISA LINEAR.

4.-TOMAR M PRIMO

$$\begin{cases} T(I) \text{ QUALQUER} \\ T1(I) = \begin{cases} 1 & \text{SE } T(I)=0 \\ M-T(I) & \text{SE } T(I)>0 \end{cases} \end{cases}$$

ESTA E UMA DAS SOLUCOES EM QUE T1(I) DEPENDE DE T(I). FOI SUGERIDA POR GARY KNOTT, EM 1968, E BASTANTE RAPIDA, MAS PRODUZ ALGUM AMONTOAMENTO SECUNDARIO.

DE UM MODO GERAL, OS METODOS DE PESQUISA POR DUPLO "HASH" PARECEM SER A MELHOR VARIACAO EM TORNO DO ALGORITMO C. A IDEIA BASICA ORIGINAL DEVE-SE A GUY DE BALBINE (TESE DE PH.D. NO "CALIFORNIA INSTITUTE OF TECHNOLOGY", 1968, PAGINAS 149-150). VER MAIORES DETALHES EM KNUTH(14), PAGINAS 521 A 524, QUE AFIRMA QUE TALVEZ A UNICA VARIACAO EM TORNO DO ALGORITMO C QUE TALVEZ SUPERE A PESQUISA COM DUPLO "HASH" SEJA A SUGERIDA POR R. KRUTAR (EXERC. 20, PAGINA 544 DE KNUTH), E QUE CONSISTE EM EVITARMOS O CALCULO DE T1(I) ACRESCENTANDO NA ETAPA C.1 A INSTRUCAO ISKIP=0 E MODIFICARMOS A ETAPA C.3 1 PARA:

ISKIP=ISKIP+1;
IBUCKT=(IBUCKT+ISKIP)MOD. M.

3.2.3-PESQUISA POR DUPLO "HASH"; VARIACAO DE BRENT:

MELHORA O TEMPO MEDIO DAS PESQUISAS BEM SUCEDIDAS, PARA FATORES DE CARGA ALTOS. BRENT BASEOU-SE NO FATO DE QUE, EM MUITAS APLICACOES, PESQUISAS BEM SUCEDIDAS SAO MUITO MAIS FREQUENTES QUE INSERCOES. PORISSO ELE PROPOS TERMOS UM POUCO MAIS DE TRABALHO AO INSERIRMOS UM REGISTRO (REARRUMANDO MESMO OS REGISTROS JA INSERIDOS, SE CONVENIENTE), PARA QUE TENHAMOS UMA REDUCAO NO TEMPO MEDIO ESPERADO DE RECUPERACAO, OPERACAO BEM MAIS FREQUENTE.

VEJAMOS COMO FUNCIONA O METODO DE BRENT, PARA A INSERCAO. SUPONHAMOS QUE UMA PESQUISA MAL SUCEDIDA, POR DUPLO "HASH", EXAMINOU OS "BUCKETS" CUJOS NUMEROS DE ORDEM ESTAO ARMAZENADOS NO VETOR LOCAL(0), LOCAL(1),..., LOCAL(NOVERF-1), LOCAL(NOVERF), ONDE LOCAL(J)=(T(I)+J*T1(I))MOD. M, E O "BUCKET" B(LOCAL(NOVERF)) TEM UM "SLOT" DESOCUPADO. SE NOVERF<=1, INSERIMOS I NO "BUCKET" B(LOCAL(NOVERF)) COMO USUAL; MAS SE NOVERF>=2, VERIFICAMOS PRIMEIRO, PARA CADA CHAVE K PRESENTE NO "BUCKET" B(LOCAL(0)), SE O "BUCKET" B((LOCAL(0)+T1(CHAVE DE B(LOCAL(0))))MOD. M) TEM ALGUM "SLOT" VAZIO. SE SIM, TRANSFERIMOS O REGISTRO DE CHAVE K PARA ESSE NOVO "SLOT" E ARMAZENAMOS O REGISTRO QUE PESQUISAVAMOS, DE CHAVE I, NESTA "BRECHA" QUE ABRIMOS NO "BUCKET" B(LOCAL(0)), DE MODO QUE AUMENTAMOS DE +1 O NUMERO DE "OVERFLOWS" PARA RECUPERACAO DE K, MAS DIMINUIMOS DE NOVERF >=2 O NUMERO DE "OVERFLOWS" PARA RECUPERACAO DE I, RESULTANDO GLOBALMENTE NUM MELHORAMENTO. SIMILARMENTE, SE OS "BUCKETS" B((LOCAL(0)+T1(CHAVES DE B(LOCAL(0))))MOD. M) ESTAO TOTALMENTE OCUPADOS PARA CADA K DO "BUCKET" B(LOCAL(0)) E NOVERF>=3, IREMOS VERIFICAR OS "BUCKETS" B((LOCAL(0) + 2*T1(CHAVES DE B(LOCAL(0))))MOD. M); SE ESTES TAMBEM ESTIVEREM CHEIOS, TENTAREMOS COM OS "BUCKETS" B((LOCAL(1)+T1 (CHAVES DE B(LOCAL(1))))MOD. M); ETC. EM GERAL, SE ENCONTRARMOS OS "BUCKETS" B(LOCAL(J)+L*T1(CHAVES DE B(LOCAL(J))

))MOD. M) OCUPADOS PARA TODOS INDICES J E L TAIS QUE J+L<N E SE NOVERF>=N+1, EXAMINAMOS OS "BUCKETS"

B((LOCAL(0)+NOVERF*T1(CHAVES DE B(LOCAL(0))))MOD. M),
B((LOCAL(1)+(NOVERF-1)*T1(CHAVES DE B(LOCAL(1))))MOD. M),

.....

B((LOCAL(NOVERF-1)+T1(CHAVES DE B(LOCAL(NOVERF-1))))
MOD. M);

SE DESCOBRIRMOS O PRIMEIRO "SLOT" VAZIO NO "BUCKET" B((LOCAL(J)+(NOVERF-J)*T1(ALGUMA CHAVE DE B(LOCAL(J))))) MOD. M), ENTAO TRANSFERIMOS O REGISTRO DO "BUCKET" B(LOCAL(J)), CUJA CHAVE POSSIBILITOU ISTO, PARA AQUELE "SLOT" VAZIO ENCONTRADO, E DEPOIS INSERIMOS O REGISTRO DE IDENTIFICADOR I, QUE ESTAVAMOS PESQUISANDO, NA "BRECHA" QUE ABRIMOS.

VER MAIORES DETALHES EM BRENT(5) E KNUTH(14).

3.2.4-PESQUISA QUADRATICA E SUAS EXTENSOES:

ESTE METODO CONSISTE EM MODIFICARMOS A ETAPA C.3.1 DO ALGORITMO C (VER 3.2.1) PARA

```
IBUCKT=(IHASH+J*K+J**2*L)MOD. M
J=J+1
```

APOS ACRESCENTARMOS A INSTRUCAO J=0 NA ETAPA C.1, SENDO 'K' E 'L' NUMEROS INTEIROS (POSITIVOS) PREVIA E ADEQUADAMENTE ESCOLHIDOS.

A FORMULA COMECARA A GERAR, EM UM CERTO PONTO, UMA SEQUENCIA CICLICA DE NUMEROS, COMO EXEMPLIFICADO ABAIXO, PARA K=4, L=8, M=32 (NUMERO DE "BUCKETS") E T(I)=5 ("HOME BUCKET"):

J	IBUCKT(J)= T(I)+J*K+J**2*L	IBUCKT(J) MOD. M
0	5	5
1	17	17
2	45	13
3	89	25
4	149	21
5	225	1
6	317	29
7	425	9
8	549	5
9	689	17
10	845	13
11	1017	25
12	1205	21
13	1409	1
14	1629	29
15	1865	9

A PESQUISA TERMINA UMA VEZ QUE O CICLO COMECA. MAURER (30) OBSERVOU QUE O NUMERO DE ELEMENTOS DO CICLO E PEQUENO DEMAIS PARA SER EFICIENTE QUANDO O NUMERO DE "BUCKETS", M, E UMA POTENCIA DE 2, MAS QUE A PESQUISA COBRIRIA EXATAMENTE METADE DO ARQUIVO SE ESCOLHESSEMOS UM VALOR DE M PRIMO QUALQUER. PARA NUMEROS PRIMOS DA FORMA 4*K+3, RADKE (29) E DAY (07) MOSTRARAM QUE TODO O ARQUIVO SERIA COBERTO EM M PASSOS. ACKERMAN (01) MOSTROU QUE, SE M=P**N, SENDO P PRIMO E N>1, A PESQUISA TAMBEM COBRIRIA TODOS OS REGISTROS DO ARQUIVO EM M PASSOS. BATAGELJ (49) OBTEVE ESTE MESMO RESULTADO PARA M=PRODUTORIO DE P(I)**N, ONDE OS NUMEROS P(I) SAO PRIMOS E N>1.

A IDEIA ORIGINAL DESTE METODO DEVE-SE A MAURER (22). NOTEMOS QUE, BASICAMENTE, A INTENCAO E A DE CONTORNAR O PROBLEMA DO AMONTOAMENTO ("CLUSTERING") DO METODO DA PESQUISA LINEAR, O QUE E CONSEGUIDO, MUITO EMBORA, AO CONTRARIO DO QUE VIMOS EM 3.2.3, NADA SEJA FEITO A RESPEITO DO PROBLEMA DE UM GRANDE NUMERO DE SINONIMOS PARA UM CERTO REGISTRO.

EXISTEM NUMEROSAS EXTENSOES DO METODO ORIGINAL, DEVIDAS A BELL (04 E 05), RADKE (40), DAY (08), LUCCIO (23), BRENT (06) E ACKERMAN (01), ALGUMAS DELAS VISANDO TAMBEM SOLUCIONAREM O PROBLEMA DOS MUITOS SINONIMOS.

RADKE (40) MOSTROU QUE NAO HA VANTAGEM ALGUMA EM TOMARMOS UMA FORMULA QUADRATICA GENERICA IBUCKT(J)=(T(I)+J*K+J**2*L) MOD. M EM LUGAR DE, SIMPLESMENTE,

IBUCKT(J)=(T(I)+J**2) MOD. M.

DAY (08) E BELL (04) APRESENTARAM ALGORITMOS COMPUTACIONALMENTE BEM SIMPLES E QUE COBREM O ARQUIVO COMPLETAMENTE.

3.2.5-PESQUISA RANDOMICA:

ANTES DE PROSSEGUIRMOS, RELEIAMOS 2.3.4.

COMO VIMOS, O METODO RANDOMICO PARA OBTENCAO DE UM "HOME BUCKET" TAMBEM PODE SER USADO PARA FORNECER UM ENDERECO "SUBSEQUENTE", A SER USADO NO CASO DE OCORRENCIA DE "BUCKET OVERFLOW".

EM OUTRAS PALAVRAS, ESTE METODO CONSISTE EM MODIFICARMOS A ETAPA C.3.1 DO ALGORITMO "C" PARA

IBUCKT=T(IBUCKT)

ONDE ESSA TRANSFORMACAO T E REALIZADA, NA VERDADE, PELA CHAMADA DE UMA SUBROTINA DE GERACAO DE NUMEROS PSEUDO-ALEATORIOS, AJUSTANDO-SE O ARGUMENTO DE RETORNO AO INTERVALO DE 0 A M-1.

3.2.6-PESQUISA DE VYSSOTSKY:

CONSISTE EM USARMOS UMA SERIE DE TRANSFORMACOES T1(I), T2(I), T3(I),...TN(I), E NAO APENAS UMA. SE NAO CONSEGUIRMOS ENCONTRAR UM "SLOT" DESOCUPADO NO "BUCKET" CORRESPONDENTE A T1(I), ENTAO TENTAMOS COM A PROXIMA TRANSFORMACAO DA SEQUENCIA, T2(I), E ASSIM POR DIANTE. SOMENTE APOS EXAURIMOS TODAS AS N TRANDFORMACOES SEM SUCESSO E QUE UM OUTRO METODO CONVENCIONAL (COMO O DA PESQUISA LINEAR, POR EXEMPLO) SERA TENTADO.

AO QUE SAIBAMOS, QUEM PRIMEIRO PUBLICOU ESTUDOS SOBRE ESTE METODO, COMO APRESENTAMOS, FOI MCILROY (32), MAS QUE O ATRIBUIU A VYSSOTSKY.

3.2.7-PESQUISA POR ENCADEAMENTO (LISTAS DE SINONIMOS)

ASSOCIAMOS UM APONTADOR DO "SUCESSOR" A CADA "BUCKET" QUE TRANSBORDOU, INDICANDO EXPLICITAMENTE QUAL O "PROXIMO" "BUCKET" A SER EXAMINADO. SEMPRE QUE O ULTIMO "BUCKET" DUMA LISTA (OU CADEIA) DE SINONIMOS TRANSBORDA DURANTE UMA INSERCAO, UM OUTRO "BUCKET" VAZIO E LOCALIZADO (POR UM ALGORITMO CONVENIENTE) E LIGADO A LISTA. ESSAS LISTAS DE "BUCKETS" SAO SEGUIDAS TANTO PARA ARMAZENAMENTO COMO PARA RECUPERACAO DE REGISTROS.

APARENTEMENTE, A IDEIA DA RESOLUCAO DE "BUCKET OVERFLOWS" POR ENCADEAMENTO RESULTOU DOS TRABALHOS DE PROCESSAMENTO DE LISTAS USADOS NOS ESTUDOS SOBRE INTELIGENCIA ARTIFICIAL NA DECADA DE 1950.

3.2.7.1-ENCADEAMENTO USANDO LISTAS SEPARADAS E CABECAS DE LISTA:

TALVEZ SEJA ESTA A MANEIRA MAIS OBVIA DE RESOLVER "BUCKET OVERFLOWS": MANTER M LISTAS ENCADEADAS, UMA PARA CADA VALOR POSSIVEL DA TRANSFORMACAO "HASH". UM CAMPO DE APONTADOR ("LINK") PARA O SUCESSOR E INCLUIDO EM CADA "BUCKET" E EXISTEM M CABECAS ("HEADS") DE LISTA, NUMERADAS, DIGAMOS, DE 0 A M-1. APOS OBTERMOS 0<=T(I)<=M-1, SIMPLESMENTE FAZEMOS UMA PESQUISA SEQUENCIAL NA LISTA COM "CABECA" DE NUMERO DE ORDEM T(I). SE CHEGARMOS AO FINAL DA LISTA SEM SUCESSO, INSERIMOS O REGISTRO DE CHAVE I OU NO ULTIMO "SLOT" LIVRE ENCONTRADO NA LISTA (SE ESTE EXISTIR), OU NO "SLOT" 1 DE UM NOVO "BUCKET" RETIRADO DO ESPACO DISPONIVEL E LIGADO AO FINAL DA LISTA. (SE NAO HAVIA "SLOT" LIVRE NA LISTA).

-ALGORITMO D:

PESQUISA E INSERCAO DE UM REGISTRO DE IDENTIFICADOR I, RESOLVENDO OS "BUCKET OVERFLOWS" POR PESQUISA ENCADEADA, COM LISTAS SEPARADAS E CABECAS DE LISTA. PODERIAMOS TER FEITO O ALGORITMO DE MODO QUE NAO FOSSEM PERMITIDOS VAZIOS INTERMEDIARIOS NA CADEIA, COM O QUE A INSERCAO SERIA FEITA NO PRIMEIRO "SLOT" VAZIO ENCONTRADO NA CADEIA. MAS ISSO PODERIA TER O INCONVENIENTE DE, NA DELECAO, TERMOS QUE EFETUAR DESLOCAMENTOS DE REGISTROS PARA EVITAR VAZIOS INTERMEDIARIOS NA CADEIA. PORISSO ADOTAMOS UM ALGORITMO QUE PERMITE MAIS QUE 1 "SLOT" INTERMEDIARIO VAZIO NA CADEIA (SO NAO PERMITINDO UM "BUCKET" TOTALMENTE VAZIO), FAZENDO A INSERCAO NO ULTIMO "SLOT" VAZIO ENCONTRADO NA CADEIA.

ARGUMENTOS DE ENTRADA E DE RETORNO:
OS MESMOS DO ALGORITMO C, MAS M E O NUMERO DE CABECAS DE LISTAS (10 NA FIGURA 3.2), USUALMENTE PRIMO.

OUTROS SIMBOLOS:
OS MESMOS DO ALGORITMO C, E MAIS:

IHEAD E UM VETOR QUE SERVE DE CABECA PARA AS M LISTAS. VARIA DE IHEAD (0) A IHEAD (M-1).

IDESOC E UM APONTADOR PARA O ULTIMO REGISTRO DESOCUPADO QUE ENCONTRAMOS NA PESQUISA.

LINK E O CAMPO DE APONTADOR DO SUCESSOR DE CADA "BUCKET".

```
CONDICAO INICIAL: AO INICIALIZARMOS O ARQUIVO, POR ALGUMA
CONVENCAO (POR EXEMPLO, -1) MARCAMOS TODOS OS REGISTROS
COMO DESOCUPADOS, E TODOS OS VALORES DO VETOR IHEAD E DOS
CAMPOS LINK COM A CONVENCAO DE FIM DE LISTA (POR EXEMPLO ,
-1). UM OUTRO ALGORITMO SEPARADO TRATARA DA ADMINISTRACAO
DO ESPACO DISPONIVEL.

D.1-(FAZ O "HASH" E TESTA IHEAD (IHASH))
    IHASH=T(I)
    IDESOC=-1
    SE IHEAD(IHASH)=-1, ASSINALANDO LISTA VAZIA, OBTEVE-SE
        INSUCESSO. FACA FOUND=".FALSE.", APANHE UM NOVO "
        BUCKET", NBUCKT, DO ESPACO DISPONIVEL, E VA PARA
        D.3.3.2.
    SE NAO, FACA IBUCKT=IHEAD(IHASH).
D.2-(PROSSEGUE ATRAVES DO "BUCKET". ESTE "BUCKET" CONTEM O
    REGISTRO PESQUISADO ?)
    INICBK=IBUCKT*NRGBKT
    IREGI=INICBK
    D.2.1-(O REGISTRO R(IREGI) ESTA DESOCUPADO ?)
        SE SIM, FACA IDESOC=IREGI E VA PARA D.2.3.
        SE NAO:
    D.2.2-(O REGISTRO R(IREGI) TEM A CHAVE IGUAL A I ?)
        SE SIM, OBTEVE-SE SUCESSO. FACA FOUND=".TRUE.".
            FIM.
        SE NAO:
    D.2.3-(EXPERIMENTA NO PROXIMO REGISTRO DO "BUCKET")
        FACA IREGI=IREGI+1
        SE IREGI<=INICBK+(NRGBKT-1) VA PARA D.2.1.
        SE NAO:
D.3-(O "BUCKET" NAO CONTEM O REGISTRO PESQUISADO. SEGUE  A
    LISTA DE SINONIMOS.)
    D.3.1-(CHEGOU AO FINAL DO "BUCKET". CALCULA O "PROXIMO
        " "BUCKET")
        SE LINK(IBUCKT)=-1, APONTANDO PARA O VAZIO, A
            LISTA DE SINONIMOS TERMINOU;
            SE IDESOC=-1 VA PARA D.3.3.
            SE NAO, FACA IREGI=IDESOC E VA PARA D.3.2
        SE NAO, FACA IBUCKT=LINK(IBUCKT) E VA PARA D.2.
    D.3.2-(A LISTA DE SINONIMOS TERMINOU SEM SUCESSO, MAS
        HA UM ULTIMO "SLOT" DESOCUPADO E O REGISTRO E AI
        INSERIDO).
        OBTEVE-SE INSUCESSO. FACA FOUND=".FALSE." E
        IREGI=IDESOC. MARQUE O REGISTRO R(IREGI) COMO O-
        CUPADO E ARMAZENE O REGISTRO QUE ESTAVA PESQUI -
        SANDO. FIM.
    D.3.3-(A LISTA DE SINONIMOS TERMINOU, SEM SUCESSO, E
        SEM NENHUM "SLOT" DESOCUPADO. APANHA UM "BUCKET"
        DO ESPACO DISPONIVEL E INSERE NO SEU "SLOT" 1  )
        OBTEVE-SE INSUCESSO. FACA FOUND=".FALSE.". APA -
```

NHE UM NOVO "BUCKET", NBUCKT, TOTALMENTE DESOCUPADO, DO ESPACO DISPONIVEL.

D.3.3.1-(INCORPORA NBUCKT A CADEIA DE SINONIMOS)
ADOTE, CONSTANTEMENTE, UMA UNICA DAS DUAS ALTERNATIVAS ABAIXO:
A) FACA LINK(IBUCKT)=NBUCKT E VA PARA D.3.3.2.
B) FACA LINK(NBUCKT)=IHEAD(IHASH) E IHEAD(IHASH)=NBUCKT

D.3.3.2-(INSERE NO PRIMEIRO "SLOT" DE NBUCKT) FACA IREGI=NBUCKT*NRGBKT, MARQUE O REGISTRO R(IREGI) COMO OCUPADO E ARMAZENE O REGISTRO QUE ESTAVA PESQUISANDO. FIM .

COMO EXEMPLO, CONSIDEREMOS OS MESMOS DADOS USADOS PARA FORMARMOS A FIGURA 3.1, EM 3.2.1, POREM AGORA RESOLVENDO OS "OVERFLOWS" COM PESQUISA ENCADEADA, ATRAVES DE LISTAS SEPARADAS E USANDO UM VETOR IHEAD PARA AS CABECAS DE LISTA. TEREMOS A FIGURA 3.2.

SE PESQUISARMOS UM REGISTRO DE IDENTIFICADOR 73, COMO T(73)=3 EXAMINAREMOS A POSICAO IHEAD(3); COMO ELA ESTA VAZIA, SIGNIFICA QUE NAO HA AINDA SINONIMO DESTA CHAVE, PORTANTO LOCALIZAREMOS UM "BUCKET" DISPONIVEL SEGUNDO ALGUM CRITERIO, DIGAMOS B09 (FAZENDO DEPOIS AVAIL=LINK (AVAIL)=B10), INSERIMOS O REGISTRO NO PRIMEIRO "SLOT" LIVRE DE B09 E FAZEMOS IHEAD (3)=B09. SE AGORA PESQUISARMOS UM REGISTRO DE IDENTIFICADOR 22, COMO T (22)=2 E IHEAD (2)=B00, E O SEGUNDO "SLOT" DE B00 ESTA LIVRE, BASTA ARMAZENARMOS AI O REGISTRO PESQUISADO. SE AGORA, FINALMENTE, PESQUISARMOS UM REGISTRO DE IDENTIFICADOR 74, COMO T (74)=4, EXAMINAREMOS A POSICAO IHEAD (4), QUE APONTA PARA O "BUCKET" B07, QUE NAO CONTEM O REGISTRO PESQUISADO, PORISSO TENTAMOS SEGUIR A LISTA DE SINONIMOS, O QUE E IMPOSSIVEL, POIS ELA TERMINA EM B07 MESMO; PORISSO LOCALIZAMOS UM "BUCKET" DISPONIVEL SEGUNDO ALGUM CRITERIO, DIGAMOS B10 (FAZENDO DEPOIS AVAIL=LINK (AVAIL)=B11), INSERIMOS O REGISTRO NO PRIMEIRO "SLOT" LIVRE DE B10 E FAZEMOS:

A)- OU APENAS LINK (B07)=B10

B)- OU APENAS LINK (B10)=B07 E IHEAD (4)=B10.

NOTEMOS QUE O DESLOCAMENTO DA CHAVE 74 SERA AGORA APENAS DE 1, ENQUANTO QUE, COM O METODO DE PESQUISA LINEAR, FOI DE 4.

AS VEZES MANTEMOS OS REGISTROS DAS LISTAS ORDENADOS SEGUNDO SUAS CHAVES, SE ELAS SAO MAIS OU MENOS EQUIPROVAVEIS, TORNANDO MAIS RAPIDAS PESQUISAS MAL SUCEDIDAS E INSERCOES. SE A ORDEM FOR CRESCENTE (DECRESCENTE), TODOS OS APONTADORES -1 DOS "BUCKETS" DEVEM PASSAR A APONTAR PARA UM "BUCKET" ENGODO ("DUMMY"), COM UMA CHAVE IMPOSSIVEL DE ACONTECER, SIMBOLICAMENTE $+\infty$ ($-\infty$).

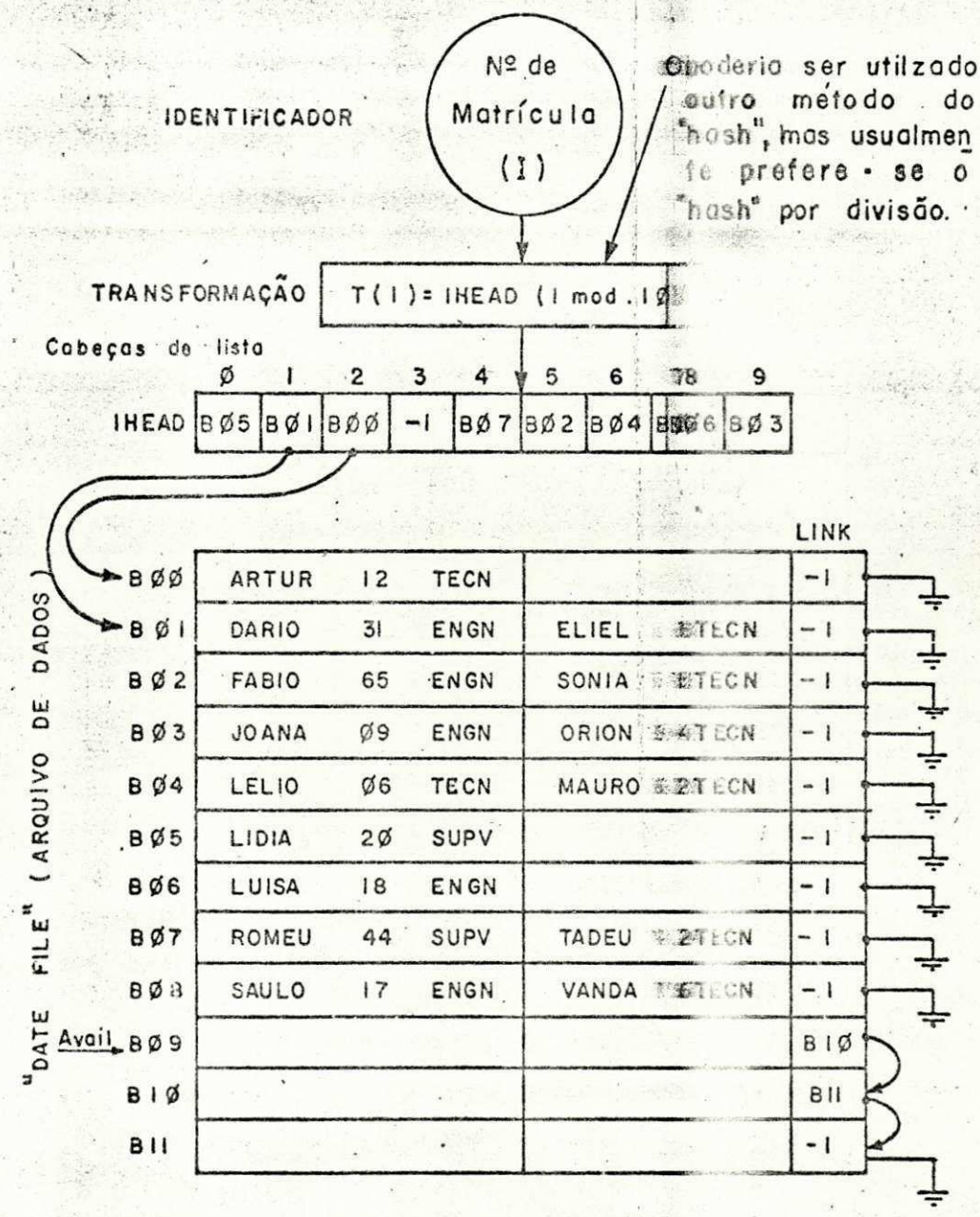


Fig.- 3.2 – Endereçamento através de uma função de "hash" por divisão resolvendo-se os "overflows" por pesquisa encadeada, com listas ligadas e cabeças de lista, usando-se o algoritmo D.

UMA OUTRA BOA IDEIA PARA MELHORAMOS A EFICIENCIA E CONSERVARMOS OS REGISTROS EM ORDEM DECRESCENTE DA SUA FREQUENCIA DE "USO" (OU PROCURA). ESTE CONCEITO, CONHECIDO COMO ARQUIVO-AUTO-ORGANIZAVEL ("SELF-ORGANIZABLE FILE") PODE SER MELHOR ESTUDADO EM KNUTH (20), PAGINAS 398-399 , ENTRE OUTROS.

ESSAS DUAS ULTIMAS SUGESTOES APLICAM-SE MELHOR NO CASO ESPECIAL DO "BUCKET SIZE" SER IGUAL A 1 (COMUMENTE EM MEMORIA PRINCIPAL OU EM MEMORIA VIRTUAL PAGINADA).

DE UM MODO GERAL, SE OS REGISTROS SAO GRANDES, DEVENDO FICAREM NA MEMORIA SECUNDARIA, E PRINCIPALMENTE SE O VETOR IHEAD, RELATIVAMENTE BEM MENOR, PODE FICAR PERMANENTEMENTE NA MEMORIA PRINCIPAL, O ALGORITMO D E O MAIS EFICIENTE DE TODOS, EM NUMERO DE ACESSOS A MEMORIA SECUNDARIA, POIS A PESQUISA SEQUENCIAL DE "BUCKETS", NUMA PESQUISA MAL SUCEDIDA, FICA REDUZIDA A LISTAS BEM CURTAS, DE COMPRIMENTO MEDIO =N/M= NUMERO DE "BUCKETS" UTILIZADOS/NUMERO DE CABECAS DE LISTA. INVERSAMENTE, SE OS REGISTROS FOREM BEM PEQUENOS, O CONSUMO RELATIVO DE MEMORIA COM O VETOR IHEAD PODERA SER (RELATIVAMENTE) JA PONDERAVEL. POR OUTRO LADO , SEMPRE PODERIAMOS DESEJAR MELHORAR A EFICIENCIA EM TEMPO AUMENTANDO M (DIMENSAO DE IHEAD), MAS ISSO TEM UM LIMITE (DIGAMOS, M GERALMENTE FICA NO INTERVALO DE 1 A 2 VEZES O NUMERO DE REGISTROS ESPERAVEIS/"BUCKET SIZE", PORQUE SENAO DESPERDICARIAMOS MUITO ESPACO COM O VETOR IHEAD, E PRINCIPALMENTE PORQUE TERIAMOS EM MEDIA MUITOS "SLOTS" VAZIOS POR "BUCKET".

ESSAS CONSIDERACOES LEVAM A UM OUTRO METODO DE PESQUISA ENCADEADA, SEM USO DO VETOR IHEAD, QUE VEREMOS EM 3.2.7.2.

CHAMAMOS A ATENCAO DE QUE O METODO QUE VIMOS AQUI EM 3.2.7.1 TAMBEM E CONHECIDO COMO "SCATTER INDEX TABLE", E SOB ESSA DESIGNACAO SEU ESTUDO SERA CONTINUADO E APROFUNDADO O MAXIMO QUE PUDERMOS, NO CAPITULO 4, PRIMEIRAMENTE CONSIDERANDO O "BUCKET SIZE" IGUAL A 1, E DEPOIS CONSIDERANDO O ARQUIVO EM MEMORIA VIRTUAL PAGINADA.

CHAMAMOS A ATENCAO, TAMBEM, DE QUE ESTE METODO, NA REALIDADE, TEM AS AREAS DE "OVERFLOW" (E AINDA MAIS DE CADA LISTA) EM SEPARADO.

AO QUE SAIBAMOS, JOHNSON (18) FOI QUEM PRIMEIRO TRATOU DO METODO DE PESQUISA ENCADEADA PARA UM "BUCKET SIZE" MAIOR QUE 1.

3.2.7.2-ENCADEAMENTO USANDO LISTAS COALESCENTES.

NESTE METODO NAO USAMOS NENHUM VETOR IHEAD COMO CABECA DE LISTAS DE SINONIMOS, TAL COMO FIZEMOS NO METODO ANTERIOR. AO INVES DE TERMOS AS LISTAS DE SINONIMOS SEPARADAS, PERMITIMOS QUE ELAS COALESCAM, ISTO E, AGLUTINEM-SE OU UNAM-SE NUMA SO, PODENDO ATE TERMOS REGISTROS NAO SINONIMOS DENTRO DE UM MESMO "BUCKET". COM ISSO USAMOS SENSI -

VELMENTE MENOS ESPACO, EMBORA GASTEMOS UM POUQUINHO A MAIS DE TEMPO NUMA RECUPERACAO OU INSERCAO.

- ALGORITMO "E":

PESQUISA E INSERCAO DE UM REGISTRO DE IDENTIFICADOR I , RESOLVENDO OS "BUCKET OVERFLOWS" POR PESQUISA ENCADEADA , COM AS LISTAS (DE SINONIMOS) COALESCENTES.

ARGUMENTOS DE ENTRADA E RETORNO: OS MESMOS DO ALGORITMO C.
OUTROS SIMBOLOS: OS MESMOS DO ALGORITMO C, E MAIS:
LINK (IBUCKT) E UM CAMPO DE APONTADOR PARA O SUCESSOR , EXISTENTE EM CADA "BUCKET";
NIVEL INDICA A PARTIR DE ONDE NENHUM "BUCKET" ESTA TO - TALMENTE VAZIO, ISTO E, BUCKET (J) NAO ESTA TO - TALMENTE VAZIO PARA NIVEL<=J<=M;
IDESOC E UM APONTADOR PARA O ULTIMO REGISTRO DESOCUPADO QUE ENCONTRAMOS NA PESQUISA.
CONDICOES INICIAIS: AO INICIALIZARMOS O ARQUIVO:
A) POR ALGUMA CONVENCAO (POR EXEMPLO, -1) MARCAMOS TO - DOS OS REGISTROS DESDE O "BUCKET" Ø ATE O "BUCKET" M-1 COMO DESOCUPADOS;
B) MARCAMOS TODOS OS REGISTROS DO "BUCKET" M (QUE E UM "BUCKET" DE ENGODO, UM "DUMMY") COMO OCUPADOS;
C) MARCAMOS TODOS OS "BUCKETS" COMO FIM DE LISTA, FA - ZENDO, POR EXEMPLO, LINK (IBUCKT)=-1, 0<=IBUCKT<=M ;
D) FAZEMOS NIVEL=M.

E.1-(FAZ O "HASH")
IHASH=T(I)
IBUCKT=IHASH
IDESOC=-1
E.2-(PROSSEGUE ATRAVES DO "BUCKET". ESTE "BUCKET" CONTEM O REGISTRO PESQUISADO ?)
INICBK=IBUCKT*NRGBKT
IREGI=INICBK
E.2.1-(O REGISTRO R (IREGI) ESTA DESOCUPADO ?)
SE SIM, FACA IDESOC=IREGI E VA PARA E.2.3.
SE NAO:
E.2.2-(O REGISTRO R (IREGI) TEM A CHAVE IGUAL A I ?)
SE SIM, OBTEVE-SE SUCESSO. FACA FOUND=".TRUE." . FIM.
SE NAO:
E.2.3-(EXPERIMENTA NO PROXIMO REGISTRO DO "BUCKET")
FACA IREGI=IREGI+1
SE IREGI<=INICBK+(NRGBKT-1) VA PARA E.2.1.
SE NAO:
E.3-(O "BUCKET" NAO CONTEM O REGISTRO PESQUISADO SEGUE A

```
LISTA DE SINONIMOS.)
E.3.1-(CHEGOU AO FINAL DO "BUCKET". CALCULA SEU SUCES-
      SOR)
      SE LINK (IBUCKT)=-1:
         SE IDESOC<>-1 FACA IREGI=IDESOC E VA PARA E.3.
                       2.
         SE NAO:
            E.3.1.1-(A LISTA DE SINONIMOS TERMINOU SEM
                     SUCESSO E SEM TER "SLOT" DESOCUPA-
                     DO. PROCURA ONDE INSERIR).
                     SE HA ALGUM REGISTRO, IREGI, EM B(
                        NIVEL), DESOCUPADO, FACA LINK (
                        IBUCKT)=NIVEL E VA PARA E.3.2 .
                     SE NAO, FACA NIVEL=NIVEL-1
                        SE NIVEL<0, OBTEVE-SE INSUCES -
                           SO. FACA FOUND=".FALSE."   E
                           ASSINALE "ARQUIVO CHEIO".
                           FIM.
                        SE NAO, VA PARA E.3.1.1.
      SE NAO:
            E.3.1.2-(A LISTA DE SINONIMOS AINDA NAO
                     TERMINOU. SEGUE PARA O  "PROXIMO "
                     "BUCKET")
                     FACA IBUCKT=LINK (IBUCKT) E VA PA-
                     RA E.2.
E.3.2-(A LISTA DE SINONIMOS TERMINOU SEM SUCESSO, MAS
      HA UM ULTIMO "SLOT" DESOCUPADO E O REGISTRO E
      INSERIDO AI).
      OBTEVE-SE INSUCESSO. FACA FOUND=".FALSE.", MAR -
      QUE R (IREGI) COMO OCUPADO E ARMAZENE AI O RE  -
      GISTRO QUE ESTAVA PESQUISANDO. FIM.
```

COMO EXEMPLO, CONSIDEREMOS OS MESMOS DADOS USADOS PARA FORMARMOS A FIGURA 3.1, EM 3.2.1, POREM AGORA RESOLVENDO OS "OVERFLOWS" POR PESQUISA ENCADEADA, COM AS LISTAS COA - LESCENTES. TEREMOS A FIGURA 3.3.

SE PESQUISARMOS UM REGISTRO DE IDENTIFICADOR 73, COMO T(73)=03 EXAMINAREMOS O "BUCKET" B03, COMO ELE NAO CONTEM O IDENTIFICADOR, TEM VAZIOS E E FIM DE LISTA, INSERIMOS O REGISTRO PESQUISADO NO SEU ULTIMO "SLOT" LIVRE, R (07). SE AGORA PESQUISARMOS UM REGISTRO DE IDENTIFICADOR 22, COMO T (22)=02 EXAMINAREMOS O "BUCKET" B02, COMO ELE NAO CONTEM O IDENTIFICADOR, TEM VAZIOS E E FIM DE LISTA, INSERIMOS O REGISTRO PESQUISADO NO ULTIMO "SLOT" VAZIO ENCONTRADO, R(04). SE AGORA, FINALMENTE, PESQUISARMOS UM REGISTRO DE I- DENTIFICADOR 74, COMO T (74)=04, EXAMINAREMOS O "BUCKET" B04, QUE NAO CONTEM O IDENTIFICADOR, NAO TEM "SLOT" VAZIO E E FIM DE LISTA, PORISSO VERIFICAMOS SE ALGUM REGISTRO NO "BUCKET" B (NIVEL) ESTA DESOCUPADO. COMO NAO ESTA, FAZEMOS

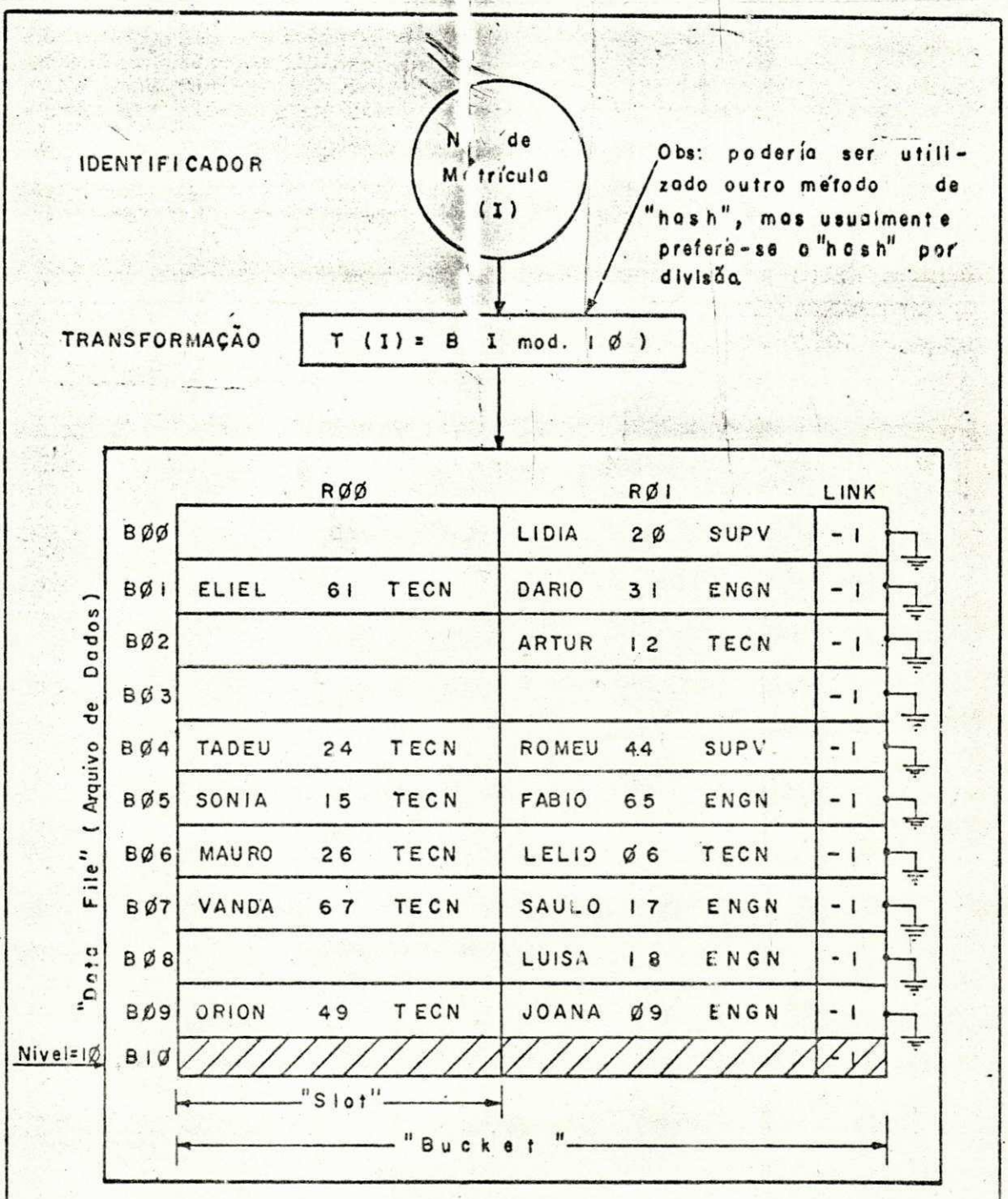


Fig.-3.3 - Endereçamento através de uma função de "hash" por divisão, resolvendo-se os "bucket overflows" por pesquisa encadeada, com listas coalescentes, usando-se o algorítimo E.

NIVEL=NIVEL-1 ATE QUE OBTENHAMOS UM "BUCKET", B08, QUE TENHA UM "SLOT" LIVRE, R (16), QUANDO FICAREMOS COM NIVEL=8, INSERIMOS O REGISTRO PESQUISADO EM R (16) E FAZEMOS LINK(B (04))=B08.

NOTEMOS QUE AS CHAVES QUE FICARAO NO "BUCKET" B (08), ISTO E, 74 E 18, NAO SAO SINONIMAS, MAS AO INVES DE TERMOS DUAS LISTAS SEPARADAS DE SINONIMOS, ELAS COALESCEM (UNEM-SE, AGLUTINAM-SE) NUMA UNICA LISTA, DONDE DECORRE O NOME DO METODO. COM ESTE ALGORITMO, DIVERSAS LISTAS DE SINONIMOS COALESCEM. O DESLOCAMENTO DA CHAVE 74 SERA AGORA APENAS DE 1, IGUAL AO OBTIDO COM O ALGORITMO D (ALGUMAS VEZES PODERIA SER UM POUCO MAIOR), BEM MENOR QUE O DESLOCAMENTO OBTIDO COM O ALGORITMO C, QUE FOI DE 4.

PODERIAMOS MODIFICAR O ALGORITMO E DE MODO QUE AS LISTAS DE SINONIMOS NAO COALESCESSEM, MAS PARA ISSO IRIAMOS MUITAS VEZES NECESSITAR DESLOCAR REGISTROS ANTERIORMENTE ARMAZENADOS. POR EXEMPLO, SE SO NAO TIVESSEMOS INSERIDO O REGISTRO DE CHAVE 18, NA FIGURA 3.3, AO INSERIRMOS O REGISTRO DE CHAVE 74 ELE FICARIA NO SEGUNDO "SLOT" DE B08. SE AGORA FOSSEMOS INSERIR O REGISTRO DE CHAVE 18, AQUELE DE CHAVE 74 TERIA QUE SER DESLOCADO PARA O "BUCKET" B03 (NECESSITANDO DE LISTAS DUPLAMENTE ENCADEADAS OU LISTAS CIRCULARES DE MODO A ACHAR O "BUCKET" ANTERIOR, B04, E MODIFICAR SEU CONTEUDO DO CAMPO LINK PARA B03), PARA ENTAO O REGISTRO DE CHAVE 18 SER INSERIDO EM B08, FICANDO SEPARADA A LISTA DE SINONIMOS DA CHAVE 18, DA LISTA DE SINONIMOS DA CHAVE 74. ESTE ALGORITMO MODIFICADO PODERIA TER UM TEMPO DE RECUPERACAO UM POUCO MELHOR, MAS SERIA MAIS COMPLICADO E TERIA INSERCOES LENTAS, GERALMENTE NAO SENDO COMPENSATORIO.

KNUTH (20), NA PAGINA 517, MOSTRA QUE, COM UMA BOA FUNCAO DE "HASH", AS LISTAS PERMANECEM CURTAS, MESMO QUE O ALGORITMO PERMITA-LHES COALESCEREM. VER FIGURA 3.7.

3.2.7.3- ENCADEAMENTO COM AUXILIO DE UM DIRETORIO DO ESPACO LIVRE:

MARTIN (27) CHAMOU A ATENCAO DE QUE UMA MANEIRA DE, NA AREA PRIMARIA, SEM USARMOS AREA EM SEPARADO PARA "OVERFLOWS", RESOLVERMOS OS "BUCKET OVERFLOWS" SEM CORRERMOS O RISCO DE PESQUISARMOS MUITOS "BUCKETS" (TAL COMO NA ETAPA E.3.1.1 DO ALGORITMO E), E USARMOS UM DIRETORIO DO ESPACO LIVRE. ESTE DIRETORIO INDICA QUAIS "BUCKETS" TEM "SLOTS" LIVRES E, SE QUISERMOS, QUANTOS "SLOTS" LIVRES ELES TEM. NA PESQUISA/INSERCAO DE UM NOVO REGISTRO DE CHAVE I, PRIMEIRAMENTE EXAMINAMOS O "BUCKET" B (T(I)). SE ESTE ESTIVER CHEIO E NAO OBTIVERMOS SUCESSO, USAMOS O CAMPO DE LINK PARA PESQUISARMOS O "PROXIMO" "BUCKET", E ASSIM PROSSEGUIRMOS ATE TERMOS SUCESSO OU ATE TERMOS CERTEZA DA INEXISTENCIA DO REGISTRO PROCURADO, SEJA POR ENCONTRARMOS UM "SLOT" LIVRE, SEJA POR CHEGARMOS A UM "BUCKET" CHEIO MAS COM IN-

DICACAO DE FINAL DE LISTA. NA 1A ALTERNATIVA FAZEMOS A INSERCAO NORMALMENTE. NA 2A ALTERNATIVA EXAMINAMOS O DIRETORIO, DETERMINAMOS UM "BUCKET" NAS PROXIMIDADES E QUE TENHA UM "SLOT" LIVRE, INSERIMOS O REGISTRO PESQUISADO NESTE "SLOT" E ENCADEAMOS ESTE "BUCKET" COM A LISTA DE SINONIMOS.

SEMPRE QUE UM REGISTRO E INSERIDO OU DELETADO O DIRETORIO DEVE SER ATUALIZADO, MAS SO USAMOS O MESMO EM INSERCOES E DELECOES, DE MODO QUE, COM UM ARQUIVO DE BAIXA VOLATILIDADE, NAO TEMOS QUE CONSULTA-LO COM FREQUENCIA.

ESTE METODO E MAIS EFICIENTE PARA "BUCKET SIZE" GRANDE (DIGAMOS, MAIOR OU IGUAL A 20), E SE O FATOR DE CARGA NAO FOR MUITO ALTO (DEVENDO SER NO MAXIMO, DIGAMOS, 0.90). O "BUCKET SIZE" GRANDE TANTO REDUZ O NUMERO DE "OVERFLOWS" (VER FIGURA 1.3), COMO O TAMANHO DO DIRETORIO, AINDA MAIS SE ADAPTARMOS O MESMO PARA INDICAR APENAS OS "BUCKETS" CHEIOS, DESDE QUE O FATOR DE CARGA NAO SEJA ALTO DEMAIS).

3.3 - METODOS COM AREA DE "OVERFLOW" EM SEPARADO

JA DISSEMOS QUE UM REGISTRO QUE TRANSBORDE PODE TANTO SER ARMAZENADO EM ALGUM OUTRO "BUCKET" NAO TOTALMENTE CHEIO DA AREA PRIMARIA DO ARQUIVO, COMO EM UMA AREA SECUNDARIA, FISICAMENTE SEPARADA DA PRIMARIA, CHAMADA "AREA DE OVERFLOW", RESERVADA ESPECIFICAMENTE PARA A RESOLUCAO DE " OVERFLOWS", DE MODO QUE A FUNCAO T (I) E CONSTRUIDA DE MANEIRA QUE NAO POSSA REFERENCIAR DIRETAMENTE ESSA AREA.

SE USARMOS AREA DE "OVERFLOW" EM SEPARADO, SURGIRA UMA PERGUNTA: DEVE HAVER UMA AREA DE "OVERFLOW" ESPECIFICA PARA CADA "BUCKET" QUE TRANSBORDE, OU DEVEM OS "OVERFLOWS" DE MUITOS "BUCKETS" SEREM FUNDIDOS NUMA SO AREA ?

CONFORME MARTIN (27), SAO DUAS AS PRINCIPAIS TECNICAS EM USO:

A) - ENCADEAMENTO DO "OVERFLOW"
B) - ESPACO DISTRIBUIDO PARA O "OVERFLOW"

3.3.1-ENCADEAMENTO DO "OVERFLOW" EM AREA SEPARADA:

A) SE A CAPACIDADE DO "BUCKET" DE "OVERFLOW" FOR 1, O ENCADEAMENTO DO "OVERFLOW" SERA FEITO DE UMA MANEIRA SIMPLES E DIRETA. SE UM REGISTRO TIVER A FALTA DE SORTE DE SER DIRIGIDO (PELA FUNCAO DE "HASH") PARA UM "BUCKET" JA TOTALMEN-

TE CHEIO, ENTAO UM "BUCKET" LIVRE NA AREA DE "OVERFLOW" E TOMADO, O REGISTRO E ARMAZENADO NESTE "BUCKET" DE "OVER - FLOW" E SEU ENDERECO E ARMAZENADO NO "HOME BUCKET". SE UM OUTRO REGISTRO FOR DIRIGIDO PARA O MESMO "HOME BUCKET" JA CHEIO, ENTAO ELE SERA ARMAZENADO EM UM OUTRO "BUCKET" DE "OVERFLOW" E SEU ENDERECO SERA ARMAZENADO NO PRIMEIRO " BUCKET" DE "OVERFLOW", E ASSIM POR DIANTE. DESSA MANEIRA, FORMAREMOS UMA CADEIA DE "BUCKETS" DE "OVERFLOW" A PARTIR DO "HOME BUCKET" (O QUAL SALIENTAMOS QUE PODE TER UMA CA - PACIDADE >= 1). ISTO ESTA ILUSTRADO NA FIGURA 3.4, ADAPTA- DA DE MARTIN (27). SE O "HOME BUCKET SIZE" E O FATOR DE CARGA TIVEREM SIDO ADEQUADAMENTE ESCOLHIDOS, O COMPRI - MENTO MEDIO DA CADEIA CONSERVAR-SE-A BAIXO, SENDO RARA UMA CADEIA TAO LONGA QUANTO A DA FIGURA 3.4. MESMO ASSIM, O RISCO DE MULTIPLAS LEITURAS ("SEEKS") PARA ENCONTRAR UM REGISTRO PODERA SER DIMINUIDO SE OS "BUCKETS" DE "OVER - FLOW" TIVEREM UMA CAPACIDADE MAIOR QUE 1. SEGUNDO KNUTH (20), PAGINA 535, USUALMENTE NAO HA VANTAGEM EM TERMOS A CAPACIDADE DO "BUCKET" DE "OVERFLOW" MAIOR QUE 1, DESDE QUE, COMPARATIVAMENTE, POUCOS "OVERFLOWS" OCORREM. ESTE E O METODO QUE DA GERALMENTE MELHORES RESULTADOS GLOBAIS NA PRATICA, QUANDO O ARQUIVO ESTA NA MEMORIA SECUNDARIA (A NAO SER QUE O "BUCKET SIZE" SEJA RAZOAVELMENTE GRANDE E O FATOR DE CARGA NAO SEJA MUITO ALTO, QUANDO O SIMPLES METO- DO DA PESQUISA LINEAR PODERA SER MAIS ATRAENTE).

B) SE A CAPACIDADE DO "BUCKET" DE "OVERFLOW" FOR MAIOR QUE 1 O PROCEDIMENTO SERA UM POUCO DIFERENTE. NA FIGURA 3.5, TAMBEM ADAPTADA DE MARTIN (27), SUPUSEMOS QUE TANTO O "HO- ME BUCKET" COMO O "OVERFLOW BUCKET" TEM A MESMA CAPACIDA - DE, DIGAMOS, DE 10 REGISTROS. O PRIMEIRO "BUCKET" A TRANS- BORDAR E DESIGNADO PARA UM "BUCKET" NA AREA DE "OVERFLOW". E IMPROVAVEL PREENCHERMOS ESTE "BUCKET" APENAS COM "OVER - FLOWS" DE UM MESMO "HOME BUCKET", ASSIM OS PROXIMOS "BUC - KETS" A TRANSBORDAREM SAO TAMBEM DESIGNADOS PARA O MESMO "BUCKET" DE "OVERFLOW". E IMPROVAVEL QUE O "BUCKET" DE "O- VERFLOW" TAMBEM TRANSBORDE, MAS SE ISTO OCORRER O "BUCKET" DE "OVERFLOW" DEVERA SER DESIGNADO PARA UM OUTRO "OVERFLOW BUCKET", DO MESMO MODO COMO O FORAM OS "HOME BUCKETS".

SE UM REGISTRO FOR DELETADO DE UMA CADEIA CONSTITUIDA DE REGISTROS ISOLADOS, COMO NA FIGURA 3.4, A CADEIA SERA RECO - NECTADA DE MODO BASTANTE SIMPLES. MAS SE UM REGISTRO FOR DE - LETADO DE UMA CADEIA DE "BUCKETS" COMO DA FIGURA 3.5 E SE E - XIGIRMOS QUE A CADEIA NAO CONTENHA "SLOTS" VAZIOS INTERMEDIA- RIOS, A CADEIA NAO PODERA SER RECONECTADA EXCETO COM MUITA DIFICULDADE, ENVOLVENDO MUITOS DESLOCAMENTOS DE REGISTROS. AO INVES DISSO, UMA SOLUCAO MUITO USADA E DEIXARMOS O REGISTRO COM UM SINAL ESPECIAL SIGNIFICANDO "DELETADO", DE MODO QUE UM OUTRO "OVERFLOW" POSSA OCUPA-LO POSTERIORMENTE. SE O ARQUIVO SOFRER MUITAS DELECOES, PODE SER DESEJAVEL REORGANIZA-LO PE -

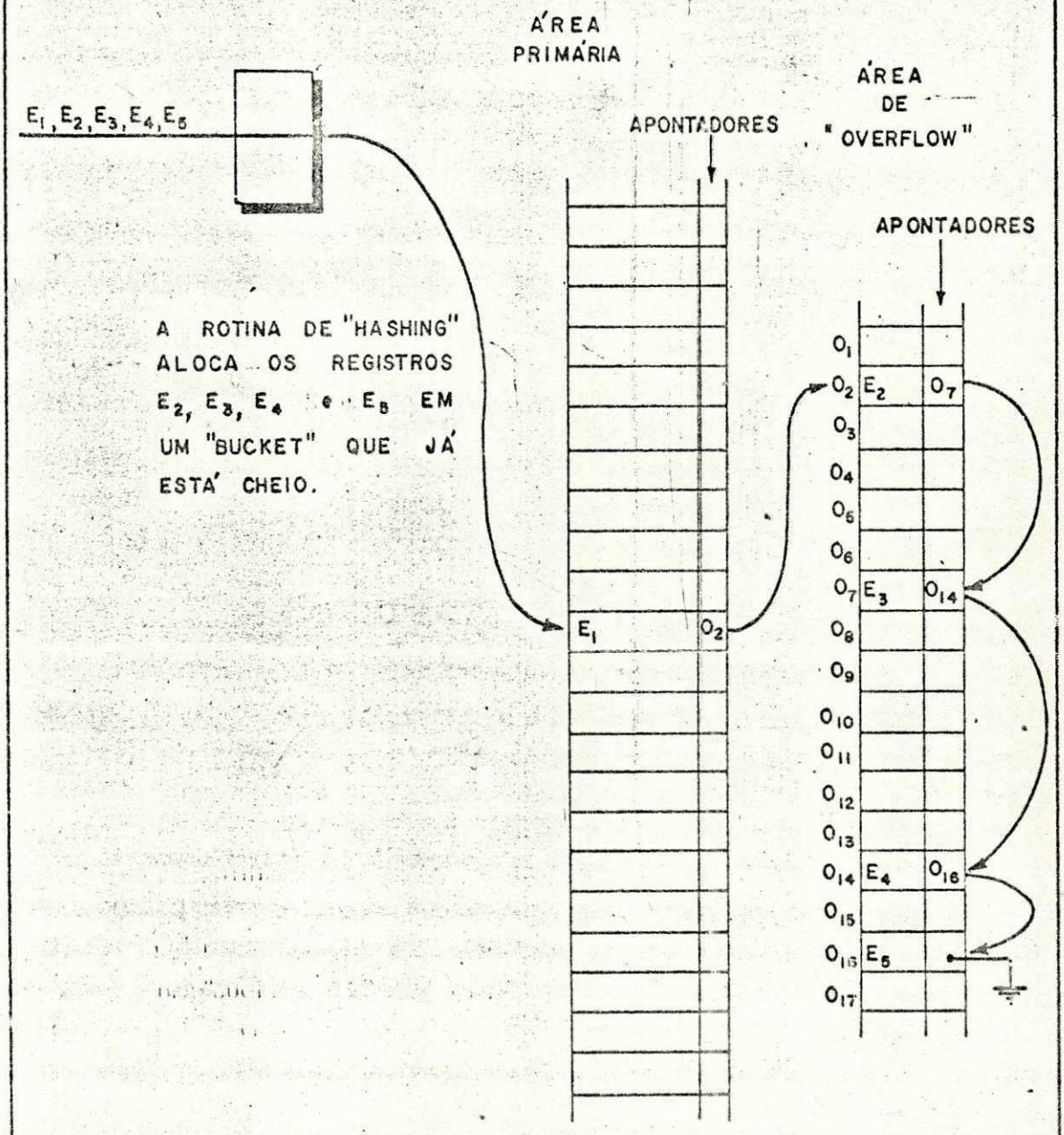


Fig. - 3.4 - Encadeamento do "overflow", com a capacidade dos "buckets" de "overflow" igual a 1. Figura retirada de Martin [27].

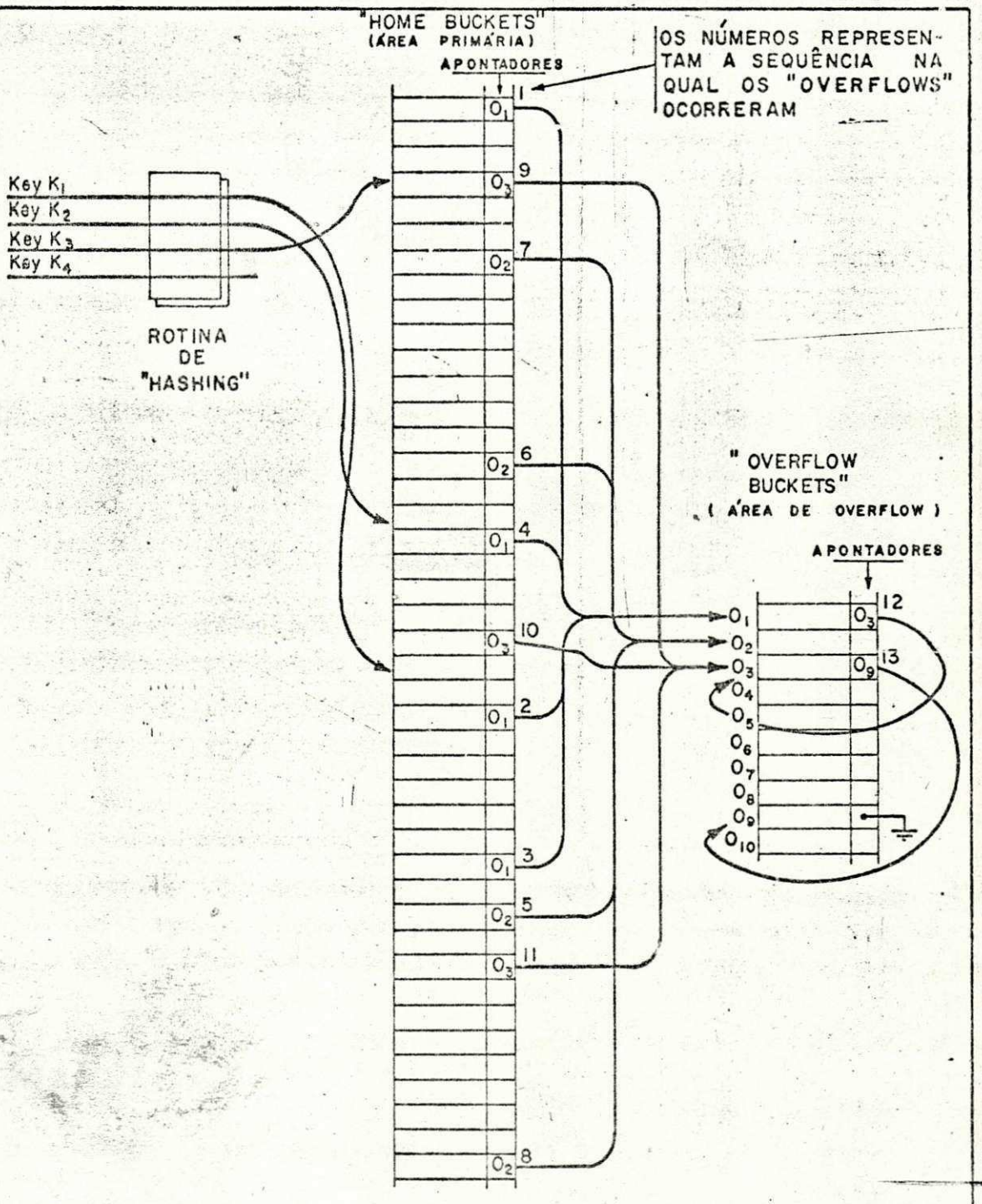


Fig.-3.5 - Encadeamento do "overflow", com a capacidade dos "buckets" de "overflow" maior que 1. Figura retirada de Martin [27].

RIODICAMENTE.

3.3.2-ESPACO DISTRIBUIDO PARA O "OVERFLOW" EM SEPARADO:

PODEMOS, AO INVES DE USAR CADEIAS, DISTRIBUIR "BUCKETS" DE "OVERFLOW" A INTERVALOS REGULARES DENTRO DOS "HOME BUC - KETS".

A FIGURA 3.6, TAMBEM ADAPTADA DE MARTIN (27), ILUSTRA ESTE METODO, ADOTANDO-SE 9 "BUCKETS" COMO O INTERVALO ACIMA REFERIDO. SE UM "HOME BUCKET" TRANSBORDAR, TENTAREMOS ARMAZE- NAR O REGISTRO NO PROXIMO "BUCKET" DE "OVERFLOW". NA MAIORIA DAS VEZES ISTO E CONSEGUIDO, COM A VANTAGEM DE QUE O "BUCKET" DE "OVERFLOW" ESTA FISICAMENTE PROXIMO DO "HOME BUCKET". NAO TEREMOS QUE MANTER CADEIAS E GERALMENTE NAO TEREMOS QUE MOVI- MENTAR O BRACO DE ACESSO NUM DISCO DE CABECA MOVEL.

SE UM "BUCKET" DE "OVERFLOW" TAMBEM TRANSBORDAR, O " BUCKET" DE "OVERFLOW" IMEDIATAMENTE CONSECUTIVO SERA USADO, COMO INDICADO NA FIGURA 3.6, REQUERENDO UM SEGUNDO ACESSO DE "BUCKET" DE "OVERFLOW", E ASSIM POR DIANTE.

NOTEMOS QUE TODOS OS METODOS DE "HASH" VISTOS NO CAPI - TULO 2 ENDERECAM UMA EXTENSAO SEQUENCIAL DE NUMEROS RELATIVOS DE "BUCKETS". UM ALGORITMO DEVE ENTAO SER APLICADO A ESSA EX- TENSAO, DE MODO QUE NAO ENDERECEMOS DIRETAMENTE OS "BUCKETS" DE "OVERFLOW", ISTO E, DE MODO QUE ELES SEJAM "SALTADOS" POR ESTE ULTIMO ALGORITMO. SEJA

IHASH O NUMERO PRODUZIDO DIRETAMENTE PELO ALGORITMO DE " HASH", T, APLICADO SOBRE A CHAVE I. ISTO E, IHASH= T (I).

M O NUMERO DE "HOME BUCKETS" DO ARQUIVO. SE USARMOS O "HASH" POR DIVISAO, USUALMENTE M NAO DEVE SER DIVI - SIVEL POR NUMEROS MUITO PEQUENOS, (DIGAMOS, MENORES QUE 17).

INTERB O INTERVALO ENTRE OS "BUCKETS" DE "OVERFLOW", USU - ALMENTE UM DOS MENORES FATORES DE M. NA FIGURA, INTERB=9.

$\lfloor E \rfloor$ O SIMBOLO PARA O MAIOR INTEIRO CONTIDO EM E, OU SE- JA, "E" ARREDONDADO PARA MENOS.

IBUCKT O NUMERO RELATIVO DO "BUCKET".

ENTAO TEREMOS:

$$IBUCKT = IHASH + \lfloor IHASH/INTERB \rfloor$$

DEIXAMOS A CARGO DO LEITOR OS ALGORITMOS CORRESPONDENTES A ESTE METODO, TANTO PARA PESQUISA E INSERCAO, COMO PARA DELE - CAO.

SAO EXTENSOES DESTE METODO TERMOS UMA SO AREA DE "OVER - FLOW" PARA CADA CILINDRO NO DISCO, OU PARA CADA VOLUME.

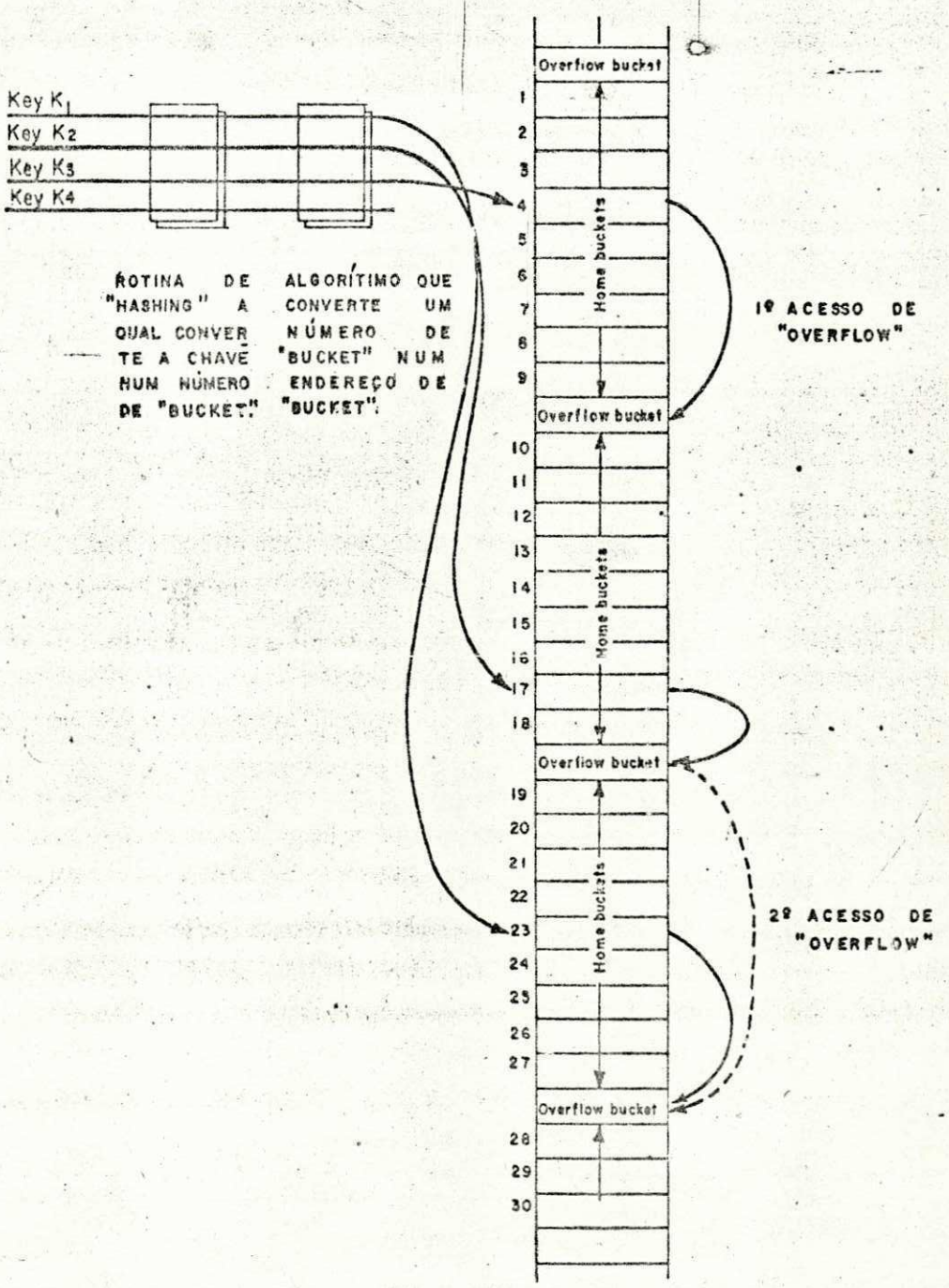


Fig. 3.6- Espaço distribuído para "overflow". Figura retirada de Martin [27].

OLSON (36), ENTRE OUTRAS COISAS, COMPAROU O USO DE ESPACO DISTRIBUIDO PARA O "OVERFLOW", COM O METODO DE PESQUISA LINEAR.

3.3.3-OUTROS METODOS DE "OVERFLOW" EM SEPARADO:

CRIADA UMA AREA EM SEPARADO PARA RESOLUCAO DOS "BUCKET OVERFLOWS", CADA TRANSBORDAMENTO DE "BUCKET" PODE SER RESOLVIDO POR METODOS BEM SEMELHANTES AOS VISTOS EM 3.2, POREM ADAPTADOS, E CLARO, PARA SO RESOLVEREM OS EVENTUAIS E SUCESSIVOS "BUCKET OVERFLOWS" DENTRO DESSA AREA. ASSIM, TEREMOS OS SEGUINTES METODOS PARA RESOLUCAO DOS "BUCKET OVERFLOWS":

1 . - PESQUISA LINEAR DENTRO DA AREA DE "OVERFLOW" EM SEPARADO, MAS COM O "HOME BUCKET" APONTANDO PARA O "BUCKET" DE "OVERFLOW" ONDE DEVEMOS INICIAR A PESQUISA LINEAR.
2 . - PESQUISA POR DUPLO "HASH", MAS COM A APLICACAO SUCESSIVA DO SEGUNDO "HASH" RESTRINGINDO-SE A AREA DE "OVERFLOW" EM SEPARADO.
3 . - PESQUISA QUADRATICA DENTRO DA AREA DE "OVERFLOW" EM SEPARADO.
4 . - PESQUISA RANDOMICA, MAS COM TODOS OS EVENTUAIS ENDERECAMENTOS SUBSEQUENTES AO PRIMEIRO REFERINDO-SE A AREA DE "OVERFLOW" EM SEPARADO.
5 . - PESQUISA DE VYSSOTSKY, MAS COM TODAS AS EVENTUAIS TRANSFORMACOES SUBSEQUENTES A PRIMEIRA TRANSFORMACAO REFERINDO-SE A AREA DE "OVERFLOW" EM SEPARADO.

DE UM MODO GERAL, CADA UM DESTES METODOS TEM, NA MAIORIA DAS VEZES, UM DESEMPENHO UM POUCO MELHOR, EM NUMERO DE ACESSOS A MEMORIA SECUNDARIA, QUE O SEU CORRESPONDENTE EM 3.2.

DEIXAMOS A CARGO DO LEITOR O DESENVOLVIMENTO DOS ALGORITMOS CORRESPONDENTES AOS METODOS ACIMA SUGERIDOS.

3.4 - DELECOES

A MAIORIA DOS PROGRAMADORES QUE ESTUDA O "HASHING" PELA PRIMEIRA VEZ SURPREENDE-SE AO VERIFICAR QUE AS DELECOES EM ARQUIVOS CONSTRUIDOS COM METODOS DE "HASH" NAO SAO TAO TRIVIAIS COMO PARECEM A PRIMEIRA VISTA. COM EFEITO, A MANEIRA APA-

RENTEMENTE OBVIA PARA DELETAR UM REGISTRO MUITAS VEZES NAO FUNCIONA. POR EXEMPLO, SUPONHAMOS QUE JA INSERIMOS NO "SLOT" R17 DA FIGURA 3.1 O REGISTRO DE CHAVE 74, PELO ALGORITMO C; SE AGORA QUISERMOS DELETAR O REGISTRO DE CHAVE 24, NAO PODEREMOS SIMPLESMENTE MARCA-LO COMO DESOCUPADO, PORQUE ASSIM O REGISTRO DE CHAVE 74 SERIA "ESQUECIDO", ISTO E, NAO PODERIAMOS MAIS TER ACESSO AO MESMO PELO ALGORITMO C, UMA VEZ QUE O "BUCKET" B (T(74))=B04 TERIA UM "SLOT" VAZIO E A PESQUISA PARARIA AI, COM INSUCESSO.

TEMOS DUAS CLASSES DE SOLUCOES, QUE VEREMOS A SEGUIR.

3.4.1-SOLUCAO SEM DESLOCAMENTO DE REGISTROS:

CONSISTE EM MARCARMOS COM UM CODIGO ESPECIAL O REGISTRO QUE QUEREMOS DELETAR, DE MODO QUE TEREMOS 3 TIPOS DE "SLOTS": DESOCUPADO, OCUPADO E DELETADO. AO PESQUISARMOS UMA CHAVE, PODEMOS "SALTAR" OS "SLOTS" DELETADOS, COMO SE ESTIVESSEM OCUPADOS. NO CASO DE OBTERMOS INSUCESSO NUMA PESQUISA, PODEMOS INSERIR O REGISTRO PESQUISADO NUM CERTO "SLOT", SELECIONADO PELO ALGORITMO ENTRE AQUELES DESOCUPADOS OU DELETADOS QUE TENHAM SIDO ENCONTRADOS.

DO MODO QUE EXPUSEMOS A IDEIA, ELA SO SERIA VIAVEL SE O ARQUIVO FOSSE POUCO VOLATIL, PORQUE OS "SLOTS" NAO MAIS VOLTARIAM A SER DESOCUPADOS DEPOIS QUE FOSSEM OCUPADOS UMA VEZ. APOS UMA GRANDE SUCESSAO DE INSERCOES E DELECOES, TODOS OS "SLOTS" VAZIOS DESAPARECERIAM, E CADA PESQUISA COM INSUCESSO EXIGIRIA M TENTATIVAS. ALEM DISSO, MESMO NUMA PESQUISA COM SUCESSO, O NUMERO DE ACESSOS A MEMORIA SECUNDARIA SERIA UM POUCO MAIOR, PARA UM ARQUIVO VELHO, QUE PARA UM ARQUIVO NOVO COM O MESMO NUMERO DE "SLOTS" VALIDOS.

NO ENTANTO, PODEMOS APERFEICOAR A IDEIA DE MODO A SER VALIDA MESMO PARA ARQUIVOS DE ALTA VOLATILIDADE: BASTA QUE REORGANIZEMOS O ARQUIVO E/OU SUA AREA DE "OVERFLOW" EM PERIODOS APROPRIADOS, OU QUANDO O ARQUIVO JA TENHA CHEGADO A UM CERTO GRAU DE DEGRADACAO, EMITINDO UM "SINAL DE ALERTA". LOGICAMENTE, NUM AMBIENTE DE "DATA BASE" QUE FACA PROCESSAMENTO EM TEMPO REAL, COMO NUMA REDE BANCARIA, ESTA REORGANIZACAO DEVE SER FEITA NUMA OCASIAO QUE NAO TRAGA PROBLEMAS (O QUE, NO EXEMPLO, PODERIA SER O PERIODO NOTURNO OU DO FIM DE SEMANA).

PARA MAIORES DETALHES, VER MORRIS (33), KNUTH (20) E MARTIN (27).

3.4.2-SOLUCAO COM DESLOCAMENTO DE REGISTROS:

SE ESTIVERMOS USANDO UM ALGORITMO DIFERENTE DO C (PES -

QUISA LINEAR) PARA A PESQUISA E INSERCAO, TALVEZ POSSAMOS CHEGAR A UM ALGORITMO PARA A DELECAO, DENTRO DO MESMO ESPIRITO DO QUE APRESENTAREMOS ABAIXO, MAS ESSE ALGORITMO A QUE TALVEZ CHEGUEMOS DEVERA SER MUITO COMPLICADO, RAZAO PORQUE PREFERIMOS GERALMENTE O PROCEDIMENTO SUGERIDO EM 3.4.1.

CONTUDO, SE ESTIVERMOS USANDO O ALGORITMO C PARA RESOLUCAO DE SINONIMOS POR PESQUISA LINEAR, PODEMOS FAZER A DELECAO DE UM REGISTRO DE UM MODO QUE, EMBORA MAIS TRABALHOSO, NAO DEGRADA O ARQUIVO A MEDIDA QUE ELE ENVELHECE, NAO NECESSITA MARCAR "SLOTS" COMO DELETADOS, NEM REORGANIZAR PERIODICAMENTE O ARQUIVO. PARA ISSO, NECESSITAMOS DESLOCAR ALGUNS REGISTROS APOS O DELETADO, DE MODO QUE CADA CHAVE FIQUE TAO PERTO QUANTO POSSIVEL DO SEU "HOME BUCKET", INEXISTINDO "SLOTS" VAZIOS INTERMEDIARIOS NUMA SEQUENCIA DE SINONIMOS.

- ALGORITMO F:

SUPONDO-SE QUE JA SE USOU O ALGORITMO C PARA LOCALIZAR O REGISTRO DE CHAVE I, NO "SLOT" IREGI DO "BUCKET" NBUCKT, ESTE ALGORITMO DELETA AQUELE REGISTRO E DESLOCA OS REGISTROS SEGUINTES PARA O MAIS PROXIMO POSSIVEL DO INICIO DOS SEUS "HOME BUCKETS".

ARGUMENTOS DE ENTRADA:

IREGI, INICBK, NRGBKT, IBUCKT, M, T, TODOS COM O MESMO SIGNIFICADO QUE TINHAM NO ALGORITMO C (VER 3.2.1)

OUTROS SIMBOLOS:

IATRAS E O APONTADOR PARA O "SLOT" VAZIO QUE ESTA PARATRAS DO REGISTRO IREGI SENDO EXAMINADO, E PARA O QUAL ELE TALVEZ SEJA DESLOCADO.

IHOME E O APONTADOR PARA O INICIO DO "HOME BUCKET" DA CHAVE DO REGISTRO IREGI SENDO EXAMINADO.

F.1 - (DELETA UM REGISTRO)
MARQUE R (IREGI) COMO VAZIO E FACA IATRAS=IREGI

F.2 - (PASSA PARA O PROXIMO REGISTRO)
IREGI=IREGI+1
SE IREGI<=INICBK+(NRGBKT-1) VA PARA F.3.
SE NAO, IREGI ESTA NUM NOVO "BUCKET".
SE IREGI>=NRGBKT*M FACA IREGI=0 E VA PARA F.3.
SE NAO:

F.3 - (INSPECIONA R (IREGI))
SE R (IREGI) ESTA VAZIO, ENTAO "FIM".
SE NAO: FACA IHOME=T (CHAVE DE R (IREGI))*NRGBKT, O INICIO DO "HOME BUCKET" DA CHAVE AGORA ARMAZENADA EM IREGI.
SE IREGI>=IHOME>IATRAS
OU
IHOME>IATRAS>IREGI

OU
IATRAS>IRE I>=IHOME
ISTO E, SE IHOME ESTIVER CICLICAMENTE NO ESPACO DES E IATRAS (EXCLUSIVE) ATE IREGI (INCLUSIVE) VA PARA F.2.
SE NAO:

F.4 - (DESLOCA O CONTEUDO DE UM REGISTRO)
FACA R (IATRAS)=R (IRE I) E VA PARA F.1.

DELECOES EM ARQUIVOS Q E FORAM CONSTRUIDOS RESOLVENDO OS "BUCKET OVERFLOWS" POR ENCADEAMENTO (QUER EM LISTAS SEPA - RADAS, QUER EM LISTAS COALESCENTES), SAO RELATIVAMENTE FA - CEIS, ESPECIALMENTE SE TOMARMOS AS LISTAS DUPLAMENTE ENCADEA- DAS, E COLOCARMOS EM CADA "BUCKET" UM INDICADOR DE QUANTOS " SLOTS" ESTAO VAZIOS. UM "BUCKET" SERA RETIRADO DA SUA LISTA DUPLAMENTE ENCADEADA SOMENTE QUANDO ESTE INDICADOR TORNAR-SE ZERO.

3.5 - OTIMIZACAO DO "HASHING" LEVANDO EM CONTA A FREQUENCIA DE UTILIZACAO DOS REGISTROS

A IDEIA ABAIXO FOI PRIMEIRAMENTE EXPOSTA POR HEISING (15), EM 1963, QUE NOTOU QUE, NUM ARQUIVO COMERCIAL TIPICO, A ATIVIDADE NAO E RANDOMICAMENTE DISTRIBUIDA ENTRE OS REGIS - TROS, AO CONTRARIO, APROXIMADAMENTE 80% DAS ATIVIDADES RECAEM SOBRE 20% DOS REGISTROS.

DE UM MODO BASTANTE SIMPLES, PODEMOS OTIMIZAR A PERFOR- MANCE DE UM ARQUIVO ENDERECADO POR UM METODO QUALQUER DE "HASH": A PROPORCAO DE REGISTROS QUE SOFREM "OVERFLOW" E RE - LATIVAMENTE PEQUENA, MAS ELES TOMAM UM TEMPO DE ACESSO MAIOR QUE OS OUTROS, RAZAO PELA QUAL O IDEAL SERIA QUE OS REGISTROS QUE SOFRESSEM "OVERFLOW" FOSSEM AQUELES MENOS FREQUENTEMENTE USADOS; PARA CONSEGUIRMOS ISTO QUANDO O ARQUIVO E CARREGADO INICIALMENTE, OS REGISTROS MAIS FREQUENTEMENTE REFERENCIADOS DEVEM SER CARREGADOS PRIMEIRAMENTE E AQUELES MENOS FREQUENTE- MENTE REFERENCIADOS POR DERRADEIRO; SE POSSIVEL, O IDEAL E QUE O CARREGAMENTO DOS REGISTROS SEJA FEITO EM PERFEITA ORDEM DECRESCENTE DA SUA "POPULARIDADE", ISTO E, DA FREQUENCIA COM QUE SAO REFERENCIADOS.

PODEREMOS IR FAZENDO ESTATISTICAS A RESPEITO DA FRE - QUENCIA DE REFERENCIAMENTO OS REGISTROS, A MEDIDA QUE O AR - QUIVO VAI SENDO USADO, DE MODO QUE ESSAS ESTATISTICAS POSSAM SER USADAS NA MELHORIA DO DESEMPENHO DO ARQUIVO QUANDO O MES- MO FOR RECARREGADO. TAL COMO EM MUITAS ORGANIZACOES FISICAS DE "DATA BASES", A REORGANIZACAO PERIODICA DOS ARQUIVOS (QUER FEITA OU NAO AUTOMATICAMENTE) PODE MELHORAR SUA PERFORMANCE.

KNUTH (20), PAGINAS 398 A 401, DISCUTE ALTERNATIVAS PA- RA A MANUTENCAO DE ARQUIVOS AUTO-ORGANIZAVEIS ("SELF-ORGANI - ZING FILES"). VER TAMBEM MARTIN (27).

3.6 - ANALISE TEORICO-COMPARATIVA DOS METODOS DE RESOLUCAO DE "BUCKET OVERFLOWS"

NAO E NOSSO OBJETIVO, NESTE TRABALHO, APRESENTAR A ANA- LISE TEORICO-COMPARATIVA DOS DIVERSOS METODOS DE RESOLUCAO DE "BUCKET OVERFLOWS", MAS SIM APENAS OS RESULTADOS E CONCLUSOES A QUE OS ESTUDIOSOS DO ASSUNTO JA CHEGARAM.

ESTES ESTUDOS BASEIAM-SE NA HIPOTESE DE QUE A FUNCAO DE "HASH" E AS CHAVES SAO PERFEITAMENTE RANDOMICAS, E PODEM SER ENCONTRADOS EM KNUTH (20). VER TAMBEM PETERSON (37), SCHAY & SPRUTH (42) E ULLMAN (46).

NOTEMOS QUE, COMO GERALMENTE PESQUISAS BEM SUCEDIDAS SAO MUITO MAIS FREQUENTES QUE PESQUISAS MAL SUCEDIDAS, O NU - MERO MEDIO DE ACESSOS NECESSARIOS PARA UMA PESQUISA BEM SUCE- DIDA E UM INDICE BEM MAIS SIGNIFICATIVO QUE O NUMERO MEDIO DE ACESSOS NECESSARIOS PARA UMA PESQUISA MAL SUCEDIDA.

3.6.1-QUANDO, USUALMENTE EM MEMORIA PRINCIPAL OU PAGINADA, O "BUCKET SIZE" E IGUAL A 1:

1.-SUPONHAMOS O "HASHING" 'UNIFORME', ISTO E, QUE CADA UMA DAS C (M,N) CONFIGURACOES DE N "SLOTS" OCUPADOS E M-N DE - SOCUPADOS E IGUALMENTE PROVAVEL, DESPREZANDO ASSIM O AMON- TOAMENTO ("CLUSTERING") PRIMARIO E SECUNDARIO.

SEJA:

N=NUMERO DE "SLOTS" OCUPADOS, NO ARQUIVO,
M=NUMERO TOTAL DE "SLOTS" DO ARQUIVO,
P=N/M=FATOR DE CARGA DO ARQUIVO E

C (M,N)=COMBINACAO DE M ELEMENTOS, TOMADOS N A N.

TEORICAMENTE, DADA UMA CHAVE QUALQUER, O NUMERO MEDIO DE ACESSOS NECESSARIOS, PARA TODOS OS METODOS DE PESQUISA NAO ENCADEADA, SERA:

-PESQUISA BEM SUCEDIDA:

C=(M+1)*(1/(M+1)+1/M+...+1/(M-N+2))/N

≈LOG (1/(1-P))/P

-PESQUISA MAL SUCEDIDA

C'=(M+1)/(M-N+1)

≈1/(1-P)=1+P+P**2+P**3+...

NA PRATICA, APENAS OS METODOS DE PESQUISA POR DUPLO "HASH" (3.2.2), O DE VYSSOTSKY (3.2.6) E A VARIACAO DE KRUTAR EM TORNO DA PESQUISA LINEAR (3.2.1) COMPORTAM-SE EXATAMENTE SEGUNDO AS FORMULAS ACIMA, PARA TODOS FINS PRATICOS, ENQUANTO OS METODOS DE PESQUISA LINEAR, QUADRATICA E RANDOMICA TEM UM DESEMPENHO PIOR. LEMBREMO-NOS, NO ENTANTO, QUE NA MEMORIA PRINCIPAL, O ESFORCO DE CALCULO DAQUELES METODOS (EXCETO O DA VARIACAO DE KRUTAR) PODE ANULAR COMPLETAMENTE SUAS VANTAGENS SOBRE UM METODO MAIS SIMPLES, COMO O DA PESQUISA LINEAR.

VER EM KNUTH (20), PAGINA 523, OS RESULTADOS DA ANALISE PARA O METODO DO DUPLO "HASH", POREM COM AS TRANSFORMACOES T (I) E T1 (I) NAO INDEPENDENTES.

2.-SUPONHAMOS QUE CADA UMA DAS M**N SEQUENCIAS

A(1) A(2)...A(N) 0<=A(J)<M

E IGUALMENTE PROVAVEL, ONDE A (J) E O "HOME ADDRESS" OU "HASH" DA J-ESIMA CHAVE INSERIDA NO ARQUIVO, O QUE EQUIVALE A CONSIDERARMOS O EFEITO DOS AMONTOAMENTOS PRIMARIO E SECUNDARIO, CONSTITUINDO-SE ASSIM NUM MODELO MATEMATICO BEM MAIS PLAUSIVEL QUE O ANTERIOR, EMBORA, NA PRATICA, OS METODOS AINDA TENHAM UM DESEMPENHO LEVEMENTE SUPERIOR AO DESCRITO PELAS FORMULAS QUE SE SEGUEM.

DADA UMA CHAVE QUALQUER, O NUMERO DE ACESSOS SERA:

A) METODO DA PESQUISA LINEAR:

-PESQUISA BEM SUCEDIDA:

C=(1+Q0(M,N-1))/2=(1+(M-1)/(M-N))/2=1+P/(2*(1-P))

=(1/2)*(1+1/(1-P))

-PESQUISA MAL SUCEDIDA:

C'=(1+Q1(M,N))/2=(1/2)*(1+(1/(1-P))**2)

ONDE

QR (M,N)=SOMATORIO PARA K>=0 DE

C (R+K,K)*(N/M)*((N-1)/M)*...*((N-K+1)/M), PARA R=0 OU R=1

O METODO TORNA-SE RAPIDAMENTE INEFICIENTE, A MEDIDA QUE P TENDE PARA 1.

B) METODO DA PESQUISA ENCADEADA COM LISTAS COALESCENTES:

-PESQUISA BEM SUCEDIDA:

C=1+(1/8)*(M/N)*((1+2/M)**N-1-2*N/M)+(1/4)*(N-1)/M

=1+(1/(8*P))*(E**(2*P)-1-2*P)+P/4

-PESQUISA MAL SUCEDIDA:

C'=1+(1/4)*((1+2/M)**N-1-2*N/M)
=1+(1/4)*(E**(2*P)-1-2*P)

O METODO E BASTANTE EFICIENTE, MESMO PARA FATORES DE CARGA ALTOS.

O NUMERO MEDIO DE SUCESSOS LOGO NA 1A TENTATIVA, QUANDO SO PESQUISAMOS CHAVES JA EXISTENTES, É:
S=1-(N-1)/(2*M)=1-P/2.
QUANDO, ALEM DA AREA PRIMARIA COM M "SLOTS", DISPOMOS DUMA AREA EM SEPARADO, ESPECIFICAMENTE RESERVADA PARA RESOLUCAO DE COLISOES, SE JA TEMOS M+L REGISTROS INSERIDOS, O PROXIMO REGISTRO A SER INSERIDO NECESSITARA EM MEDIA DE 1L/(2*M)+1/4)*((1+2/M)**M-1)+1/2 ACESSOS. KNUTH (14) ATRIBUI ESTA FORMULA A L. GUIBAS.

C) METODO DA PESQUISA ENCADEADA COM LISTAS SEPARADAS:

-PESQUISA BEM SUCEDIDA:
C=1+(N-1)/(2*M)
=1+P/2

-PESQUISA MAL SUCEDIDA:
C'=(1-1/M)**N+N/M
=E**(-P)+P

-O METODO E MUITO EFICIENTE, MESMO PARA FATORES DE CARGA ALTOS. AS FORMULAS SAO VALIDAS MESMO PARA P>1, USANDO-SE AREA DE "OVERFLOW" EM SEPARADO, SENDO P O FATOR DE CARGA PRINCIPAL.

3.6.2-QUANDO, USUALMENTE EM MEMORIA SECUNDARIA, O "BUCKET SIZE", B, E MAIOR QUE 1:

SEJA:
B="BUCKET SIZE",
N=NUMERO DE "SLOTS" OCUPADOS NO ARQUIVO,
M=NUMERO TOTAL DE "BUCKETS" DO ARQUIVO,
P=N/(M*B) FATOR DE CARGA DO ARQUIVO,
R (P,K)=K/(K+1)+K**2*P/((K+1)*(K+2))+
+K**3*P**2/((K+1)*(K+2)*(K+3))+...
TK (P)=E**(-K*P)*K**K*P**K*(1-(1-P)*R (P,K))/K
OBS : O K DE TK E UM INDICE . ASSIM TEMOS T1 (P) ; T2 (P),...
O (1/M)=QUANTIDADE QUE NAO E EXPLICITAMENTE CONHECIDA, EXCETO QUE SUA MAGNITUDE NAO E MUITO GRANDE. LER "O GRANDE " ("BIG OH").
E ———=BASE DOS LOGARITMOS NEPERIANOS (2.718..)

1.-PARA PESQUISA LINEAR, O NUMERO MEDIO DE ACESSOS PARA RECUPERACAO DE UMA CHAVE QUALQUER E

-PESQUISA BEM SUCEDIDA:
C=1+TB (P)+T2*B (P)+T3*B (P)+...

-PESQUISA MAL SUCEDIDA:
NAO ENCONTRAMOS FORMULA MATEMATICA DESENVOLVIDA NA LITERATURA.
-ESTE METODO E SUAS MELHORES VARIACOES, PARA B>1 E FATOR DE CARGA NAO MUITO ALTO E MELHOR QUE OS DESCRITOS DE 3.2.2 A 3.2.6, PORQUE O INCREMENTO ISKIP PODE SER ESCOLHIDO DE MODO QUE MINIMIZE O TEMPO DE LATENCIA ("LATENCY DELAY") ENTRE ACESSOS CONSECUTIVOS. NA VERDADE, PARA UM "BUCKET SIZE" RAZOAVELMENTE GRANDE (DIGAMOS, B>=20) E FATOR DE CARGA NAO MUITO GRANDE (DIGAMOS P<=0.85), ESTE METODO PODE MESMO EQUIPARAR-SE AO DA PESQUISA ENCADEADA, OU ATE SOBREPUJA-LO, SENDO BASTANTE SIMPLES.
-VEJAMOS A TABELA 3.1 ABAIXO, RETIRADA DE KNUTH (20)

2.-PARA PESQUISA ENCADEADA COM LISTAS SEPARADAS (*), E CAPACIDADE DO "BUCKET" DE "OVERFLOW" IGUAL A 1, O NUMERO MEDIO DE ACESSOS PARA RECUPERACAO DUMA CHAVE QUALQUER É:
-PESQUISA BEM SUCEDIDA:

$$C = 1 + (1 - B*(1-P)/2*TB(P) + E**(-B*P)* B**B*P**B*R(P,B)/(2*B) + O(1/M)$$

-PESQUISA MAL SUCEDIDA:

$$C' = 1 + P*B*TB(P) + O(1/M)$$

PARA "BUCKET SIZES" MENORES QUE 20 OU PARA FATORES DE CARGA BASTANTE ALTOS ESTE METODO E MELHOR QUE O DA PESQUISA LINEAR. NOS DEMAIS CASOS E EQUIPARAVEL OU POUCO PIOR.
-VEJAMOS AS TABELAS 3.2 E 3.3 ABAIXO, RETIRADAS DE KNUTH (20), E QUE PODEM SERVIR INCLUSIVE PARA ESTIMARMOS O MINIMO NECESSARIO PARA A AREA DE "OVERFLOW" EM SEPARADO=M* (C'-1).

3.-OUTROS METODOS DE PESQUISA PARA RESOLUCAO DOS "BUCKET OVERFLOWS" COM O "BUCKET SIZE" MAIOR QUE 1, NAO FORAM AINDA ANALISADOS, PARA ARQUIVOS NA MEMORIA SECUNDARIA, MAS OS MAIS PASSIVEIS DE USO PARECEM SER:
A) METODO DA PESQUISA ENCADEADA COM LISTAS SEPARADAS E COM A CAPACIDADE DOS "BUCKETS" DE "OVERFLOW" MAIOR QUE 1. (ALGORITMO D).
B) METODO DA PESQUISA ENCADEADA COM LISTAS COALESCENTES, SEM AREA DE "OVERFLOW" EM SEPARADO E COM A CAPACIDADE DOS "BUCKETS" DE "OVERFLOW" MAIOR QUE 1 (ALGORITMO E, POSSIVELMENTE MELHORADO PELO USO DE LISTAS DUPLAMENTE ENCADEADAS E INDICES OU CONTADORES DE "SLOTS" DESOCUPADOS, DENTRO DE CADA "BUCKET").

4.-OLSON (36) APRESENTOU UMA EXCELENTE DISCUSSAO DAS CONSIDERACOES PRATICAS ENVOLVIDAS NO PROJETO DE UM ARQUIVO EM MEMORIA SECUNDARIA PELO USO DE METODOS DE "HASH". ACONSELHAMOS O LEITOR INTERESSADO EM APROFUNDAR-SE NO ASSUNTO A CONSULTAR O REFERIDO ARTIGO.

---

* NUMA AREA ESPECIFICA PARA "OVERFLOW".

TABELA 3.1, retirada de Knuth (20)

NÚMERO MÉDIO DE ACESSOS NECESSÁRIOS NUMA PESQUISA BEM SUCEDIDA, (geralmente mais importante que numa mal sucedida)
PARA "OVERFLOW" POR PESQUISA LINEAR

"Bucket size", b	FATOR DE CARGA, P									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	95%
1	1.0556	1.1250	1.2143	1.3333	1.5000	1.7500	2.167	3.000	5.500	10.5
2	1.0062	1.0242	1.0553	1.1033	1.1767	1.2930	1.494	1.903	3.147	5.6
3	1.0009	1.0066	1.0201	1.0450	1.0872	1.1584	1.286	1.554	2.378	4.0
4	1.0001	1.0021	1.0085	1.0227	1.0497	1.0984	1.190	1.386	2.000	3.2
5	1.0000	1.0007	1.0039	1.0124	1.0307	1.0661	1.136	1.289	1.777	2.7
10	1.0000	1.0000	1.0001	1.0011	1.0047	1.0154	1.042	1.110	1.345	1.8
20	1.0000	1.0000	1.0000	1.0000	1.0003	1.0020	1.010	1.036	1.144	1.4
50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.001	1.005	1.040	1.1

TABELA 3.2, retirada de Knuth (20)

NÚMERO MÉDIO DE ACESSOS NECESSÁRIOS NUMA PESQUISA BEM SUCEDIDA, (geralmente mais importante que numa mal sucedida), PARA "OVERFLOW" ENCADEADO, LISTAS SEPARADAS E CAPACIDADE DO "BUCKET" DE "OVERFLOW" IGUAL A 1

FATOR DE CARGA, P

"Bucket size", b	10%	20%	30%	40%	50%	60%	70%	80%	90%	95%
1	1.0500	1.1000	1.1500	1.2000	1.2500	1.3000	1.350	1.400	1.450	1.5
2	1.0063	1.0242	1.0520	1.0883	1.1321	1.1823	1.238	1.299	1.364	1.4
3	1.0010	1.0071	1.0216	1.0458	1.0806	1.1259	1.181	1.246	1.319	1.4
4	1.0002	1.0023	1.0097	1.0257	1.0527	1.0922	1.145	1.211	1.290	1.3
5	1.0000	1.0008	1.0046	1.0151	1.0358	1.0699	1.119	1.186	1.286	1.3
10	1.0000	1.0000	1.0002	1.0015	1.0070	1.0226	1.056	1.115	1.206	1.3
20	1.0000	1.0000	1.0000	1.0000	1.0005	1.0038	1.018	1.059	1.150	1.2
50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.001	1.015	1.083	1.2

TABELA 3.3, retirada de Knuth (20)

NÚMERO MÉDIO DE ACESSOS NECESSÁRIOS NUMA PESQUISA MAL SUCEDIDA, PARA "OVERFLOW" ENCADEADO, LISTAS SEPARADAS E CAPACIDADE DO "BUCKET" DE "OVERFLOW" IGUAL A 1

(Curiosamente, quando P tende para 1 o número de acessos necessários cresce com b = "home bucket size")

FATOR DE CARGA, P

ucket ize", b	10%	20%	30%	40%	50%	60%	70%	80%	90%	95%
1	1.0048	1.0187	1.0408	1.0703	1.1065	1.1488	1.197	1.249	1.307	1.3
2	1.0012	1.0088	1.0269	1.0581	1.1036	1.1638	1.238	1.327	1.428	1.5
3	1.0003	1.0038	1.0162	1.0433	1.0898	1.1588	1.252	1.369	1.509	1.6
4	1.0001	1.0016	1.0095	1.0314	1.0751	1.1476	1.253	1.394	1.571	1.7
5	1.0000	1.0007	1.0056	1.0225	1.0619	1.1346	1.249	1.410	1.620	1.7
10	1.0000	1.0000	1.0004	1.0041	1.0222	1.0773	1.201	1.426	1.773	2.0
20	1.0000	1.0000	1.0000	1.0001	1.0028	1.0234	1.113	1.367	1.898	2.3
50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0007	1.018	1.182	1.920	2.7

3.6.3-COMPARACAO TEORICA ENTRE OS METODOS:

E DIFICIL RESUMIRMOS EM POUCAS PALAVRAS OS ASPECTOS, MESMO SO OS IMPORTANTES, RELATIVOS AS COMPENSACOES ("TRADE-OFFS") ENVOLVIDAS NA COMPARACAO ENTRE OS METODOS DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS". APESAR DISSO, OS PONTOS ABAIXO NOS PARECEM SER OS DE MAIOR RELEVANCIA:

1.- QUANDO O "BUCKET SIZE" E IGUAL A 1 (USUALMENTE EM MEMORIA PRINCIPAL OU PAGINADA), A FIGURA 3.7, RETIRADA DE KNUTH (20) SUMARIZA O NUMERO MEDIO DE ACESSOS NECESSARIOS PARA DIVERSOS METODOS. MAS A FIGURA NAO DIZ TUDO, POIS O TEMPO MEDIO PARA OS ACESSOS NAO E SEMPRE PROPORCIONAL AO NUMERO MEDIO DE ACESSOS, HAVENDO VARIACOES SEGUNDO AS SUBVARIANTES ADOTADAS PARA OS METODOS, A PROGRAMACAO, AS CARACTERISTICAS PECULIARES DA APLICACAO, ETC. COMO ILUSTRACAO, VER KNUTH (20), PAGINA 524.

O METODO DE PESQUISA LINEAR E O QUE NECESSITA DO MAIOR NUMERO DE ACESSOS AO ARQUIVO, MAS TEM A VANTAGEM DA SIMPLICIDADE, RAZAO PORQUE, SENDO A PRIMEIRA VISTA O PIOR, PELA OBSERVACAO DA FIGURA 3.7, E ALGUMAS VEZES O METODO ESCOLHIDO QUANDO O FATOR DE CARGA NAO E ALTO (DIGAMOS, E MENOR QUE 0.60) E O ARQUIVO E POUCO VOLATIL.

DOS OUTROS METODOS QUE NAO USAM ENCADEAMENTO, A VARIACAO DE BRENT EM TORNO DO METODO DE DUPLO "HASH" E RELATIVAMENTE O QUE NECESSITA DO MENOR NUMERO MEDIO DE ACESSOS, PRINCIPALMENTE SE O FATOR DE CARGA FOR JA RELATIVAMENTE ALTO (DIGAMOS, MAIOR QUE 0.85) E NOS CASOS DE PESQUISA COM SUCESSO. NOTEMOS, NO ENTANTO, QUE ESTE METODO E BASTANTE COMPLICADO, EM RELACAO AOS OUTROS.

OS METODOS DE PESQUISA ENCADEADA SAO RELATIVAMENTE MUITO EFICIENTES QUANTO AO NUMERO DE ACESSOS NECESSARIOS, PRINCIPALMENTE A MEDIDA QUE O FATOR DE CARGA TENDE PARA 1. ESTES METODOS SAO MUITO BONS E SAO MUITAS VEZES OS ESCOLHIDOS (SALVO SE OS REGISTROS FOREM EXTREMAMENTE PEQUENOS, DE MODO QUE A UTILIZACAO DE ESPACO PARA OS APONTADORES SEJA BEM SIGNIFICATIVA, E SE O FATOR DE CARGA FOR RELATIVAMENTE BAIXO, QUANDO UM OUTRO METODO DE PESQUISA NAO ENCADEADA PODERIA SER MAIS ATRAENTE).

2.- QUANDO O "BUCKET SIZE" E MAIOR QUE 1 (USUALMENTE EM MEMORIA SECUNDARIA), NAO DISPOMOS AINDA DE UM COMPLETO ESTUDO TEORICO DOS METODOS QUE PODEM SER USADOS. APESAR DISSO, AS TABELAS 3.1, 3.2 E 3.3 NOS PERMITEM COMPARAR OS DOIS METODOS MAIS COMUMENTE USADOS EM MEMORIA SECUNDARIA.

O METODO DE PESQUISA LINEAR (E SUAS MELHORES VARIACOES) E MELHOR QUE OS DESCRITOS DE 3.2.2 A 3.2.6, PORQUE O INCREMENTO ISKIP PODE SER ESCOLHIDO DE MODO QUE MINIMIZE O

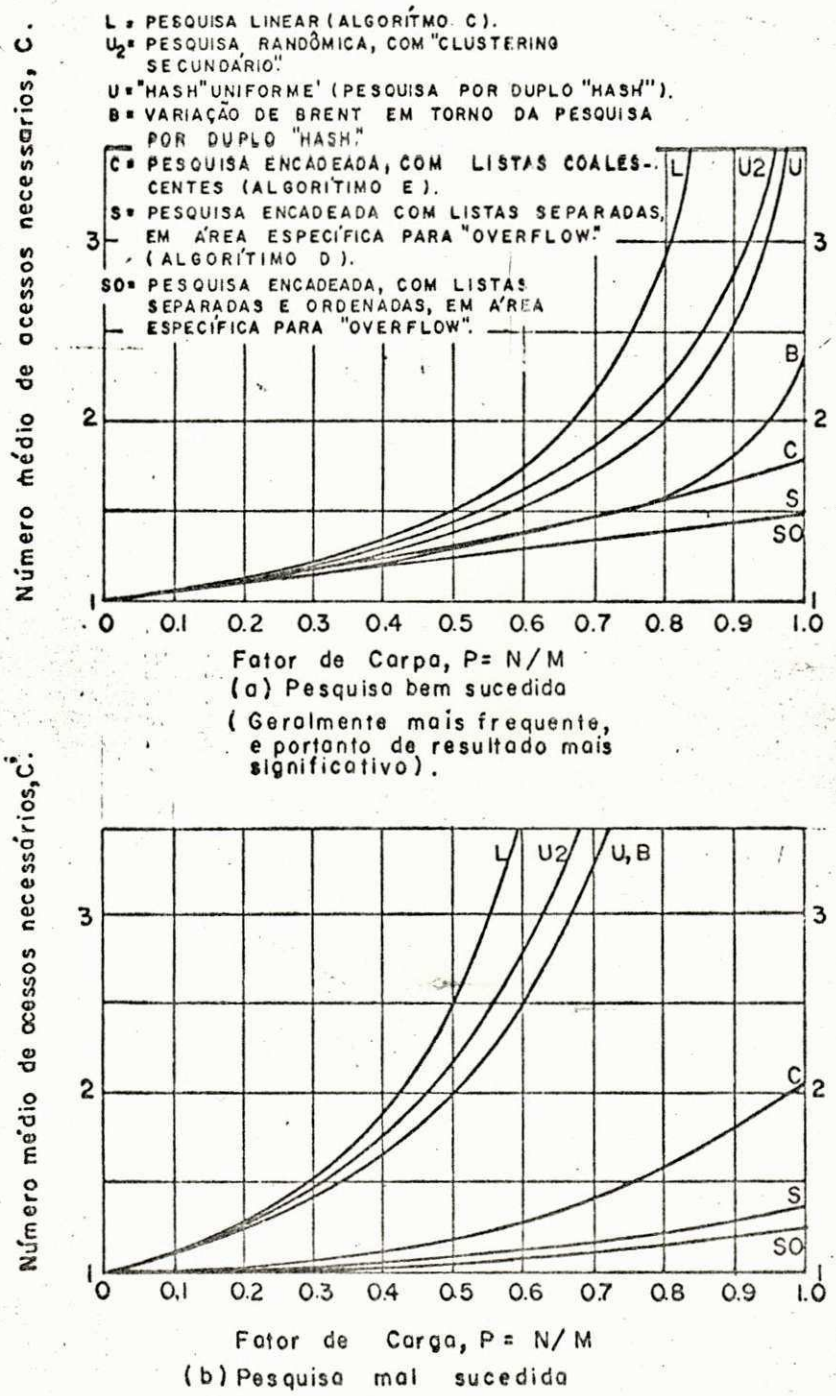


Fig.- 3.7 - Comparação entre os métodos de colisões, quando o "bucket size" é igual a 1: valores limites de número médio de acessos necessários quando M tende para ∞. Figura retirada de Knuth [20].

TEMPO DE LATENCIA ("LATENCY DELAY") ENTRE ACESSOS CONSECUTIVOS. NA REALIDADE, PARA UM "BUCKET SIZE" RAZOAVELMENTE GRANDE (DIGAMOS, B>=20) E FATOR DE CARGA NAO MUITO GRANDE (DIGAMOS, P<=0.85), ESTE METODO PODE EQUIPARAR-SE AO DA PESQUISA ENCADEADA, OU ATE SOBREPUJA-LO, TENDO AINDA A VANTAGEM DE SER BASTANTE SIMPLES. PARA "BUCKET SIZE" PEQUENO (DIGAMOS, B<=10), E ACONSELHAVEL FAZER O "OVERFLOW" NUMA AREA EM SEPARADO. PARA "BUCKET SIZE" GRANDE (DIGAMOS, B>=10), PODE SER MAIS EFICIENTE FAZERMOS O "OVERFLOW" NA AREA PRIMARIA. VER A FIGURA 23.11 DE MARTIN (27), BASEADA NOS ESTUDOS COMPARATIVOS DE LUM, YUEN E DODD (24).

O METODO DE PESQUISA ENCADEADA, COM LISTAS SEPARADAS NUMA AREA ESPECIFICA PARA "OVERFLOW" E CAPACIDADE DO "BUCKET" DE "OVERFLOW" IGUAL A 1, E MELHOR QUE O METODO DA PESQUISA LINEAR, DESDE QUE O "BUCKET SIZE" SEJA RELATIVAMENTE PEQUENO (DIGAMOS, B<20) OU O FATOR DE CARGA SEJA BASTANTE ALTO.

OS METODOS SUGERIDOS EM 3.6.2, ITEM 3, NAO FORAM AINDA ANALISADOS, MAS DEVEM TER UMA PERFORMANCE UM POUCO MELHOR QUE OS ACIMA, AO MENOS QUANTO AO NUMERO DE ACESSOS NECESSARIOS, DESPREZANDO-SE A UTILIZACAO MAIOR OU MENOR DE ESPACO E A EVENTUAL MAIOR COMPLEXIDADE DOS ALGORITMOS. ESTES METODOS PODEM SER USADOS, EMBORA SEJA DISCUTIVEL SE COMPENSA OU NAO TERMOS ESSA MAIOR COMPLEXIDADE E MAIOR UTILIZACAO DE MEMORIA, PARA UMA ECONOMIA TALVEZ INSIGNIFICANTE NO NUMERO DE ACESSOS NECESSARIOS.

3.7 - ANALISE EXPERIMENTAL-COMPARATIVA DOS METODOS DE RESOLUCAO DE "BUCKET OVERFLOWS"

NAO FOI FEITO AINDA UM ESTUDO COMPARATIVO-EXPERIMENTAL SOBRE TODAS AS COMBINACOES POSSIVEIS ENTRE O METODO DE CALCULO DA FUNCAO DE "HASH" E O METODO DE RESOLUCAO DE "BUCKET OVERFLOWS".

APESAR DISSO, LUM, YUEN, DODD & GHOSH (24, 25, 13, E 26) FIZERAM UM EXCELENTE ESTUDO PARCIAL. NA REFERENCIA 24, QUE NAO PODE DEIXAR DE SER CONSULTADA POR QUEM SE INTERESSA PELO ASSUNTO, SAO APRESENTADOS OS RESULTADOS DE UM ESTUDO COMBINANDO 9 METODOS DE CALCULO DA FUNCAO DE "HASH" (METODOS DA DIVISAO, "MID-SQUARE", RANDOMICO, "RADIX", ANALISE DOS DIGITOS, "SHIFTING", "FOLDING", DA DIVISAO POLINOMIAL NO GF (2) E

NO GF (16)) COM 2 METODOS DE RESOLUCAO DOS "BUCKET OVERFLOWS" (METODO DA PESQUISA ENCADEADA COM LISTAS SEPARADAS, AREA DE "OVERFLOW" EM SEPARADO E CAPACIDADE DO "BUCKET" DE "OVERFLOW" IGUAL A 1; E METODO DA PESQUISA LINEAR NA AREA PRIMARIA). SUAS TABELAS DE II A XI APRESENTAM O NUMERO MEDIO DE ACESSOS NECESSARIOS NUMA PESQUISA BEM SUCEDIDA, E O DESVIO PADRAO; SUAS TABELAS DE XII A XXI APRESENTAM A PERCENTAGEM MEDIA DE REGISTROS QUE SOFREPAM "OVERFLOW", NA CRIACAO DO ARQUIVO, E O DESVIO PADRAO; TODAS AS TABELAS APRESENTAM OS RESULTADOS PARA OS 8 METODOS DE CALCULO DA FUNCAO DE "HASH", COMBINADOS COM OS 2 METODOS DE RESOLUCAO DE "BUCKET OVERFLOWS", ISTO PARA FATORES DE CARGA P=0.50, 0.55,..., 0.95, E PARA "BUCKET SIZES" B=1, 2, 5, 10, 20, 50.

QUANTO AO METODO DE CALCULO DA FUNCAO DE "HASH", OBSERVOU-SE QUE, DE UM MODO GENERICO, NA GRANDE MAIORIA DOS CASOS O METODO DA DIVISAO E O MELHOR. VER 2.4.2.

QUANTO AO METODO DE RESOLUCAO DE "BUCKET OVERFLOWS", OBSERVOU-SE QUE, DE UM MODO GENERICO, NA GRANDE MAIORIA DOS CASOS:SE O "BUCKET SIZE" E<=10, O METODO ENCADEADO E MELHOR; SE O "BUCKET SIZE" E>=20, O METODO DE PESQUISA LINEAR E PREFERIVEL; OS CASOS INTERMEDIARIOS RESULTAM NUMA ESCOLHA DIFICIL, NAO GENERALIZAVEL E QUE DEVE SER FEITA APOS CUIDADOSAS PONDERACOES. LEMBRAR QUE O FATOR DE CARGA APRESENTADO E O FATOR DE CARGA PRIMARIO, E QUE , COMO USAMOS AREA DE "OVERFLOW" EM SEPARADO, NECESSITAMOS DE ESPACO ADICIONAL PARA A MESMA, ALEM DE ESPACO PARA OS APONTADORES. ESTAS CONCLUSOES CONCORDAM COM OS ESTUDOS TEORICOS EXISTENTES.

3.8 - "HASHING" USANDO MULTIPLAS CHAVES

JOHNSON (18) EXPLOROU E EXPANDIU A IDEIA DO ENCADEAMENTO , DE MODO A PERMITIR A UTILIZACAO DE UM IDENTIFICADOR SECUNDARIO NA RECUPERACAO DE UM REGISTRO. ISTO SERIA BASTANTE UTIL, POR EXEMPLO, NO CASO DE UM USUARIO DESEJAR TER ACESSO A UM MESMO REGISTRO FISICO DE UM ARQUIVO DE PESSOAL, ORA USANDO COMO CHAVE O NOME DO EMPREGADO, ORA SEU NUMERO DE MATRICULA . A SOLUCAO SUGERIDA CONSISTE EM ARMAZENARMOS O REGISTRO NA POSICAO DETERMINADA PELO "HASHING" SOBRE A CHAVE PRIMARIA, ARMAZENADO-SE EM SEGUIDA UM APONTADOR PARA ESTA POSICAO, NUM CAMPO ESPECIAL DA OUTRA POSICAO DETERMINADA PELO "HASHING" SOBRE A CHAVE SECUNDARIA. OBVIAMENTE, COLISOES E/OU "BUCKET OVERFLOWS" PODEM OCORRER SEJA COM UM OU COM AMBOS OS IDENTIFICADORES. RESOLVEMOS ESSAS COLISOES E/OU "BUCKET OVERFLOWS " USANDO DOIS ENCADEAMENTOS (POR UMA DAS MANEIRAS USUAIS JA

VISTAS) SEPARADOS, UM PARA CADA TIPO DE IDENTIFICADOR (NOME E NUMERO DE MATRICULA). ASSIM, DESPENDEMOS DUAS VEZES MAIS ESPACO, PARA OS APONTADORES, QUE SE SO TIVESSEMOS UM CAMPO COMO IDENTIFICADOR.

A ANALISE DESTE METODO, NO QUE TOCA A PESQUISA DE UM REGISTRO, DADO A SUA CHAVE PRIMARIA, E A MESMA JA VISTA EM 3.6.

A ANALISE DESTE METODO, NO QUE TOCA A PESQUISA DE UM REGISTRO, DADO A SUA CHAVE SECUNDARIA, CONCORDA INTEIRAMENTE COM OS RESULTADOS JA VISTOS EM 3.6, SALVO PELA REFERENCIA INDIRETA AO APONTADOR INICIAL. ASSIM, COM O "BUCKET SIZE" IGUAL A 1 E USANDO-SE O METODO DE PESQUISA COM LISTAS SEPARADAS PARA A RESOLUCAO DOS "BUCKET OVERFLOWS", O NUMERO MEDIO DE ACESSOS NECESSARIOS NUMA PESQUISA BEM SUCEDIDA SERA 1+(1+P/2)= 2+P/2.

3.9 - CONCLUSOES SOBRE A RESOLUCAO DE "BUCKET OVERFLOWS"

1. - QUANTO A FUNCAO DE "HASH", GERALMENTE O MELHOR E USARMOS O "HASH" POR DIVISAO, SEGUNDO AS RECOMENDACOES DE 2.3.1.
2. - A NAO SER QUE ESTEJAMOS USANDO ENCADEAMENTO, OS METODOS DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS" COMECAM RAPIDAMENTE A CAIR DE DESEMPENHO QUANDO O FATOR DE CARGA VAI TENDENDO PARA 1.00 (PRINCIPALMENTE SE O "BUCKET SIZE" FOR 1 OU SO POUCO MAIS QUE 1). NO CASO EXTREMO EM QUE O ARQUIVO ESTA TOTALMENTE CHEIO, EXCETO POR UM "SLOT" LIVRE, E RESOLVEMOS OS "BUCKET OVERFLOWS" PELO METODO DE PESQUISA LINEAR SEM AREA DE "OVERFLOW" EM SEPARADO, UMA PESQUISA BEM SUCEDIDA, SEGUIDA DUMA INSERCAO NO "SLOT" VAZIO, NECESSITARA EM MEDIA M/2 ACESSOS A MEMORIA SECUNDARIA, ONDE M E O NUMERO DE "BUCKETS" DO ARQUIVO. NA PRATICA, NUNCA DEVEREMOS PERMITIR QUE UM ARQUIVO FIQUE TAO CHEIO. MUITAS VEZES ESPECIFICAMOS QUE, QUANDO O FATOR DE CARGA CHEGAR A UM DETERMINADO VALOR (DIGAMOS, 0.70 A 0.80 PARA A PESQUISA LINEAR), O ARQUIVO DEVE TER SUA CAPACIDADE AUMENTADA (DIGAMOS, DUPLICADA), SENDO TODOS OS REGISTROS DO VELHO ARQUIVO REALOCADOS, PELO MESMO ALGORITMO DE "HASH", NO NOVO ARQUIVO.

   SE O ARQUIVO ESTIVER NA MEMORIA SECUNDARIA, A PESQUISA LINEAR (E A VARIACAO PROPOSTA POR KRUTAR) PROVAVELMENTE SERA MELHOR QUE AS DESCRITAS DE 3.2.2 A 3.2.6. REALMENTE, SE O "BUCKET SIZE" FOR RAZOAVELMENTE GRANDE (DI-

GAMOS B> = 20) E O FATOR DE CARGA NÃO FOR MUITO ALTO (DIGAMOS P< = 0.85) ESTE METODO PODE EQUIPARAR-SE A (OU MESMO SOBREPUJAR) OS METODOS DE PESQUISA ENCADEADA.

UM METODO BASTANTE INTERESSANTE DE RECALCULARMOS OS "HASH CODES", QUANDO EXPANDIMOS O TAMANHO DE UMA TABELA DE "HASH", E DADO POR BAYS (3).

3. - QUANDO USAMOS ENCADEAMENTO, A PERFORMANCE DO METODO DE RESOLUCAO DE "BUCKET OVERFLOWS" DETERIORA-SE LENTAMENTE, AO INVES DE RAPIDAMENTE, A MEDIDA QUE O ARQUIVO VAI SE ENCHENDO, DE MODO QUE, USANDO-SE AREA DE "OVERFLOW" EM SEPARADO, O FATOR DE CARGA PRIMARIO PODE MESMO SUPERAR 1.00, SEM GRAVES PROBLEMAS.

USUALMENTE NAO HA VANTAGENS EM TERMOS A CAPACIDADE DO "BUCKET" DE "OVERFLOW" MAIOR QUE 1, SENDO ESTE O METODO QUE DA GERALMENTE O MELHOR RESULTADO GLOBAL, NA PRATICA, QUANDO O ARQUIVO ESTA NA MEMORIA SECUNDARIA (A NAO SER QUE O "BUCKET SIZE" SEJA RAZOAVELMENTE GRANDE E O FATOR DE CARGA NAO MUITO ALTO, QUANDO O SIMPLES METODO DE PESQUISA LINEAR PODERA SER MAIS ATRAENTE).

4. - EM MEMORIA PRINCIPAL, COM "BUCKET SIZE" IGUAL A 1 E FATOR DE CARGA CRESCENDO, OS MELHORES METODOS PARECEM SER GERALMENTE, EM ORDEM:

A) PESQUISA ENCADEADA COM LISTAS SEPARADAS, SE O DISPENDIO COM APONTADORES NAO FOR SIGNIFICATIVO.
B) PESQUISA ENCADEADA COM LISTAS COALESCENTES, SE O DISPENDIO COM APONTADORES NAO FOR SIGNIFICATIVO.
C) PESQUISA LINEAR, VARIACAO PROPOSTA POR KRUTAR (VER 3.2.1).
D) PESQUISA LINEAR, ALGORITMO C DE 3.2.1.
E) PESQUISA POR DUPLO "HASH", OU DE VYSSOTSKY, OU QUADRATICA OU RANDOMICA.

5. - EM MEMORIA SECUNDARIA, COM "BUCKET SIZE" IGUAL A 1 E FATOR DE CARGA CRESCENDO, OS MELHORES METODOS PARECEM SER, EM ORDEM:

A) PESQUISA ENCADEADA, COM LISTAS SEPARADAS (ALGORITMO D DE 3.2.7.1).
B) PESQUISA ENCADEADA, COM LISTAS COALESCENTES (ALGORITMO "E" DE 3.2.7.2).
C) PESQUISA LINEAR, VARIACAO PROPOSTA POR KRUTAR (VER 3.2.1).
D) PESQUISA POR DUPLO "HASH", VARIACAO PROPOSTA POR BRENT (VER 3.2.3).
E) PESQUISA POR DUPLO "HASH" (VER 3.2.2).
F) PESQUISA DE VYSSOTSKY, QUADRATICA OU RANDOMICA (VER 3.2.6, 3.2.5 E 3.2.4).
G) PESQUISA LINEAR (ALGORITMO C DE 3.2.1).

6. - EM MEMORIA SECUNDARIA, COM "BUCKET SIZE" CRESCENDO E

FATOR DE CARGA TAMBEM, OS MELHORES METODOS PARECEM SER , EM ORDEM:

A) PESQUISA LINEAR E SUAS VARIACOES, DESDE QUE O FATOR DE CARGA NAO SEJA MUITO ALTO (DIGAMOS, P < = 0.85) E O "BUCKET SIZE" JA SEJA RAZOAVELMENTE GRANDE (DIGAMOS , B> = 10, OU MESMO B> = 20).

B) PESQUISA ENCADEADA COM LISTAS SEPARADAS E AREA DE "OVERFLOW" EM SEPARADO, TENDO O "BUCKET" DE "OVERFLOW " CAPACIDADE IGUAL A 1 SE O FATOR DE CARGA NAO FOR MUITO ALTO, E CAPACIDADE MAIOR QUE 1 EM CASO CONTRARIO.

C) PESQUISA ENCADEADA COM LISTAS COALESCENTES, TENDO O "BUCKET" DE "OVERFLOW" SUA CAPACIDADE IGUAL A 1 SE O FATOR DE CARGA NAO FOR MUITO ALTO, E MAIOR QUE 1 EM CASO CONTRARIO.

D) OS OUTROS METODOS DE PESQUISA, OU SEJA, POR DUPLO "HASH" (SIMPLES E VARIACAO DE BRENT), PESQUISA QUADRATICA, RANDOMICA E DE VYSSOTSKY, SAO POUCO USADOS, E QUANDO O SAO O "BUCKET SIZE" TAMBEM DEVE SER MAIOR QUE 10 OU 20, E O FATOR DE CARGA DEVE SER NAO MUITO ALTO (DIGAMOS, P < = 0.85), QUANDO OS "BUCKETS" DE "OVERFLOW" PODERAO TER CAPACIDADE IGUAL A 1. EM CASO CONTRARIO, OS "BUCKETS" DE "OVERFLOW" DEVERAO TER CAPACIDADE MAIOR QUE 1.

7. - DE UM MODO GERAL, O USO DE AREA EM SEPARADO PARA "OVERFLOW" TRAZ CERTAS VANTAGENS. COM "BUCKET SIZE" PEQUENO (DIGAMOS, B< = 10) E USANDO-SE PESQUISA LINEAR, A RESOLUCAO DE "OVERFLOWS" NUMA AREA EM SEPARADO E MAIS EFICIENTE.

8. - EM CERTOS CASOS, O USO DE UM DIRETORIO (LISTA DE ENDERECOS) DO ESPACO DISPONIVEL TRAZ ALGUMAS VANTAGENS.

9. - A REALOCACAO PERIODICA DOS SINONIMOS EM ORDEM DECRESCENTE DE FREQUENCIA DE USO SEMPRE TRAZ VANTAGENS NA VELOCIDADE MEDIA DE RECUPERACAO DE UM REGISTRO, QUALQUER QUE SEJA O METODO QUE TENHAMOS ADOTADO PARA CALCULO DO "HASH" E RESOLUCAO DAS COLISOES E/OU "BUCKET OVERFLOWS".

10.- SE OS SINONIMOS FOREM EQUIPROVAVEIS (OU QUASE), A COLOCACAO (E MANUTENCAO) DOS MESMOS NA ORDEM CRESCENTE (OU DECRESCENTE) DAS CHAVES REDUZIRA O NUMERO MEDIO DE ACESSOS NECESSARIOS NUMA PESQUISA MAL SUCEDIDA.

4 - "SCATTER INDEX TABLES", "VIRTUAL SCATTER TABLES" E "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA

4.1 - CONCEITOS GERAIS

4.1.1- DESCRICAO DO METODO:

MORRIS (33), EM 1968, RESSUSCITOU A AREA DE TECNICAS DE ENDERECAMENTO COM UM ARTIGO NOTAVEL. ELE DISCUTIU O "HASHING" INTRODUZINDO O CONCEITO DE UMA "SCATTER INDEX TABLE", OU SIMPLESMENTE "INDEX". ESTE METODO DE ENDERECAMENTO REQUER O USO DE DOIS ARQUIVOS, COMO REPRESENTADO NA FIGURA 4.1. UM DOS ARQUIVOS E UM "DATA FILE", NO QUAL OS REGISTROS SAO ARMAZENADOS SEM LACUNAS, SENDO O ESPACO ALOCADO PARA AS ENTRADAS SOMENTE NA MEDIDA DO NECESSARIO, ENQUANTO AS COLISOES SAO RESOLVIDAS USANDO (PREFERENCIALMENTE) ENCADEAMENTO (*). O OUTRO

---

* COMO ALTERNATIVA, PODERIAM SER USADOS OUTROS METODOS DE RESOLUCAO DOS SINONIMOS. NO ENTANTO, MORRIS CHAMA A ATENCAO DE QUE SO EXCEPCIONALMENTE HA RAZOES PARA PREFERIRMOS OUTRO METODO QUE NAO O DO ENCADEAMENTO (MAS RESSALVAMOS AQUI NOSSAS CONSIDERACOES EM 4.3 SOBRE MAQUINAS COM MEMORIA VIRTUAL PAGINADA). ISTO PORQUE, O ESPACO SENDO ALOCADO SOMENTE A MEDIDA QUE NECESSARIO E PROVINDO DE UMA AREA DE ARMAZENAMENTO LIVRE (ESPACO DISPONIVEL), O METODO DE ENDERECAMENTO E TAO FACIL DE PROGRAMAR QUANTO OS OUTROS E E MAIS EFICIENTE. ADEMAIS, A DELECAO DE UM REGISTRO TORNA-SE MAIS FACIL. - UM REGISTRO A SER DELETADO E SIMPLESMENTE RETIRADO DA CADEIA EM QUE ESTA E DEVOLVIDO AO ESPACO DISPONIVEL.

ARQUIVO E UMA "SCATTER INDEX TABLE" (TAMBEM CHAMADO DE INDEX, INDICE OU DICIONARIO DE CONSULTAS), NA QUAL A FUNCAO "HASH" "MAPEIA" (FAZ SUA APLICACAO). ASSIM, ENQUANTO OS DADOS PROPRIAMENTE DITOS FICAM NO "DATA FILE", A "SCATTER INDEX TABLE" CONSISTE APENAS DE APONTADORES PARA OS REGISTROS DO "DATA FILE". CHAMAMOS ESSA TABELA DE "SCATTER INDEX TABLE", OU INDEX, JUSTAMENTE PARA ENFATIZAR QUE ELA NAO CONTEM OS REGISTROS DE DADOS PROPRIAMENTE DITOS, MAS APENAS APONTADORES PARA OS MESMOS.

- EXEMPLO: A FIGURA 4.1 APRESENTA UMA "SCATTER INDEX TABLE" SENDO APLICADA COMO UMA ALTERNATIVA PARA ARMAZENAMENTO DOS REGISTROS DAS FIGURAS 3.2 E 3.1. EMBORA OS 15 "DATA RECORDS" CONTINUEM AINDA SENDO IDENTIFICADOS PELO NUMERO DE MATRICULA, AGORA ELES ESTAO ARMAZENADOS NOS REGISTROS R00 ATE R14, SEM LACUNAS E ORDENADOS SEGUNDO O CAMPO NOME (SUPONDO-SE QUE DE ALGUM MODO FORAM INSERIDOS ASSIM). O ARQUIVO FOI COMO QUE "AMPLIADO" PELA "SCATTER INDEX TABLE" CONSISTINDO DE 30 APONTADORES, COM ENDERECOS VARIANDO DE P00 ATE P29. ASSIM, NA FIGURA 4.1 O FATOR DE CARGA E 15/30 = 0.5, MELHOR PORTANTO QUE O DA FIGURA 3.2, QUE ERA 15/18 = 0.83, E QUE O DA FIGURA 3.1, QUE ERA 15/20 = 0.75. SE OS "BUCKET SIZES" DAS FIGURAS 3.2 E 3.1 FOSSEM IGUAIS A 1, TERIAMOS EM AMBOS 6 COLISOES, ENQUANTO AGORA SO TEMOS 1 COLISAO. DEVEMOS SUPOR, POR SIMPLICIDADE, QUE TODOS OS APONTADORES NA "SCATTER INDEX TABLE" SAO AI COLOCADOS SOMENTE APOS O "DATA RECORD" TER SIDO INSERIDO NO ARQUIVO. PARA LOCALIZAR O REGISTRO DO EMPREGADO 31 (DARIO) SEU NUMERO DE MATRICULA E TRANSFORMADO NO ENDERECO P01, ONDE UM APONTADOR PARA R02 ESTA ARMAZENADO. COMO O IDENTIFICADOR DO REGISTRO R02 E 61≠31, ENTAO O CAMPO DE APONTADOR DE SINONIMOS E UTILIZADO PARA TERMOS ACESSO AO REGISTRO R01, CUJO IDENTIFICADOR E 31, COINCIDINDO COM AQUELE QUE PROCURAVAMOS, TENDO ASSIM O EMPREGADO 31 SIDO LOCALIZADO NO REGISTRO R01. SE QUISESSEMOS INSERIR UM NOVO REGISTRO "WATTS63ENGN", SEU NUMERO DE MATRICULA SERIA TRANSFORMADO NO ENDERECO P03, DE CONTEUDO AINDA "EM BRANCO", LOGO O CONTEUDO DE R15, QUE E O PROXIMO REGISTRO UTILIZAVEL, DEVERIA TORNAR-SE "WATTS63ENGN----", PARA ENTAO O CONTEUDO DE P03 TORNAR-SE "R15". MAS SE, AO INVES DISSO, TIVESSEMOS QUERIDO INSERIR COMO NOVO REGISTRO "ZARUR 91 TECN", OBTERIAMOS, A PARTIR DO SEU NUMERO DE MATRICULA, O ENDERECO P01, CUJO CONTEUDO APONTA PARA R02, E DE IDENTIFICADOR 61≠91, COM SINONIMO R01, DE IDENTIFICADOR 31≠91 E SEM MAIS SINONIMOS COMO SUCESSORES; ASSIM, O CONTEUDO DE R15 PASSARIA A SER "ZARUR 91 TECN R02", ENQUANTO QUE O DE P01 SERIA R15. A DELECAO DE UM REGISTRO E TRIVIAL, BASTANDO QUE SEJA REMOVIDO DA CADEIA DE SINONIMOS EM QUE ESTA E SEJA DEVOLVIDO AO ESPACO DISPONIVEL (NO CASO DO REGISTRO A SER DELETADO CONSTITUIR O PRIMEIRO OU O UNICO DA LISTA DE SINONIMOS, A "SCATTER INDEX TABLE" DEVE SER ALTERADA).

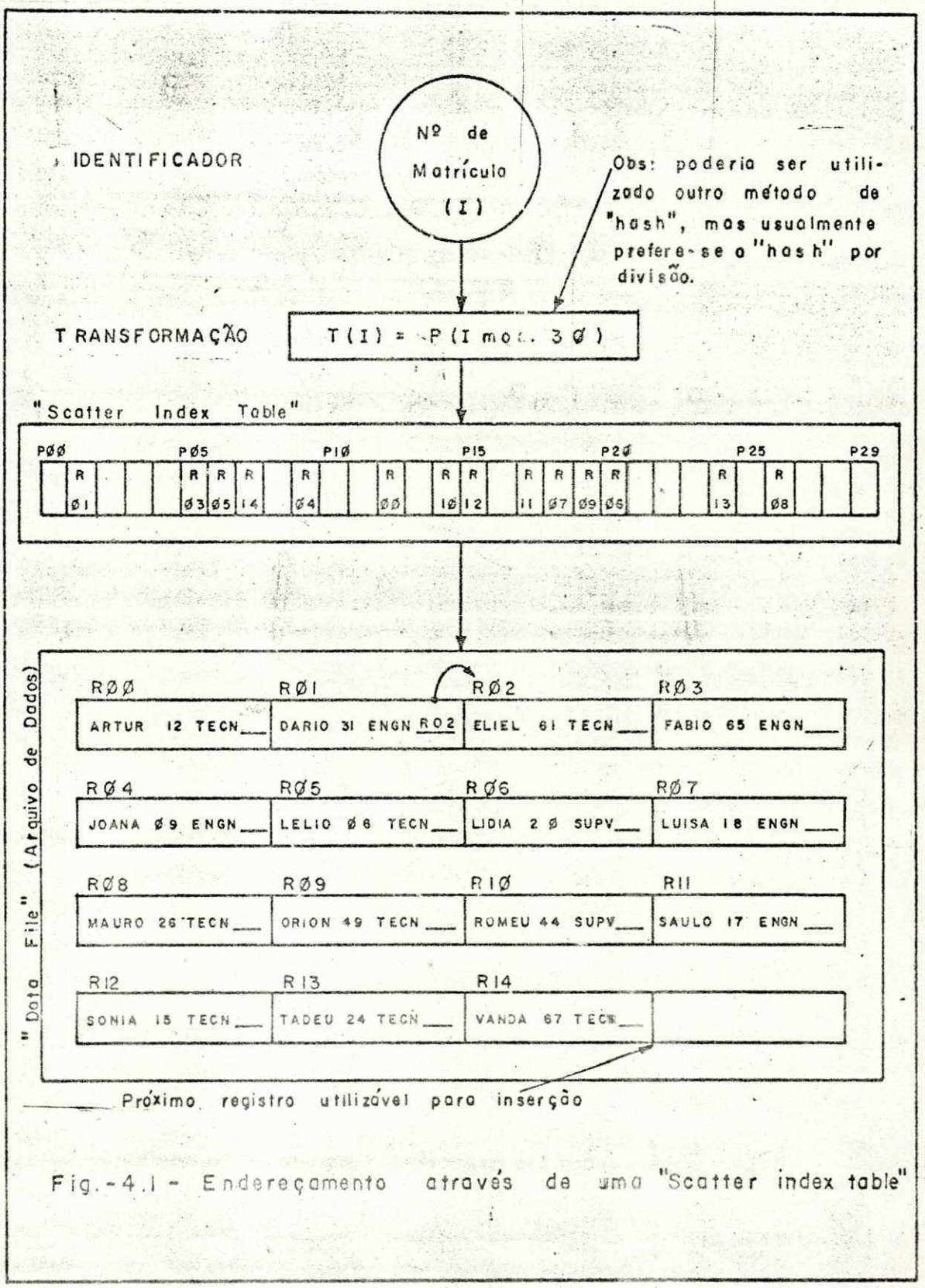


Fig.-4.1 - Endereçamento através de uma "Scatter index table"

4.1.2- INDEPENDENCIA ENTRE LOCALIZACAO FISICA DO REGISTRO E SEU IDENTIFICADOR:

E OBVIO QUE SO POR NOSSA ESCOLHA TOMAMOS O ARQUIVO ORDENADO PELOS NOMES, O QUE NAO E NECESSARIO E ATE, PARA UM CONJUNTO DE DADOS MUITO VOLATIL, PODE IMPLICAR NUMA IMPRATICAVEL QUANTIDADE DE ATUALIZACOES; ASSIM, EM ESSENCIA, ESTE EXEMPLO ILUSTRA APENAS UMA CARACTERISTICA MUITA VEZES VALIOSA DO METODO DE MORRIS: A LOCALIZACAO FISICA DE UM REGISTRO E INDEPENDENTE DO CORRESPONDENTE ENDERECO DADO PELA FUNCAO "HASH", E ASSIM DO SEU IDENTIFICADOR. ESTA PROPRIEDADE SERA A BASE DA 3A E 4A VANTAGENS ADIANTE APONTADAS.

4.1.3- VANTAGENS DO USO DA "SCATTER INDEX TABLE":

COMO FOI MUITO BEM EXPLICADO POR SEVERANCE (43), O USO DE UMA "SCATTER INDEX TABLE" TEM APARENTEMENTE, SOB UM EXAME SUPERFICIAL, DUAS DESVANTAGENS:

1. O ESPACO ADICIONAL QUE E EXIGIDO PELA TABELA;
2. O ACESSO ADICIONAL A TABELA, QUE SERA SEMPRE NECESSARIO PARA TERMOS ACESSO AO ARQUIVO.

NO ENTANTO, COM UM ESTUDO MAIS APROFUNDADO, VEREMOS QUE NAO EXISTEM NECESSARIAMENTE ESSAS DESVANTAGENS, EM MUITOS CASOS PRATICOS. EM GRANDES ARQUIVOS, COMO AQUELES ARQUIVOS ENCONTRADOS NOS "DATA BASES" COMERCIAIS COMUNS, QUE TIPICAMENTE TEM MILHARES DE REGISTROS, CADA UM COM CENTENAS DE "BYTES" DE COMPRIMENTO, O USO DA "SCATTER INDEX TABLE" TEM 5 PRINCIPAIS VANTAGENS:

1A VANTAGEM: REDUCAO NA UTILIZACAO DE MEMORIA:

USEMOS O EXEMPLO DE SEVERANCE (43). SUPONHAMOS INICIALMENTE QUE UM CONJUNTO DE N=9,000 REGISTROS E ARMAZENADO SEM UMA "SCATTER INDEX TABLE", COM UM FATOR DE CARGA P=0.9, USANDO UMA FUNCAO DE "HASHING" UNIFORME E "OVERFLOW" ENCADEADO, SEM AREA ESPECIFICA PARA "OVERFLOW". SUPONHAMOS TAMBEM QUE CADA "RECORD SLOT" TEM UM COMPRIMENTO DE L=500 "BYTES", INCLUINDO UM CAMPO PARA APONTADOR DE SINONIMOS, COM O TAMANHO DE 2 "BYTES". ENTAO O ESPACO TOTAL REQUERIDO E:

N * L / P = 9,000 * 500 / 0.9 = 5,000,000 "BYTES".

SUPONHAMOS AGORA, AO CONTRARIO, QUE OS REGISTROS FORAM ARMAZENADOS USANDO UMA "SCATTER INDEX TABLE". SUPONHAMOS MESMO, PARA UM MELHOR DESEMPENHO CONTRA AS COLISOES (E DESDE QUE A PENALIDADE EM ESPACO POR UMA ENTRADA DE "HASH" INATIVA E DE APENAS 2 "BYTES", BEM MENOR QUE 500), QUE O FATOR DE CARGA FOI REDUZIDA DE 0.9 PARA, POR EXMPLO, 0.6. O ESPACO REQUERIDO PELA "SCATTER INDEX TABLE" SERA N* (COMPRIMENTO DO APONTADOR)/P = 9,000 * 2/0.6 = 30,000. NO ENTANTO, DESDE QUE O TA-

MANHO PARA O ARQUIVO FOI AGORA REDUZIDO PARA N* L=9,000 * 500 = 4,500,000, O ESPACO TOTAL REQUERIDO SERA 30,000 + 4,500,000 = 4,530,000. ASSIM, O RESULTADO LIQUIDO SERA UMA ECONOMIA DE 5,000,000 - 4,530,000 = 470,000 "BYTES" ≈ 10% ... DESTE MODO, PARA ARQUIVOS COM UM COMPRIMENTO DE REGISTROS RELATIVAMENTE GRANDE, O USO DUMA "SCATTER INDEX TABLE" USUALMENTE IMPLICARA EM ECONOMIA NA UTILIZACAO DE ESPACO NA MEMORIA SECUNDARIA.

2A VANTAGEM: REDUCAO DO TEMPO NECESSARIO PARA ACESSO A UM REGISTRO:

USEMOS AINDA O MESMO EXEMPLO ACIMA, LEMBRANDO QUE, QUANDO NAO USAMOS A "SCATTER INDEX TABLE", O NUMERO MEDIO DE ACESSOS PARA RECUPERAR UM REGISTRO E DADO POR 1 + P/2 = 1 + 0.9/2 = 1.45.

USEMOS AGORA A "SCATTER INDEX TABLE"; O NUMERO MEDIO DE ACESSOS A MEMORIA SECUNDARIA PARECE APARENTEMENTE TER QUE CRESCER PARA 1 + (1 + P/2) = 1 + (1 + 0.6/2) = 2.30, DEVIDO A REFERENCIA INICIAL A TABELA. NO ENTANTO, SE A ATIVIDADE DE RECUPERACOES E ALTA, PODEMOS CONSIDERAR A RELATIVAMENTE PEQUENA "SCATTER INDEX TABLE" PERMANENTEMENTE NA MEMORIA PRINCIPAL. ORA, UMA VEZ QUE O ACESSO A ESSA TABELA, APOS CARREGADA, E DESPREZIVEL, TEMOS PARA PESAR APENAS 2.30 - 1 = 1.30 ACESSOS A MEMORIA SECUNDARIA, DE MODO QUE O RESULTADO LIQUIDO (DEPENDENDO DO POSICIONAMENTO FISICO DOS SINONIMOS), PODERA VIR A SER UMA REDUCAO LIQUIDA NO TEMPO DE RECUPERACAO DE UMA INFORMACAO, DE UM FATOR NA ORDEM DE (1.45 - 1.30)/1.45 = 0.15/1.45 ≈ 10%. DESTE MODO, DESDE QUE A RELATIVAMENTE PEQUENA "SCATTER INDEX TABLE" FIQUE NA MEMORIA PRINCIPAL, TEREMOS USUALMENTE UMA SENSIVEL REDUCAO NO TEMPO NECESSARIO PARA ACESSO A UM REGISTRO DO ARQUIVO.

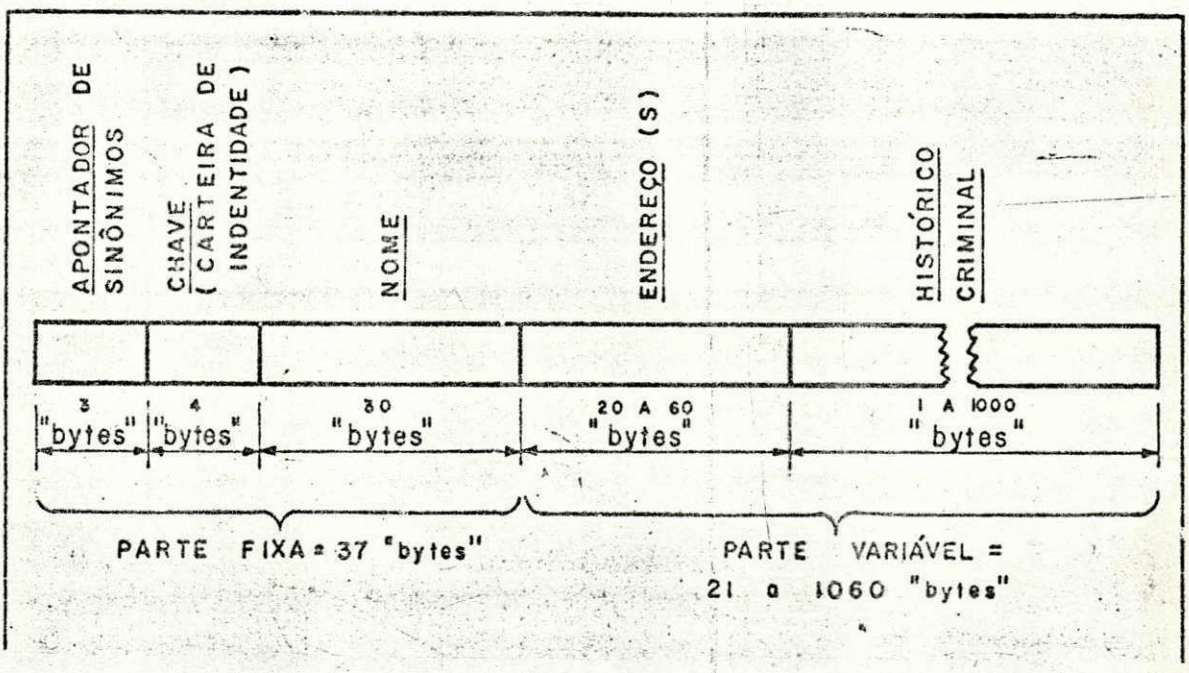
3A VANTAGEM: FACILIDADE PARA LIDAR COM REGISTROS DE TAMANHO VARIAVEL:

COMO JA SE DEVE TER PERCEBIDO, TODOS OS OUTROS ESQUEMAS DE "HASH" JA VISTOS NOS OUTROS CAPITULOS ANTERIORES, PRESTAM-SE DIRETAMENTE APENAS PARA ARQUIVOS COM REGISTROS DE TAMANHO FIXO, DE MODO QUE A SOLUCAO IMEDIATA NESSES ESQUEMAS ANTERIORES E DIMENSIONAR OS REGISTROS DE ACORDO COM O MAXIMO DE POSICOES QUE ELES POSSAM VIR A EXIGIR. ASSIM, SE O MAXIMO COMPRIMENTO PREVISIVEL DE UM REGISTRO FOR DE 500 "BYTES", MESMO QUE O COMPRIMENTO MEDIO PREVISTO SEJA DE APENAS 100 "BYTES", ESSA SOLUCAO SIMPLISTA, DIMENSIONANDO TODOS OS REGISTROS COM 500 "BYTES", IMPLICARA NUM DESPERDICIO MEDIO DE 400 "BYTES" POR REGISTRO, OU SEJA, 80% DO SEU COMPRIMENTO

EVIDENTEMENTE UMA SOLUCAO DESSE TIPO, NA MAIORIA DAS VEZES, E INDESEJAVEL. NUMA SITUACAO DESSAS, DESTACAMOS COMO UMA DAS SOLUCOES MAIS FACEIS E EFICIENTES O USO DUMA "SCATTER INDEX TABLE" QUE CONTENHA NAO SO APONTADORES PARA O INICIO DO REGISTRO, COMO TAMBEM OUTROS TIPOS DE INFORMACOES (VER MAX-

WELL & SEVERANCE (31)), QUE PERMITAM DETERMINAR A LOCALIZACAO DE CADA CAMPO NO REGISTRO.

COMO EXEMPLO ILUSTRATIVO, CONSIDEREMOS UM ARQUIVO PARA 12,000 REGISTROS, QUE PODERIA SER USADO PELA POLICIA DE UMA CIDADE, COM OS REGISTROS ASSIM FORMADOS:



USAREMOS AGORA UMA "SCATTER INDEX TABLE" EM FORMA DE MATRIZ, COM 3 LINHAS E 16,387 COLUNAS, A 1A LINHA INDICANDO A POSICAO DO INICIO DO REGISTRO E A 2A E 3A INDICANDO RESPECTIVAMENTE OS COMPRIMENTOS DOS CAMPOS "ENDERECO" E "HISTORICO CRIMINAL"; ASSIM, SE FORMOS ARMAZENAR, SEM OCORRENCIA DE SINONIMOS, UM REGISTRO LOGICO COM CHAVE 34,256, COM OS ENDERECOS NECESSITANDO 50 "BYTES" E O HISTORICO CRIMINAL 120 "BYTES", TEREMOS A SITUACAO MOSTRADA NA FIGURA 4.2; USANDO-SE ESTE ESQUEMA, O REGISTRO OCUPOU APENAS 207 "BYTES", ENQUANTO QUE UMA SOLUCAÓ COM REGISTRO DE TAMANHO FIXO EXIGIRIA PARA O MESMO 1,097 "BYTES", DOS QUAIS 890 SERIAM DESPERDICADOS!...

COMO SE VE, O PROBLEMA DE ARMAZENAMENTO DE REGISTROS DE COMPRIMENTO VARIAVEL FICA RESOLVIDO DE UMA MANEIRA MUITO EFICIENTE PELO USO DE UMA "SCATTER INDEX TABLE", E JA QUE ESTA FICA NA MEMORIA PRINCIPAL, ISTO IMPLICARA NUM ACRESCIMO DE TEMPO INSIGNIFICANTE.

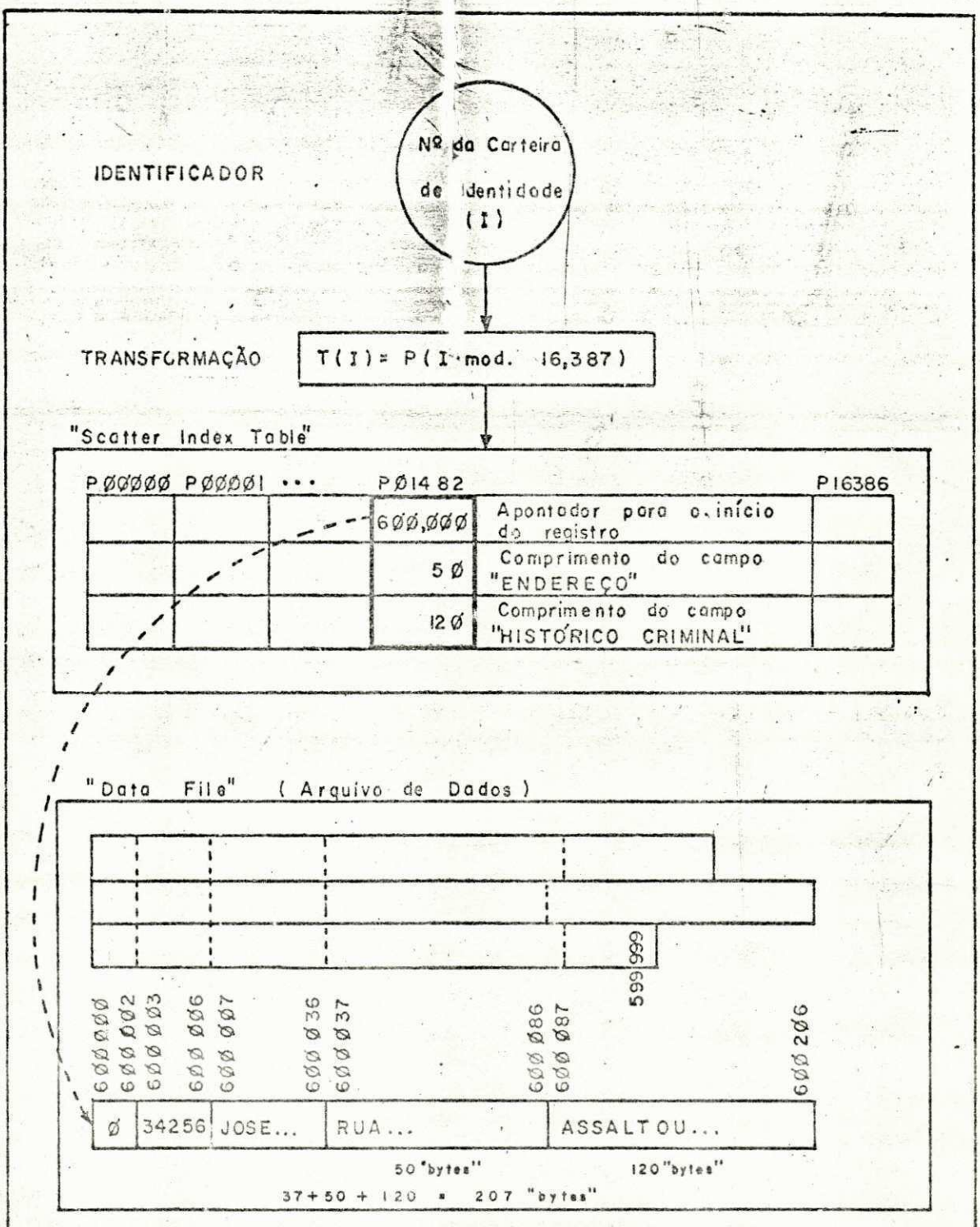


Fig. - 4.2 - Exemplo do uso duma "scatter index table" para registros de tamanho variável.

4A VANTAGEM: FACILIDADE PARA SE USAR MAIS QUE UM CAMPO COMO IDENTIFICADOR:

EM ALGUMAS APLICACOES, E ALTAMENTE DESEJAVEL TERMOS UM ACESSO RAPIDO E EFICIENTE A UM REGISTRO DO ARQUIVO, ORA UTILIZANDO-SE UM CAMPO COMO CHAVE, ORA OUTRO CAMPO. ESTE PROBLEMA DE ACESSO A UM REGISTRO PELO USO DE MULTIPLOS CAMPOS COMO CHAVE ("MULTIKEY FILE") SO PODIA SER RESOLVIDO PELO ESQUEMA DE "HASH" VISTO EM 3.8 (SEM MENCIONARMOS UMA MANEIRA SIMPLORIA E INEFICIENTE, QUASE SEMPRE INTEIRAMENTE INACEITAVEL: CRIAR TANTOS ARQUIVOS EM DUPLICATA QUANTOS CAMPOS QUEREMOS USAR COMO CHAVE E FAZER O "HASH" INDEPENDENTEMENTE SOBRE CADA UM DELES, SEGUNDO SUA CHAVE).

NO ENTANTO, SUPONHAMOS O MESMO EXEMPLO DA FIGURA 4.1, E QUE AGORA QUEREMOS TER ACESSO A UM REGISTRO QUER USANDO O CAMPO "NO. DE MATRICULA", QUER USANDO O CAMPO "NOME" COMO IDENTIFICADOR. SUPONHAMOS QUE AS REPRESENTACOES INTERNAS DOS CARACTERES ALFABETICOS USADOS SAO A=11; B=12; C=13; D=14; E=15; F=16; G=17; H=18; I=19; J=20; K=21; L=22; M=23; N=24; O=25; P=26; Q=27; R=28; S=29; T=30; U=31; V=32; W=33; X=34; Y=35; Z=36; B(BRANCO)=37. DEVEREMOS USAR DUAS TRANSFORMACOES DIFERENTES:

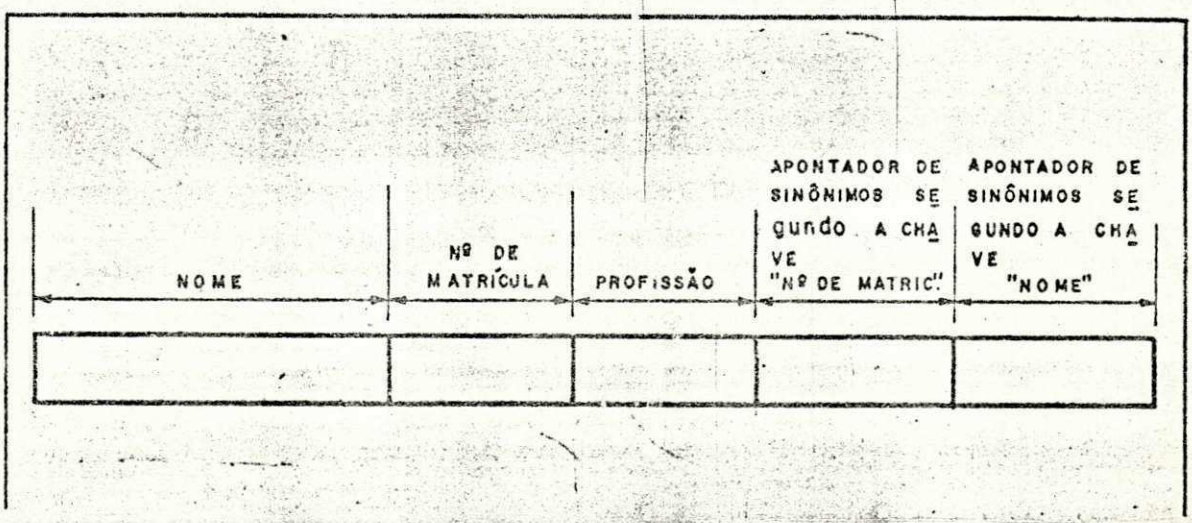
A) SE A CHAVE FOR O "NO. DE MATRICULA" (I):
T(I) = P(I MOD. 30);

B) SE A CHAVE FOR "NOME" (I) :
T(I) = Q(VALOR ABSOLUTO DA SOMA DAS REPRESENTACOES INTERNAS DO CAMPO "NOME", MOD. 30).

EXEMPLOS:

T("ARTUR") = Q((11+28+30+31+28) MOD. 30) = Q08
T("JOANA") = Q((20+25+11+24+11) MOD. 30) = Q01
T("ORION") = Q((25+28+19+25+24) MOD. 30) = Q01

O REGISTRO DEVERA AGORA TER 5 CAMPOS:



CONTINUAREMOS USANDO UM UNICO ARQUIVO DE DADOS, POREM AGORA DUAS "SCATTER INDEX TABLES", CONFORME A FIGURA 4.3, UMA PARA UM TIPO DE CHAVE E A OUTRA PARA O OUTRO TIPO. NA INSERCAO, COLOCAMOS O REGISTRO NA PRIMEIRA POSICAO DISPONIVEL E DEPOIS FAZEMOS AS INDICACOES NECESSARIAS EM AMBAS AS CADEIAS DE SINONIMOS QUE COMECAM PELAS POSICOES NAS "SCATTER INDEX TABLES" REFERENCIADAS PELOS "HASHES" FEITOS TANTO SOBRE UM CAMPO COMO CHAVE, COMO TAMBEM SOBRE O OUTRO. O ACESSO, SEGUNDO QUALQUER DAS CHAVES, SERA ASSIM POSSIVEL. NA DELECAO, DEVEMOS TOMAR CUIDADOS PARA DELETAR O REGISTRO DE AMBAS AS CADEIAS DE SINONIMOS A QUE O REGISTRO PERTENCE.

E FACIL DE SE VER QUE ESTE PROCEDIMENTO PODERIA SER GENERALIZADO PARA 3 OU MAIS CAMPOS SERVINDO COMO CHAVE, SE NECESSARIO E CONVENIENTE. ASSIM, O EXEMPLO ILUSTRA COMO O USO DE "SCATTER INDEX TABLES" POSSIBILITA O USO DE DIFERENTES CAMPOS COMO CHAVE. EM ALGUNS CASOS, QUANDO TRABALHAMOS EM TEMPO REAL, ESQUEMAS COMO O SUGERIDO PODEM VIR A SER MUITO VANTAJOSOS.

5A VANTAGEM: REDUCAO DO NUMERO DE COLISOES:

ISTO JA ESTAVA IMPLICITO E INTER-RELACIONADO COM A APRESENTACAO DA 1A E 2A VANTAGENS, MAS RESOLVEMOS EXPLICITAR MELHOR: QUANDO O COMPRIMENTO DOS REGISTROS DO ARQUIVO E MUITO MAIOR (USUALMENTE NA FAIXA DE 50 A 500 VEZES) QUE O COMPRIMENTO DOS REGISTROS DA(S) "SCATTER INDEX TABLE(S)" NECESSARIA(S), PODEMOS MUITAS VEZES TOMAR O FATOR DE CARGA BEM BAIXO (NA ORDEM DE 0.2 A 0.6) PARA A(S) "SCATTER INDEX TABLE(S)", E AINDA ASSIM SER POSSIVEL TANTO ECONOMIA NA UTILIZACAO TOTAL DE MEMORIA COMO TAMBEM QUE A(S) "SCATTER INDEX TABLE(S)" CAIBA (M) COMPLETAMENTE NA MEMORIA INTERNA. ESTA REDUCAO NO FATOR DE CARGA (QUE, NOS ESQUEMAS TRADICIONAIS DE "HASH", E DA ORDEM DE 0.7 A 0.9) IMPLICARA NUMA SENSIVEL REDUCAO DO NUMERO DE COLISOES. E E ISTO, EM VERDADE, QUE, ALIADO AO FATO DA(S)

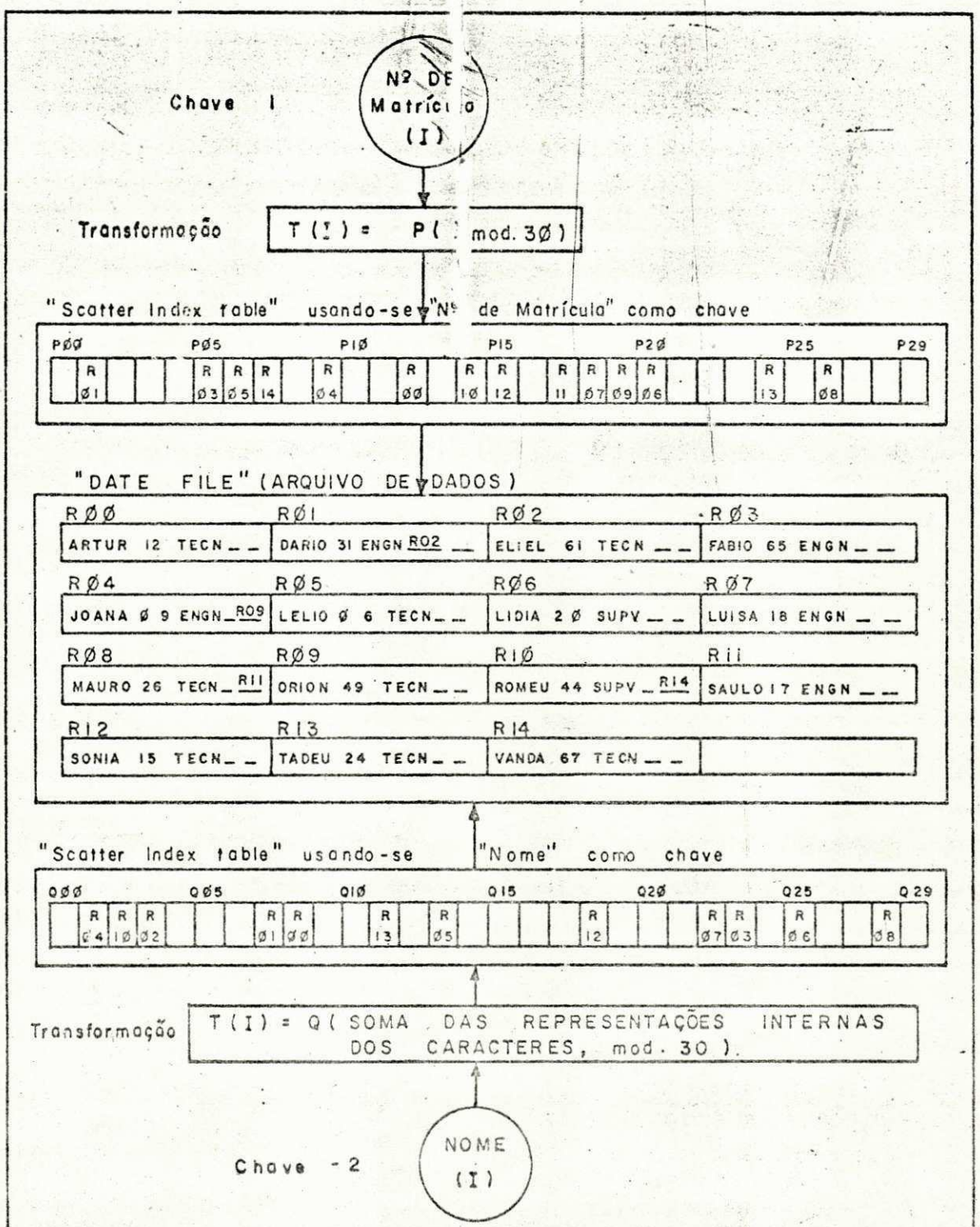


Fig. 4.3 - Exemplos do uso de múltiplas "scatter index tables" para acesso a um registro pelo uso de múltiplos campos como chave.

"SCATTER INDEX TABLE(S)" ESTIVAR(EM) NA MEMORIA INTERNA, IMPLICARA NUMA REDUCAO DO TEMPO NECESSARIO PARA A PESQUISA.

4.1.4- ALGORITMO G:
PESQUISA(ACESSO) A UM REGISTRO, USANDO UMA "SCATTER INDEX TABLE":

ARGUMENTO DE ENTRADA: E A CHAVE (KEY) QUE QUEREMOS PESQUISAR.
ARGUMENTOS DE RETORNO:
A) A VARIAVEL LOGICA FOUND, QUE SE TORNARA ".TRUE." SE A PESQUISA TIVER SUCESSO, E ".FALSE." EM CASO CONTRARIO;
B) A VARIAVEL VALUE, QUE ASSUMIRA UM VALOR ASSOCIADO A CHAVE, NO CASO DA PESQUISA TER SUCESSO.

O "DATA FILE" E CONSIDERADO EM FORMA MATRICIAL, CADA LINHA CONSTITUINDO UM REGISTRO COM 3 POSICOES; A 1A POSICAO DE CADA REGISTRO E UM APONTADOR DE SINONIMOS (NULO, SE NAO EXISTIR NENHUM SINONIMO); A 2A POSICAO ARMAZENA A CHAVE E A 3A O VALOR QUE LHE E ASSOCIADO.

SUPOMOS QUE OS REGISTROS FISICOS SAO ALOCADOS PROVINDO DE UMA LISTA DE ESPACO DISPONIVEL ("FREE STORAGE LIST"), E SOMENTE QUANDO NECESSARIO, VOLTANDO A ESTA LISTA QUANDO SE TORNAREM DISPENSAVEIS.

O TAMANHO DA "SCATTER INDEX TABLE", ISTO E, O SEU NUMERO DE "SLOTS", E SUPOSTO SER UM PRIMO MAIOR QUE O MAXIMO NUMERO POSSIVEL DE REGISTROS A SER USADO NO ARQUIVO DE DADOS.

SIGNIFICADO DOS SIMBOLOS:

KEY = CHAVE PARA A PRESENTE CHAMADA DO ALGORITMO;
VALUE = VALOR ASSOCIADO A CHAVE ENCONTRADA;
FOUND = ".TRUE." SE CHAVE FOI ENCONTRADA;
= ".FALSE." SE A CHAVE NAO FOI ENCONTRADA;
FIRST = VARIAVEL LOGICA QUE SO E "TRUE." ANTES DA PRIMEIRA CHAMADA DESTE ALGORITMO;
KEYSAVE = CHAVE USADA NA ULTIMA CHAMADA DO ALGORITMO; E UMA VARIAVEL AUXILIAR USADA PARA TESTAR A OCORRENCIA DE FALHAS OU ENGANOS, NAS INSERCOES E DELECOES;
IHASH = ENDERECO OBTIDO POR QUALQUER METODO DE "HASH" APLICADO SOBRE A CHAVE CORRENTE, EMBORA USEMOS AQUI O METODO DA DIVISAO;
INDEX = "SCATTER INDEX TABLE";
INDEXSIZE = NO. DE REGISTROS (DE 1 POSICAO) EM INDEX;
KPLACE = INDEX CORRENTE NA "SCATTER INDEX TABLE";
ISPACE = LISTA DE ESPACO DISPONIVEL;

```
ISPACESIZE= NO. DE REGISTROS EM ISPACE;
NEXTFREE  = APONTADOR PARA O PROXIMO REGISTRO DISPONIVEL EM
--------    ISPACE;
ACTUAL    = APONTADOR PARA O REGISTRO CORRENTE EM ISPACE;
------
PRECEDENT = REGISTRO IMEDIATAMENTE ANTECESSOR DO REGISTRO
---------   CORRENTE (ACTUAL), NA CADEIA DE QUE O MESMO FAZ
            PARTE;
          = 0 SE NAO EXISTIR ESTE ANTECESSOR.
```

OBSERVACAO: ALEM DOS 3 ARGUMENTOS JA MENCIONADOS (KEY, VALUE, FOUND), AS VARIAVEIS GRIFADAS SERAO NECESSARIAS PARA QUE ESTE ALGORITMO TRABALHE EM CONJUNTO COM OS DE INSERCAO E DELECAO, PODENDO ASSIM, EM FORTRAN IV NIVEL E, FICAREM NA AREA COMMON. EM FORTRAN IV NIVEIS G OU H, OS TRES ALGORITMOS DE 4.1.4, 4.1.5 E 4.1.6 PODEM CONSTITUIR UMA SO SUBROTINA, COM COM DUAS "ENTRADAS" PARA AS PARTES CORRESPONDENTES AOS ALGORITMOS H E I, ENCONTRADOS EM 4.1.5 E 4.1.6.

G.0 - (CONDICOES PREVIAS):
UMA UNICA VEZ EM TODA A EXISTENCIA DO ARQUIVO DE DADOS, OU, MAIS PRECISAMENTE, QUANDO DO SEU "NASCIMENTO", DEFINIMOS:
FIRST = ".TRUE.",
E OS VALORES ISPACESIZE E INDEXSIZE.
(EM FORTRAN, ISTO PODE SER FEITO POR LEITURA OU USANDO O COMANDO DATA).

G.1 - (INICIALIZACAO):
SE FIRST = ".FALSE." VA PARA G.2;
SE NAO: INDEX (I) = 0 PARA 1 <= I <= INDEXSIZE,
ISPACE (I,1) = I-1 PARA 1 <= I <= ISPACESIZE,
NEXTFREE = ISPACESIZE,
FIRST = ".FALSE.";

G.2 - ("HASH"):
KEYSAVE = KEY,
IHASH = (IABS (KEY)) MOD. INDEXSIZE,
KPLACE = IHASH + 1;

G.3 - (EXAME DE INDEX):
PRECEDENT =0,
ACTUAL = INDEX (KPLACE);
SE ACTUAL = 0, ENTAO FOUND = ".FALSE."; FIM.

G.4 - (SEGUE A CADEIA DE SINONIMOS, PROCURANDO A CHAVE):
SE ISPACE (ACTUAL, 2) = KEY, ENTAO FOUND = ".TRUE." E
VALUE = ISPACE (ACTUAL, 3); FIM.
SE NAO: PRECEDENT = ACTUAL,
ACTUAL = ISPACE (ACTUAL, 1);
SE ACTUAL = 0, ENTAO FOUND = ".FALSE."; FIM.
SE NAO, REPITA G.4.

4.1.5 - ALGORITMO H:

INSERCAO DE UM REGISTRO CUJA CHAVE JA FOI PESQUISADA E NAO ENCONTRADA, OU FOI ENCONTRADA E QUEREMOS MODIFICAR O VALOR ASSOCIADO:

ARGUMENTOS DE ENTRADA: KEY E VALUE.

```
H.1 - (TESTA SE E A CHAVE APROPRIADA E SE A CHAVE JA ESTA
      IMPLANTADA OU NAO):
      SE KEY ≠ KEYSAVE ENTAO "ERRO"; FIM.
      SE ACTUAL = 0 VA PARA H.2.
      SE ISPACE (ACTUAL, 2) = KEYSAVE VA PARA H.3.
H.2 - (ALOCA ESPACO PARA NOVA ENTRADA):
      SE NEXTFREE = 0, ENTAO "OVERFLOW DO ESPACO DISPONIVEL";
                     FIM.
      SE NAO:        ACTUAL = NEXTFREE,
                     NEXTFREE = ISPACE (NEXTFREE, 1),
                     ISPACE (ACTUAL, 1) = INDEX (KPLACE),
                     INDEX (KPLACE) = ACTUAL;
H.3 - (INSERCAO):
      ISPACE (ACTUAL, 2) = KEYSAVE,
      ISPACE (ACTUAL, 3) = VALUE; FIM.
```

- A FIGURA 4.4 ILUSTRA O FUNCIONAMENTO DESTE ALGORITMO E DO ANTERIOR.

4.1.6 - ALGORITMO I:
DELECAO DE UM REGISTRO CUJA CHAVE JA FOI PESQUISADA E LOCALIZADA:

- ARGUMENTO DE ENTRADA: KEY

```
I.1 - (TESTA SE E A CHAVE APROPRIADA):
      SE KEY ≠ KEYSAVE ENTAO "ERRO"; FIM.
      SE KEY ≠ ISPACE (ACTUAL, 2) ENTAO "ERRO"; FIM.
I.2 - (QUEBRA A CADEIA E DEVOLVE O REGISTRO AO ESPACO DISPO-
      NIVEL):
      SE PRECEDENT = 0 FACA INDEX(KPLACE) = ISPACE(ACTUAL,1);
      SE NAO, FACA ISPACE(PRECEDENT) = ISPACE(ACTUAL,1);VA PARA I.3
I.3 - EM QUALQUER DOS CASOS, FACA
            ISPACE (ACTUAL,1) = NEXTFREE,
            NEXTFREE = ACTUAL E FIM.
```

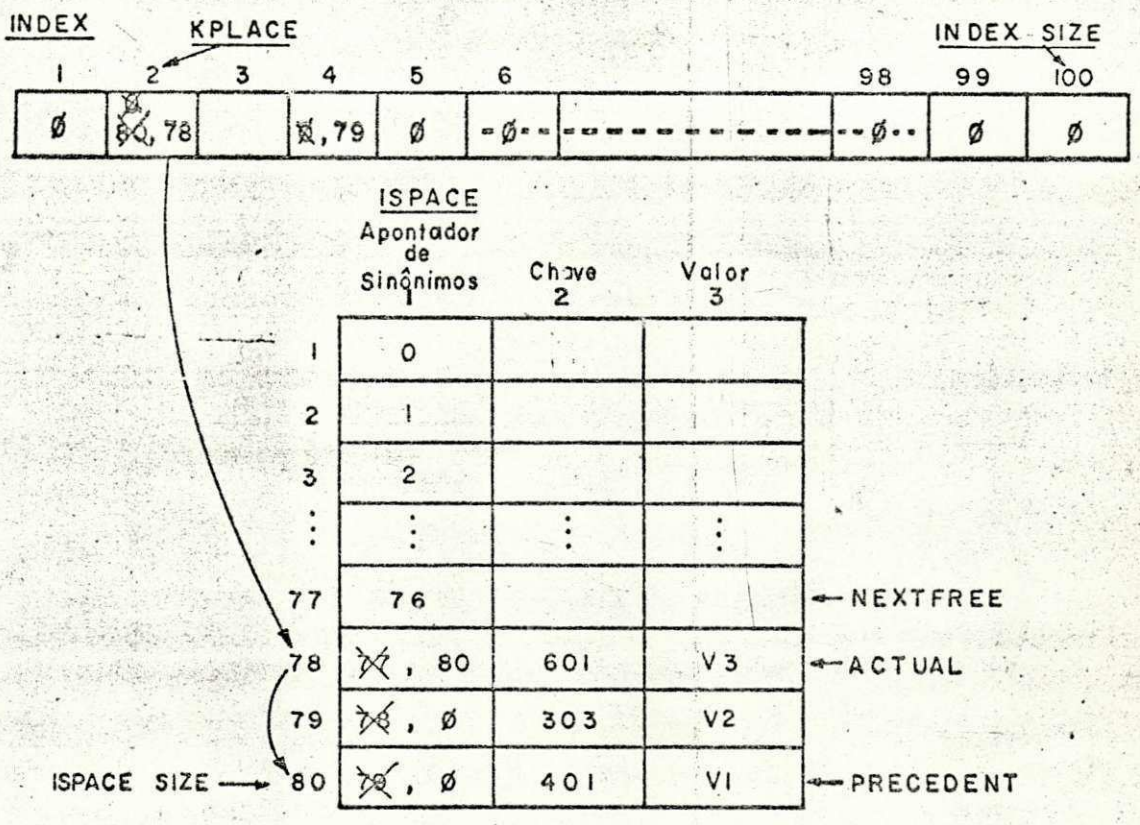


Fig.- 4.4 - Ilustração da situação que teremos utilizando os algorítimos de acesso e inserção usando uma "scatter index table" e logo após a inserção de (401, VI), (303, V2) e (601, V3) como as 3 primeiras entradas a ocorrerem.

4.1.7 - PROGRAMAS EM FORTRAM IV, NIVEL H:

NO APENDICE B DE MORRIS (33) ENCONTRAMOS UM OTIMO PROGRAMA PARA ACESSO A UM REGISTRO, INSERCAO OU DELECAO USANDO UMA "SCATTER INDEX TABLE", QUE NECESSITA APENAS LIGEIRAS MODIFICACOES PARA ADAPTAR-SE PERFEITAMENTE AOS ALGORITMOS QUE APRESENTAMOS. ESSAS MODIFICACOES DIZEM RESPEITO PRINCIPALMENTE AO METODO DE "HASH" UTILIZADO E AO FATO DE MORRIS TER USADO UM ARQUIVO "LINEAR" ENQUANTO NOS PREFERIMOS UM "MATRICAL".

4.2 - "VIRTUAL SCATTER TABLES"

4.2.1 - IDEIA GERAL:

MORRIS (33), CONSIDERANDO AGORA UMA SITUACAO EM QUE O "HASH" E FEITO DIRETAMENTE SOBRE A TABELA QUE CONTEM OS DADOS ("SCATTER TABLE"), SEM FAZER USO PORTANTO DA "SCATTER INDEX TABLE", NOTOU QUE O TEMPO NECESSARIO PARA PESQUISAR UM REGISTRO NA TABELA DEPENDE NAO SOMENTE DO NUMERO DE TENTATIVAS NECESSARIAS PARA ENCONTRAR ESTE REGISTRO, COMO TAMBEM DO TEMPO NECESSARIO PARA EFETUAR CADA COMPARACAO ENTRE AS CHAVES. SE AS CHAVES SAO BASTANTE COMPLEXAS (POR EXEMPLO, SE ELAS SAO CADEIAS DE CARACTERES RAZOAVELMENTE GRANDES E ADEMAIS DE COMPRIMENTO VARIAVEL), PODE FAZER-SE NECESSARIO UM CONSIDERAVEL ESPACO DE TEMPO PARA DETERMINAR SE DUAS CHAVES SAO OU NAO IDENTICAS.

EM ALGUMAS APLICACOES, QUANDO NOS DEPARARMOS COM O PROBLEMA DESCRITO NO PARAGRAFO ANTERIOR, UM ATRAENTE METODO PARA DIMINUIR O TEMPO NECESSARIO PARA EFETIVACAO DE UMA PROVA E O SEGUINTE:

A) CALCULAR CONVENIENTEMENTE UM "HASH ADDRESS" PARA UMA "SCATTER TABLE" MUITO MAIOR DO QUE A QUE ESTAMOS REALMENTE USANDO (HA RAZOES DE ORDEM PRATICA PARA SEMPRE TOMARMOS COMO UMA POTENCIA DE 2 TANTO O TAMANHO DA "SCATTER TABLE" REAL COMO O DA "SCATTER TABLE" VIRTUAL, QUE E BEM MAIOR QUE O DA REAL, CORRESPONDENDO A AMPLITUDE TOTAL DA FUNCAO "HASH" QUE DEVE TER SIDO ESCOLHIDA APROPRIADAMENTE PARA ISTO);

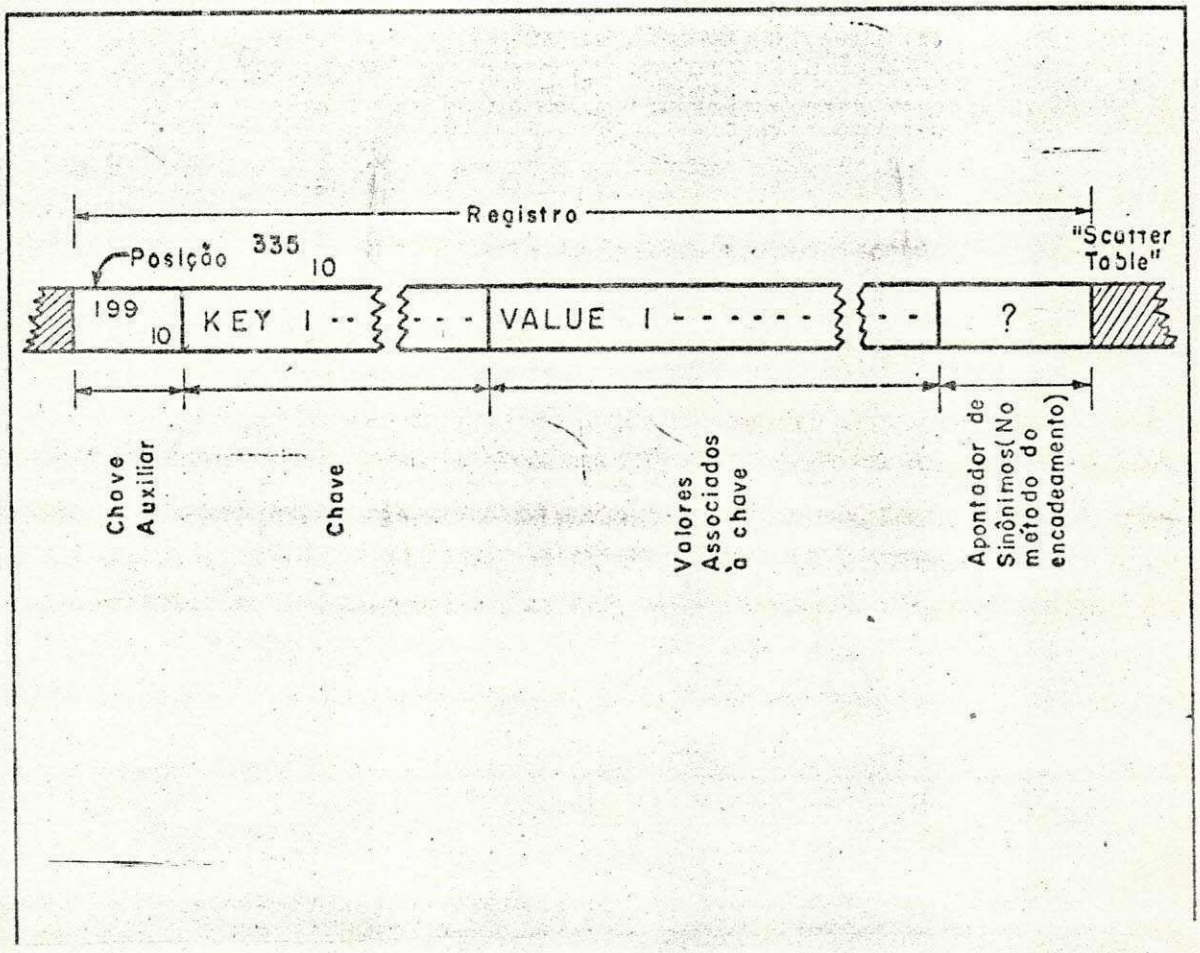
B) USAR UMA PARTE DESSE "HASH ADDRESS" (NORMALMENTE OS "BITS" MAIS A DIREITA QUE SE FIZEREM NECESSARIOS) PARA REALMENTE REFERENCIAR A "SCATTER TABLE";

C) ARMAZENAR A OUTRA PARTE DESSE "HASH ADDRESS" (NORMALMENTE OS "BITS" RESTANTES A ESQUERDA DAQUELES QUE FORAM TOMADOS PARA ENDERECAR A "SCATTER TABLE"), JUNTAMENTE COM A CHAVE E OS DEMAIS DADOS, NO REGISTRO APROPRIADO DA TABELA ("SCATTER TABLE"). VEREMOS QUE ESTAS PARTES RESTANTES DOS "HASH ADDRESSES", ARMAZENADOS NOS REGISTROS CORRESPONDENTES , FUNCIONARAO TAL E QUAL UMA CHAVE AUXILIAR OU SECUNDARIA.

EXEMPLO ILUSTRATIVO:

SUPONHAMOS QUE TEMOS UMA "SCATTER TABLE" DE 2**10 = 1024

POSICOES E ESTAMOS CALCULANDO OS "HASH ADDRESSES" COM 20 "BITS"; FORNECIDA UMA CHAVE KEY1, SUPONHAMOS QUE SEU "HASH ADDRESS" CORRESPONDENTE SEJA 0011000111 0101001111; ENTAO OS 10 "BITS" MAIS A DIREITA SAO USADOS PARA REFERENCIAREM A POSICAO 0101001111(2) = 335(10), ENQUANTO OS 10 "BITS" MAIS A ESQUERDA, OU SEJA, 0011000111(2)=199(10), SERAO ARMAZENADOS NO REGISTRO CUJA POSICAO E 335(10), JUNTAMENTE COM A CHAVE KEY1 , OS DEMAIS VALORES A ELA ASSOCIADOS (VALUE1) E O APONTADOR DE SINONIMOS (SE ESTIVERMOS RESOLVENDO AS COLISOES POR ENCADEAMENTO):



QUANDO QUISERMOS PESQUISAR UM REGISTRO, DADA A SUA CHAVE , COMPUTAMOS SEU "HASH ADDRESS" COMO CONVENCIONADO E SEGUIMOS QUALQUER DOS METODOS DE RESOLUCAO DE COLISOES QUE TENHAMOS

ESCOLHIDO; NO ENTANTO, AO INVES DE COMPARARMOS LOGO AS CHAVES, PRIMEIRAMENTE COMPARANDO OS "BITS" EXTRAS DO "HASH ADDRESS", AGORA CALCULADO, COM A CHAVE AUXILIAR, (OU SEJA, OS "BITS" EXTRAS ARMAZENADOS) DE CADA REGISTRO NA CADEIA DE SINONIMOS, COMPARACAO ESTA BEM MAIS FACIL; SOMENTE SE HOUVER COINCIDENCIA E QUE AS CHAVES PROPRIAMENTE DITAS NECESSITARAO SER COMPARADAS.

4.2.2 - APLICACOES DE "VIRTUAL SCATTER TABLES":

1A APLICACAO:

PERCEBE-SE FACILMENTE QUE A POSSIBILIDADE DE DUAS CHAVES DIFERENTES TEREM O MESMO "HASH ADDRESS" E OS MESMOS "BITS" EXTRA DO "HASH" (CHAVE AUXILIAR) E PRECISAMENTE A MESMA QUE ELAS TERIAM DE COLIDIR NA VIRTUAL "SCATTER TABLE" MUITO MAIOR. ORA, NO EXEMPLO ANTERIOR, O TAMANHO REAL DA "SCATTER TABLE" E 2**10 = 1024 POSICOES E O TAMANHO VIRTUAL E 2**20 = 1,048,576 POSICOES; E SE TODOS OS "SLOTS" DA "SCATTER TABLE" REAL ESTIVEREM OCUPADOS, O FATOR DE CARGA REAL SERA P = 1 , ENQUANTO QUE O VIRTUAL SERA P = 2**10/2**20=2**(-10) = 0.001. NESTE CASO, O NUMERO MEDIO ESPERADO DE EXAMES ENTRE AS CHAVES PROPRIAMENTE DITAS, PARA LOCALIZAR UMA CHAVE DADA, SERA APENAS E = 1 + P / 2 = 1 + 2**(- 11) = 1.0005 ... (PARA FATORES DE CARGA EXTREMAMENTE PEQUENOS, SUFICIENTEMENTE PROXIMOS DE ZERO, ESTA FORMULA E VALIDA PARA TODOS OS METODOS DE RESOLUCAO DE COLISOES QUE POSSAM SER ADOTADOS).

EM ALGUNS ASPECTOS, PORTANTO, SE PUDERMOS DESPREZAR O TEMPO DE EXAME DA CHAVE AUXILIAR FRENTE AO TEMPO DE PROVA DA CHAVE PROPRIAMENTE DITA, A TABELA PODE TER UM DESEMPENHO TAL E QUAL COMO SE FOSSE REALMENTE DO TAMANHO BEM MAIOR DA "VIRTUAL SCATTER TABLE" COM O BAIXISSIMO FATOR DE CARGA VIRTUAL CORRESPONDENTE.

2A APLICACAO:

O USO DE UMA VIRTUAL "SCATTER TABLE" PERMITE QUE ELA SEJA AUMENTADA EM TAMANHO, DURANTE A EXECUCAO, SEM TERMOS A NECESSIDADE DE FAZER NOVA E TRABALHOSAMENTE OUTRO "HASH" SOBRE TODAS AS CHAVES, COMO ACONTECERIA COM OS ESQUEMAS DE "HASH" ANTERIORES A ESTE CAPITULO. POR EXEMPLO, PARA SE DOBRAR O TAMANHO DE UMA "SCATTER TABLE", UMA UNICA VARREDURA E FEITA ATRAVES DA TABELA ANTIGA E AS ENTRADAS SAO REALOCADAS NA METADE INFERIOR OU SUPERIOR DA NOVA TABELA DEPENDENDO APENAS DO SEU "BIT" EXTRA (ACRESCENTADO NA POSICAO MAIS A ESQUERDA DO "HASH ADDRESS") SER Ø OU 1. SE AS COLISOES NAO ERAM RESOLVIDAS POR ENCADEAMENTO, ENTAO OS "BITS" ORIGINAIS NECESSITAM SER GUARDADOS EM ADICAO AOS "BITS" EXTRA (CHAVE AUXILIAR), PARA QUE

POSSAMOS DETERMINAR CORRETAMENTE EM QUE ENDERECO DEVE UMA ENTRADA SER RE-ALOCADA. SE AS COLISOES ERAM RESOLVIDAS POR ENCADEAMENTO, ENTAO ISTO E DESNECESSARIO. OBSERVE O EXEMPLO ABAIXO (FIGURA 4.5).

NATURALMENTE, SE ESTIVERMOS USANDO A "SCATTER TABLE" COMO UM INDEX (TENDO-SE ENTAO UMA "SCATTER INDEX TABLE") OS "DATA RECORDS" NAO PRECISAM SER MOVIDOS, MAS APENAS OS APONTADORES.

3A APLICACAO:

SUPONHAMOS QUE O "HASH ADDRESS" POSSA SER CALCULADO COM UMA ORDEM DE MAGNITUDE MUITISSIMO MAIOR QUE A QUE ESTAMOS REALMENTE USANDO PARA ENDERECAR A TABELA (POR EXEMPLO, SUPONHAMOS QUE NECESSITAMOS DE APENAS 20 "BITS" PARA ENDERECAR A "SCATTER TABLE" E PODEMOS CALCULAR O "HASH ADDRESS" COM 64 "BITS"). A PROBABILIDADE DE QUE DUAS CHAVES DISTINTAS TENHAM OS MESMOS "VIRTUAL HASH ADDRESSES" TORNA-SE TAO PEQUENA (NO EXEMPLO 2**(20-64)=2**(-44)=2.5*10**(-13)) QUE AS CHAVES PODEM DEIXAR COMPLETAMENTE DE SEREM EXAMINADAS, SEM QUALQUER INCONVENIENTE DE ORDEM PRATICA. SE UMA CHAVE TEM O MESMO .... "VIRTUAL HASH ADDRESS" DE UMA ENTRADA JA EXISTENTE, ENTAO PODEMOS SUPOR OU ASSUMIR , PARA FINS PRATICOS, QUE ELAS SAO A MESMA CHAVE. ASSIM, NATURALMENTE, NAO HAVERA NECESSIDADE ALGUMA DE ARMAZENARMOS AS CHAVES NOS REGISTROS, SALVO SE ELAS FOREM NECESSARIAS PARA ALGUM OUTRO PROPOSITO. TIPICAMENTE , PODEM SE PASSAR ANOS E ANOS SEM QUE ENCONTREMOS DUAS CHAVES DIFERENTES NO MESMO PROGRAMA E COM O MESMO "VIRTUAL HASH ADDRESS". NO ENTANTO LEMBRAMOS QUE E OBVIO, EM TODO ESTE ESTUDO, QUE DEVEREMOS ESTAR BEM CERTOS DE QUE OS "HASH ADDRESSES" ESTAO DISTRIBUIDOS DE UMA MANEIRA ACEITAVELMENTE UNIFORME SOBRE OS ENDERECOS VIRTUAIS DISPONIVEIS.

ESTA 3A APLICACAO DO USO DUMA "VIRTUAL SCATTER TABLE" GUARDA CERTA SEMELHANCA COM UMA SOLUCAO PROPOSTA POR BUTTER LAMPSON, VISANDO COMPENSAR O DISPENDIO COM APONTADORES NUMA "SCATTER TABLE" CONVENCIONAL (NAO "VIRTUAL") POR UMA ECONOMIA NA UTILIZACAO DE ESPACO PARA AS CHAVES, ISTO SUPONDO-SE A ADOCAO DO METODO DE PESQUISA ENCADEADA, COM LISTAS SEPARADAS , PARA A RESOLUCAO DAS COLISOES E OU "BUCKET OVERFLOWS" E QUE NAO FORAM ENCONTRADAS POR "SHIFTING" OU "FOLDING". O EXERCICIO 13 DA SECCAO 6.4 DE KNUTH (20), PAGINAS 543 E 544 EXPLICA EM QUE CONSISTE A SOLUCAO PROPOSTA: SEJA T1(I) A FUNCAO DE "HASH", E T2(I) UMA FUNCAO TAL QUE I POSSA SER DETERMINADO A PARTIR DE T1(I) E T2(I); POR EXEMPLO, NUM "HASHING" POR DIVISAO NOS PODEMOS TOMAR T1(I) = I MOD M, E T2(I) = I/M ARREDONDADO PARA MENOS; DEMONSTRA-SE MATEMATICAMENTE QUE, QUANDO O ENCADEAMENTO E USADO SEM SUPERPOSICAO DAS LISTAS, NECESSITAMOS ARMAZENAR APENAS T2(I) E NAO I EM CADA REGISTRO; ESTA ECONOMIA DE ESPACO COMPENSA, PARCIAL OU TOTALMENTE, A UTILIZACAO DO MESMO COM OS APONTADORES.

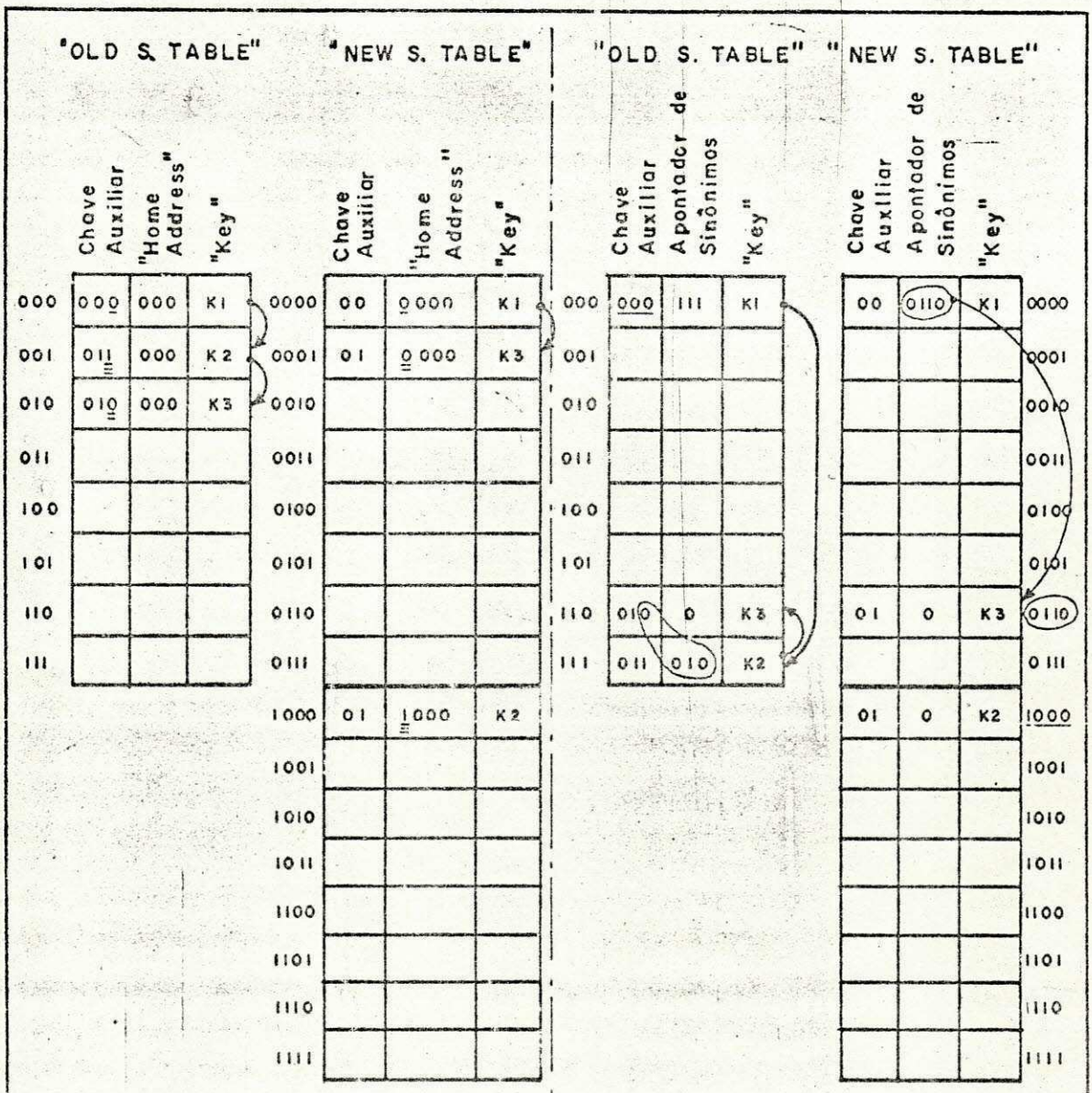


Fig. – 4.5. – Exemplo do uso de uma "virtual scatter table" com a duplicação do seu tamanho. Supomos que as chaves K1, K2, e K3 apareceram nesta ordem, e que os "hash addresses" correspondentes são: 000000, 011000, 010000.

4.2.3 - PESQUISA, INSERCAO E DELECAO NUMA "VIRTUAL SCATTER TABLE":

"MUTATIS MUTANDIS", TUDO O QUE DISSEMOS NO CAPITULO 3 APLICA-SE TAMBEM A UMA "VIRTUAL SCATTER TABLE".

4.3 - "SCATTER TABLES" EM MAQUINAS COM MEMORIA VIRTUAL PAGINADA

4.3.1 - INTRODUCAO:

O ADVENTO DE MAQUINAS COM MEMORIA VIRTUAL VEIO DAR UMA DIMENSAO ADICIONAL AOS METODOS DE "HASH", E MORRIS (33) TEVE ISTO EM MENTE AO DISCUTIR O USO DE "SCATTER TABLES" OU "SCATTER INDEX TABLES" EM MAQUINAS DESSE TIPO.

RELEMBREMOS AQUI O QUE SE ENTENDE POR MAQUINA COM MEMORIA VIRTUAL PAGINADA (OU COM PAGINACAO POR DEMANDA): NA MAIOR PARTE DAS MODERNAS E PODEROSAS MAQUINAS UM PROGRAMA PODE ENDERECAR UM ESPACO BEM MAIOR DE MEMORIA DO QUE ESTA VERDADEIRAMENTE DISPONIVEL PARA ESSE PROGRAMA NA MEMORIA INTERNA DO COMPUTADOR. QUANDO UM REGISTRO E REFERENCIADO PELO PROGRAMA E NAO ESTA JA NA MEMORIA PRINCIPAL, O BLOCO DE DADOS (CHAMADO PAGINA) QUE CONTEM O REGISTRO NA MEMORIA SECUNDARIA DEVE SER TRAZIDO AUTOMATICAMENTE PARA A MEMORIA PRINCIPAL PELO SISTEMA OPERACIONAL, OCUPANDO AI O ESPACO DE UM OUTRO BLOCO QUE, SOB CERTOS CRITERIOS, POSSA SER CONSIDERADO A ESSA ALTURA O MAIS DISPENSAVEL. NA MAIORIA DOS SISTEMAS OPERACIONAIS ESTE JULGAMENTO E FEITO COM BASE NA FREQUENCIA DE USO DA PAGINA NUM CERTO ULTIMO INTERVALO DE TEMPO, SENDO O BLOCO MAIS DISPENSAVEL AQUELE DE MENOR FREQUENCIA DE USO. EXECUCOES POSTERIORES DENTRO DO PROGRAMA TEM QUE SER RETARDADAS ATE QUE A OPERACAO DE TRANSFERENCIA DA PAGINA PARA A MEMORIA PRINCIPAL, SEJA COMPLETADA. O TAMANHO DAS PAGINAS FICA USALMENTE NA FAIXA DE 2**6 (=64) A 2**12 (=4096) PALAVRAS, SENDO O TAMANHO MAIS FREQUENTEMENTE USADO O DE 2**10 = 1024 PALAVRAS. PARA MAIORES DETALHES SOBRE MEMORIA VIRTUAL EM GERAL, CONSULTAR, ENTRE OUTROS, DENNING (9).

EM UMA MAQUINA COM MEMORIA VIRTUAL, NO ENTANTO, PODE OCORRER QUE UMA "SCATTER TABLE" TENHA SIDO DEFINIDA DE MODO QUE O

SEU TAMANHO EXCEDA A QUANTIDADE DE MEMORIA PRINCIPAL DISPONIVEL PARA O PROGRAMA, OCASIONANDO QUE CADA NOVO ACESSO A ..... "SCATTER TABLE" POSSA IMPLICAR NA OCORRENCIA DE UMA TRANSFERENCIA DE PAGINA DA MEMORIA SECUNDARIA PARA A PRINCIPAL, O QUE RESULTARIA EM UM DESEMPENHO INEFICIENTE.

PARA CONSEGUIRMOS MELHOR EFICIENCIA EM CASOS DESSE TIPO, DEVEMOS ESCOLHER MEIOS DE ACESSO AOS REGISTROS QUE NOS ASSEGUREM QUE REFERENCIAS CONSECUTIVAS A MEMORIA (NUMA CADEIA DE SINONIMOS) INCIDAM TANTO QUANTO POSSIVEL NUMA MESMA PAGINA OU EM PAGINAS QUE PROVAVELMENTE JA ESTEJAM NA MEMORIA PRINCIPAL. E FACILMENTE PERCEPTIVEL QUE TERIAMOS CONSIDERAVEL DESPERDICIO DE TEMPO SE DEFINISSIMOS UMA "SCATTER TABLE" NUMA MAQUINA COM PAGINACAO POR DEMANDA E ENTAO RESOLVESSEMOS AS COLISOES QUER POR ENCADEAMENTO, PESQUISA QUADRATICA, DE VYSSOTSKY, POR DUPLO "HASH" OU RANDOMICA, LIVREMENTE ATRAVES DE TODA MEMORIA. TODOS ESSES METODOS RESULTARIAM NO RISCO DE TRAZERMOS UMA NOVA PAGINA PARA A MEMORIA PRINCIPAL EM CADA TENTATIVA DE LOCALIZACAO DUMA CHAVE NA SUA CADEIA DE SINONIMOS.

ESTUDEMOS AGORA AS DIVERSAS SOLUCOES POSSIVEIS PARA MELHORIA DA EFICIENCIA ATRAVES DA MINIMIZACAO DO NUMERO DE TRANSFERENCIAS DA MEMORIA SECUNDARIA PARA A PRINCIPAL, QUANDO DA PESQUISA DE UMA CHAVE QUALQUER, NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, USANDO-SE O "HASHING".

4.3.2 - SOLUCAO SEM USO DE "INDEX" E SEM USO DE VIRTUALIDADE NA "SCATTER TABLE":

A FUNCAO "HASH" MAPEIA DIRETAMENTE NA "SCATTER TABLE", COM UM FATOR DE CARGA SUFICIENTEMENTE MENOR QUE 1, PARA GARANTIR UMA EFICIENCIA RAZOAVEL. GENERICAMENTE FALANDO, ESTA SOLUCAO PRESTA-SE MELHOR PARA OS CASOS EM QUE OS REGISTROS SAO RELATIVAMENTE BEM PEQUENOS, QUANDO UM BOM NUMERO DE REGISTROS CABERIA NUMA SO PAGINA E O USO DE UMA "SCATTER INDEX TABLE" NAO ECONOMIZARIA MEMORIA OU NAO TRARIA OUTRAS VANTAGENS, AO MENOS SIGNIFICATIVAMENTE. TEMOS 4 ALTERNATIVAS, QUE SERAO APRESENTADAS NA ORDEM CRESCENTE DA EFICIENCIA EM TEMPO QUE APRESENTAM.

A 1A ALTERNATIVA, SUGERIDA POR MORRIS (33), CONSISTE NO USO DO "OVERFLOW" ABERTO (PESQUISA LINEAR) NA RESOLUCAO DE SINONIMOS, SEM EMPREGO DE CIRCULARIDADE DE ENDERECOS DENTRO DAS PAGINAS. MORRIS OBSERVOU QUE UMA OU DUAS DUZIAS DE PESQUISAS NUMA MESMA PAGINA TEM UM CUSTO MENOR QUE DUAS PESQUISAS EM PAGINAS DISTINTAS E QUE ESTE FATOR PODE PESAR FORTEMENTE EM FAVOR DO USO DO "OVERFLOW" ABERTO, PORQUE ENTAO SERA BASTANTE MAIS PROVAVEL QUE EXAMES CONSECUTIVOS DE REGISTROS NUMA SEQUENCIA DE SINONIMOS RECAIAM NUMA MESMA PAGINA. NOTEMOS, NO ENTANTO, QUE, DO MODO QUE A IDEIA FOI INICIALMENTE EXPOSTA, NAO FICOU EXPLICITO O USO DE CIRCULARIDADE DE EN-

DERECOS DENTRO DUMA PAGINA, DE MODO QUE, SEM CIRCULARIDADE , TEREMOS UM INCONVENIENTE: AO TENTARMOS FAZER UMA INSERCAO , SE O "HOME ADDRESS" CALCULADO CORRESPONDER A UM DOS ULTIMOS REGISTROS NA PAGINA, E POSSIVEL QUE A INSERCAO TENHA QUE SER FEITA NA PAGINA SEGUINTE, MESMO QUE HAJA REGISTROS DISPONIVEIS NA MESMA PAGINA, POREM ANTERIORES AO "HOME ADDRESS" CALCULADO. LEMBREMOS QUE AO ULTIMO REGISTRO DO ARQUIVO SEGUE-SE O SEU PRIMEIRO.

A 2A ALTERNATIVA CONSISTE NO USO DE "BUCKETS" COINCIDENTES COM AS PAGINAS, BASEADO NO FATO DE QUE GERALMENTE NAO HA NENHUMA RAZAO PREPONDERANTE PARA QUE O COMPRIMENTO DO "BUCKET" SEJA DIFERENTE DAQUELE DA PAGINA. NESTA ALTERNATIVA O "HASH" ENDERECA TAO SOMENTE O "BUCKET" E A RESOLUCAO DE SINONIMOS E FEITA POR PESQUISA SEQUENCIAL DENTRO DA PAGINA E A PARTIR DO SEU INICIO. SOMENTE QUANDO EXAMINARMOS TODA UMA PAGINA SEM SUCESSO E QUE PASSAREMOS A PAGINA SEGUINTE. ASSIM, ESTA ALTERNATIVAS TERA UM NUMERO MEDIO DE ACESSOS A PAGINAS ALGO INFERIOR A 1A ALTERNATIVA, EMBORA O NUMERO MEDIO DE ACESSOS A REGISTROS SEJA MAIOR (AS VEZES BEM MAIOR). LEMBREMOS QUE, A ULTIMA PAGINA DO ARQUIVO, SEGUE-SE A SUA PRIMEIRA. VER O ALGORITMO C, EM 3.2.1.

A 3A ALTERNATIVA CONSISTE NO USO DE "OVERFLOW" ABERTO (PESQUISA LINEAR), POREM COM EMPREGO DE CIRCULARIDADE DE ENDERECOS DENTRO DA MESMA PAGINA: SE, AO FAZERMOS UMA PESQUISA , ENCONTRARMOS O "HOME ADDRESS" CONTENDO UM SINONIMO E ASSIM TAMBEM OS REGISTROS SUCESSIVOS, ATE O ULTIMO REGISTRO DA PAGINA, CONTINUAREMOS A PESQUISA ATRAVES DOS REGISTROS 1, 2, 3, ETC DA MESMA, ATE ENCONTRARMOS O REGISTRO PROCURADO, UM REGISTRO DISPONIVEL, OU ATE RETORNARMOS AO "HOME ADDESS" , QUANDO SO ENTAO PASSAREMOS PARA A PROXIMA PAGINA. TEREMOS ASSIM UM NUMERO MEDIO DE ACESSOS A PAGINAS IGUAL AO DA 2A ALTERNATIVA E ALGO INFERIOR A 1A, E UM NUMERO MEDIO DE ACESSOS A REGISTROS BEM MENOR QUE O DA 2A E NO ENTORNO DO DA 1A. LEMBREMOS TAMBEM QUE, A ULTIMA PAGINA DO ARQUIVO, SEGUE-SE A SUA PRIMEIRA.

A 4A ALTERNATIVA, APONTADA POR SEVERANCE (43), CONSISTE NO USO DE "OVERFLOW" ENCADEADO DENTRO DA MESMA PAGINA, DE MODO QUE UM NOVO SINONIMO NAO SEJA ARMAZENADO NUMA OUTRA PAGINA ENQUANTO HOUVER QUALQUER "SLOT" LIVRE NA PAGINA CORRENTE. SUPONDO AS LISTAS DE SINONIMOS NAO COALESCENTES, TEREMOS UM NUMERO MEDIO DE ACESSOS A PAGINAS MENOR QUE O DA 3A E 2A ALTERNATIVAS E AINDA MENOR QUE O DA 1A; SUPONDO QUE AS LISTAS SAO COALESCENTES, TEREMOS ESSE NUMERO IGUAL AOS DA 3A E 2A ALTERNATIVAS; O NUMERO MEDIO DE ACESSOS A REGISTROS SERA DE MENOR A BEM MENOR QUE O DA 2A ALTERNATIVA. PORTANTO, ESTA ALTERNATIVA, EM TERMOS DE EFICIENCIA EM TEMPO, E ALGO MELHOR QUE A 3A ALTERNATIVA E MUITISSIMO MELHOR QUE A 2A E 1A. NO ENTANTO, HA UM CUSTO EM ESPACO ASSOCIADO AO APONTADOR DE SINONIMOS E ESTE CUSTO PODERA, EM ALGUNS CASOS, TORNAR-SE O FATOR DECISIVO NA ESCOLHA ENTRE A 4A ALTERNATIVA E AS ANTERIORES, DAS

QUAIS A 3A E TALVEZ A MAIS ATRAENTE, AO MENOS TEORICAMENTE.

APRESENTAMOS NO APENDICE UM ESTUDO EXPERIMENTAL E COMPARATIVO ENTRE AS QUATRO ALTERNATIVAS JA VISTAS. A 1A PARTE TRATA DA EFICIENCIA EM TEMPO E A 2A DA EFICIENCIA EM MEMORIA. O ESTUDO FOI EXCLUSIVAMENTE EXPERIMENTAL (PELO QUE NOS PARECE ATE TER MAIS VALOR) E ENVOLVEU A UTILIZACAO DE 2000 CHAVES NUMERICAS E RANDOMICAS. PREFERIMOS NAO UTILIZAR NUMEROS PSEUDO - RANDOMICOS, FORNECIDOS POR CERTOS ALGORITMOS, POIS A LEI DE FORMACAO QUE ELES TEM PODERIA FALSEAR DE UM MODO IMPREVISIVEL O NOSSO ESTUDO. USAMOS O "HASH" POR DIVISAO (COM TODOS OS CUIDADOS POSSIVEIS PARA GARANTIR SUA OTIMA PERFORMANCE), BASEADOS NO FATO DE QUE, COMO JA VIMOS, ESTUDOS TAIS COMO O DE LUM, YUEN & DODD (24 E 26) INDICAM QUE ESTE E QUASE SEMPRE O MELHOR METODO, NO COMPUTO GLOBAL.

EMBORA NAO HAJA UMA REGRA RIGIDA E UNIVERSAL PARA UMA ESCOLHA ENTRE AS QUATROS ALTERNATIVAS APRESENTADAS, QUE DEPENDERA DE UMA SERIE DE CARACTERISTICAS PECULIARES DA APLICACAO A SER FEITA E DAS INSTALACOES, A CONSULTA AO NOSSO ESTUDO PODERA OFERECER, EM MUITOS CASOS, SUBSIDIOS DE RAZOAVEL PESO PARA ESSA ESCOLHA. VEJAMOS DOIS EXEMPLOS:

A) AS INFORMACOES NECESSITAM DE 10 "BYTES", O POSSIVEL APONTADOR 2 "BYTES", O FATOR DE CARGA DESEJADO E 0.80 E O COMPRIMENTO DA PAGINA E 2,000 "BYTES"; TEREMOS:

	1ª OPCAO	2ª OPCAO	3ª OPCAO	4ª OPCAO
	"ABSEN-CIR"	"BUCKPAGE"	"ABCOCOM-CIR"	"ENCAPA-GE"
Nº DE REG. POR PAGINA	2000/10 =200	2000/10 =200	2000/10 = 200	2000/12 ≈ 166
(COMPR. INFO.)/(COMPR. APON.)	-	-	-	10/2 =5
Nº MEDIO DE ACESSOS A PAGINAS	1.007	1.000	1.000	1.000
Nº DE ACESSOS A REGISTROS	2.768	80.831	2.849	1.371
MEMORIA REAL-MENTE UTILIZA-DA PARA INFOR.	80%	80%	80%	66.7%

NESTE CASO, IRIAMOS PROVAVELMENTE OPTAR PELA 3A ALTERTIVA (OU PELA 1A), A NAO SER TALVEZ NUMA APLICACAO EM QUE A UTILIZACAO DE 20% A MAIS DE MEMORIA PELA 4A ALTERNATIVA NAO FOSSE CONSIDERADO NENHUM OBSTACULO. PODERIA TAMBEM SER VANTAJOSO O USO DE QUALQUER OUTRO METODO DE RESOLUCAO DE COLISOES, DESDE QUE ADOTASSEMOS AREAS ESPECIFICAS DE "OVERFLOW", DISTRIBUIDAS UMA AREA PARA CADA PAGINA.

B) AS INFORMACOES NECESSITAM 400 "BYTES", O POSSIVEL APONTADOR 2 "BYTES", O FATOR DE CARGA DESEJADO E 0.80 E O COMPRIMENTO DA PAGINA E DE 4,000 "BYTES"; TEREMOS:

	1ª OPCAO	2ª OPCAO	3ª OPCAO	4ª OPCAO
	"ABSEN-CIR"	"BUCKPAGE"	"ABCOCOM-CIR"	"ENCAPA-GE"
Nº DE REG. POR PAGINA	4000/400 = 10	000/400 = 10	4000/400 = 10	4000/402 ≈ 9
(COMPR. INFO.)/(COMPR. APON.)	-	-	-	400/2=200
Nº MEDIO DE ACESSOS A PAGINAS	1.330	1.213	1.213	1.083 INTERPO-LADO
Nº DE ACESSOS A REGISTROS	4.208	7.268	4.382	1.420
MEMORIA REAL-MENTE UTILIZA-DA PARA INFOR.	90%	90%	90%	89.5%

NESTE CASO, IRIAMOS QUASE CERTAMENTE OPTAR PELA 4A ALTERNATIVA (CASO NAO QUISESSEMOS USAR A SOLUCAO COM A UTILIZACAO DE UMA "SCATTER INDEX TABLE"). A 3A ALTERNATIVA AINDA SERIA TAMBEM ATRAENTE.

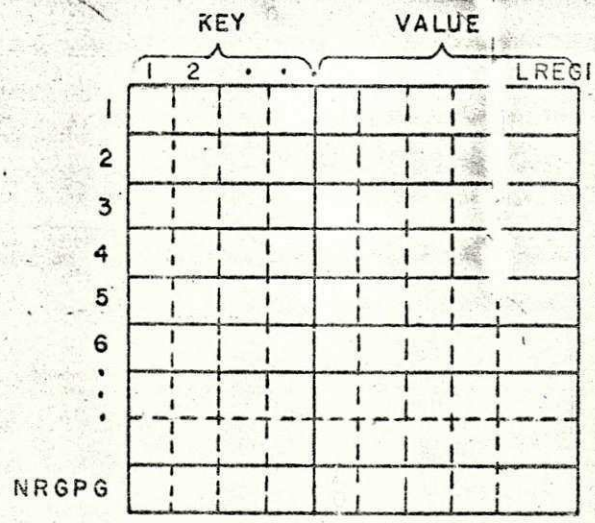
LEMBREMOS, NO ENTANTO, QUE TODAS AS CONSIDERACOES FEITAS ATE AQUI PODEM SER NEGLIGENCIADAS, SE TIVERMOS CERTEZA DE QUE TODAS AS PAGINAS REQUERIDAS PODEM E REALMENTE IRAO SER CONSERVADAS NA MEMORIA PRINCIPAL, ISTO E, SE NAO ESTIVERMOS USANDO VIRTUALIDADE DE MEMORIA.

APRESENTAMOS AGORA OS ALGORITMOS APENAS PARA A 3A E 4A AL-

TERNATIVAS, TANTO PORQUE NA MAIORIA DOS CASOS SAO AS MAIS A-
TRAENTES, COMO PORQUE OS OUTROS ALGORITMOS, SENDO MAIS SIM-
PLES, PODEM FICAR A CARGO DO LEITOR, SENDO MESMO CASOS PARTI-
CULARES DO ALGORITMO C.

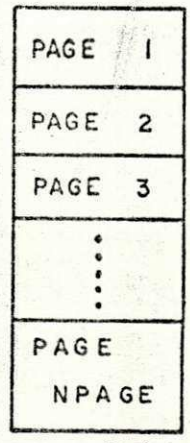
- ALGORITMO J ("ABCOMCIR"):
PESQUISA E INSERCAO DUMA CHAVE SEM USO DE INDEX, MINIMIZANDO O NUMERO DE ACESSOS A PAGINAS PELO EMPREGO DE "OVERFLOW" A-
BERTO PARA A RESOLUCAO DE SINONIMOS E COM CIRCULARIDADE DE ENDERECOS DENTRO DUMA MESMA PAGINA:
- ARGUMENTOS DE ENTRADA:
  K = CHAVE QUE QUEREMOS PESQUISAR OU INSERIR; K > 0;
  NRGPG = NUMERO DE REGISTROS POR PAGINA ("BUCKET SIZE");
  NPAGE = NUMERO TOTAL DE PAGINAS PARA O ARQUIVO;
  NREGI = NUMERO TOTAL DE "SLOTS" NO ARQUIVO = NRGPG * NPAGE.
- ARGUMENTOS DE RETORNO:
  FOUND = ".TRUE." SE A CHAVE FOI ENCONTRADA,
        = ".FALSE." SE A CHAVE NAO FOI ENCONTRADA;
  VALUE = VALOR (ES) ASSOCIADO (S) A K;
  IPOSI = APONTADOR PARA O REGISTRO QUE ESTAMOS EXAMINANDO.
- OUTRAS VARIAVEIS:
  KEY = CAMPO PARA A CHAVE NOS REGISTROS;
  INICR = APONTADOR PARA O PRIMEIRO REGISTRO QUE EXAMINAMOS NUMA PAGINA;
  INICP = APONTADOR PARA A PRIMEIRA PAGINA LOGICA QUE EXAMI-
NAMOS NO ARQUIVO;
  ICIRC = APONTADOR PARA O PROXIMO REGISTRO A SER EXAMINADO NUMA PAGINA, USANDO CIRCULARIDADE;
  KPAGE = APONTADOR PARA A PAGINA (LOGICA, NAO A DO SISTEMA OPERACIONAL) EM QUE ESTAMOS.

J.0 - (CONDICOES PREVIAS):
UMA UNICA VEZ EM TODA A EXITENCIA DO ARQUIVO, OU, MAIS PRECISAMENTE, QUANDO DO SEU NASCIMENTO, ZERAMOS O CAMPO KEY DE TODOS OS REGISTROS:
KEY (I) = 0 PARA 1< = I < = NREGI;
J.1 - (FAZ O "HASH" E GUARDA POSICOES INICIAIS):
IPOSI = K(MOD NREGI) + 1,
INICR = IPOSI,
KPAGE = (IPOSI - 1) / NRGPG + 1 (ARITMETICA INTEIRA),
INICP = KPAGE;
J.2 - (SE O NOVO REGISTRO ESTA VAZIO, INSERE):
SE KEY (IPOSI) = 0, INSIRA NA POSICAO IPOSI, FOUND = ".FALSE.", FIM.
J.3 - (AS CHAVES COINCIDEM ?):
SE KEY (IPOSI) = K, DETERMINE VALUE, FOUND = ".TRUE.", FIM.
J.4 - (AINDA NAO COMPLETOU O CIRCULO NA PAGINA ?):
ICIRC = (IPOSI + 1) - IPOSI / NRGPG * NRGPG + (KPAGE -



NRGPG * LREGI = LPAGE = COMPRIMENTO DA PÁGINA

PÁGINA QUALQUER



ARQUIVO LÓGICO

```
                1) * NRGPG (ARITMETICA INTEIRA),
        SE ICIRC = INICR, IPOSI = ICIRC E VA PARA J.2;
J.5 - (JA COMPLETOU O CIRCULO NO ARQUIVO ?):
        KPAGE = KPAGE + 1,
        SE KPAGE = INICP, FOUND = ".FALSE.", ASSINALE
                                  "OVERFLOW" DE ARQUIVO E FIM.
J.6 - (ESTA E A ULTIMA PAGINA (LOGICA) DO ARQUIVO ?):
        SE KPAGE > NPAGE,  KPAGE =1,
                           IPOSI = 1,
                           INICR = IPOSI,
                           VA PARA J.2;
        CASO CONTRARIO, IPOSI = (KPAGE - 1) * NRGPG +1,
                        INICR = IPOSI,
                        VA PARA J.2;
```

- ALGORITMO K ("ENCAPAGE"):

PESQUISA E INSERCAO DUMA CHAVE SEM USO DE INDEX, MINIMIZANDO O NUMERO DE ACESSOS A PAGINAS PELO EMPREGO DE "OVERFLOW" ENCADEADO PARA A RESOLUCAO DE SINONIMOS: PARA TERMOS UM

NUMERO DE ACESSOS A REGISTRO ALGO MENOR E PARA GUARDARMOS UMA CERTA HOMOGENEIDADE COM OS TRABALHOS DE LUM, YUEN & DODD (24 E 26), PREFERIMOS TOMAR AS LISTAS DE SINONIMOS SEPARADAS AO INVES DE COALESCENTES*; PODERIAMOS TOMAR AS LISTAS SEPARADAS SEM USAR APONTADORES PARA SEUS INICIOS E SEM AREA DE "OVERFLOW" EM SEPARADO DENTRO DE CADA PAGINA, MAS TERIAMOS QUE MODIFICAR O ALGORITMO "E , PARA LISTAS COALESCENTES, DE MODO QUE SE TORNARIA NECESSARIO MOVER MUITO FREQUENTEMENTE OS REGISTROS, NUM ARQUIVO COM CERTA VOLATILIDADE, DIMINUINDO A EFICIENCIA EM TEMPO; PODERIAMOS TOMAR AS LISTAS SEPARADAS COM AREAS DE "OVERFLOW" EM SEPARADO DENTRO DE CADA PAGINA, MAS ISSO, ALEM DE REQUERER UMA FUNCAO "HASH" UM POUCO MAIS COMPLICADA (VER 3.3.2), LEVARIA AO FATO DE QUE, PARA OBTERMOS UM DESEMPENHO EQUIVALENTE AO USO DE LISTAS SEPARADAS POREM SEM AREA (S) ESPECIFICA (S) PARA "OVERFLOW", PODERIAMOS NECESSITAR DE UM FATOR DE CARGA BEM MENOR, OU SEJA, DE BEM MAIS MEMORIA PARA A MESMA QUANTIDADE DE INFORMACOES (ISTO PODE SER VERIFICADO TANTO EXPERIMENTAL COMO TEORICAMENTE, USANDO-SE OS CONHECIMENTOS DE PROBABILIDADE E ESTATISTICA); POR TUDO ISSO, E LEVANDO EM CONSIDERACAO QUE ESTE ALGORITMO SERA MAIS USADO QUANDO OS REGISTROS FOREM RELATIVAMENTE GRANDES, QUANDO O CONSUMO DE 1 OU 2 "BYTES" A MAIS POR REGISTRO NAO REPRESENTARA PERCENTUALMENTE MUITA COISA, PREFERIMOS USAR O "OVERFLOW" ENCADEADO COM AS LISTAS DE SINONIMOS SEPARADAS E COM SEUS INICIOS APONTADOS POR UM VETOR IHEAD, QUE, PELA SUA FREQUENCIA DE USO, DEVERA ESTAR SEMPRE NA MEMORIA INTERNA, TENDO O MESMO NUMERO DE REGISTRO QUE O DO ARQUIVO DE DADOS. (VER O ALGORITMO D, EM 3.2.7.1).

DADA UMA CHAVE, FAZEMOS O "HASH" SOBRE A MESMA E NOS DIRIGIMOS AO REGISTRO CORRESPONDENTE DE IHEAD; SE ELE ESTIVER VAZIO, A LISTA DE SINONIMOS A ELE PERTINENTE ESTA AINDA VAZIA , PORTANTO DEVEMOS PROCURAR UM REGISTRO VAZIO NO ARQUIVO DE DADOS PARA FAZERMOS A INSERCAO, PRINCIPIANDO DA POSICAO TAMBEM CORRESPONDENTE AO "HASH" E PROSSEGUINDO SEQUENCIAL E CIRCULARMENTE DENTRO DAS PAGINAS; SE O REGISTRO DE IHEAD NAO ESTIVER VAZIO, PERCORREMOS A LISTA DE SINONIMOS ATE OBTERMOS SUCESSO OU ATE CHEGARMOS AO SEU FINAL, HIPOTESE EM QUE DEVEREMOS PROCURAR UM REGISTRO VAZIO NO ARQUIVO DE DADOS PARA FAZERMOS A INSERCAO, AGINDO AINDA SEQUENCIAL E CIRCULARMENTE DENTRO DAS PAGINAS.

NA VERDADE, ESTA SOLUCAO E COMO QUE UMA PARTICULARIZACAO DO USO DUMA "SCATTER INDEX TABLE", POIS O USO DO VETOR IHEAD

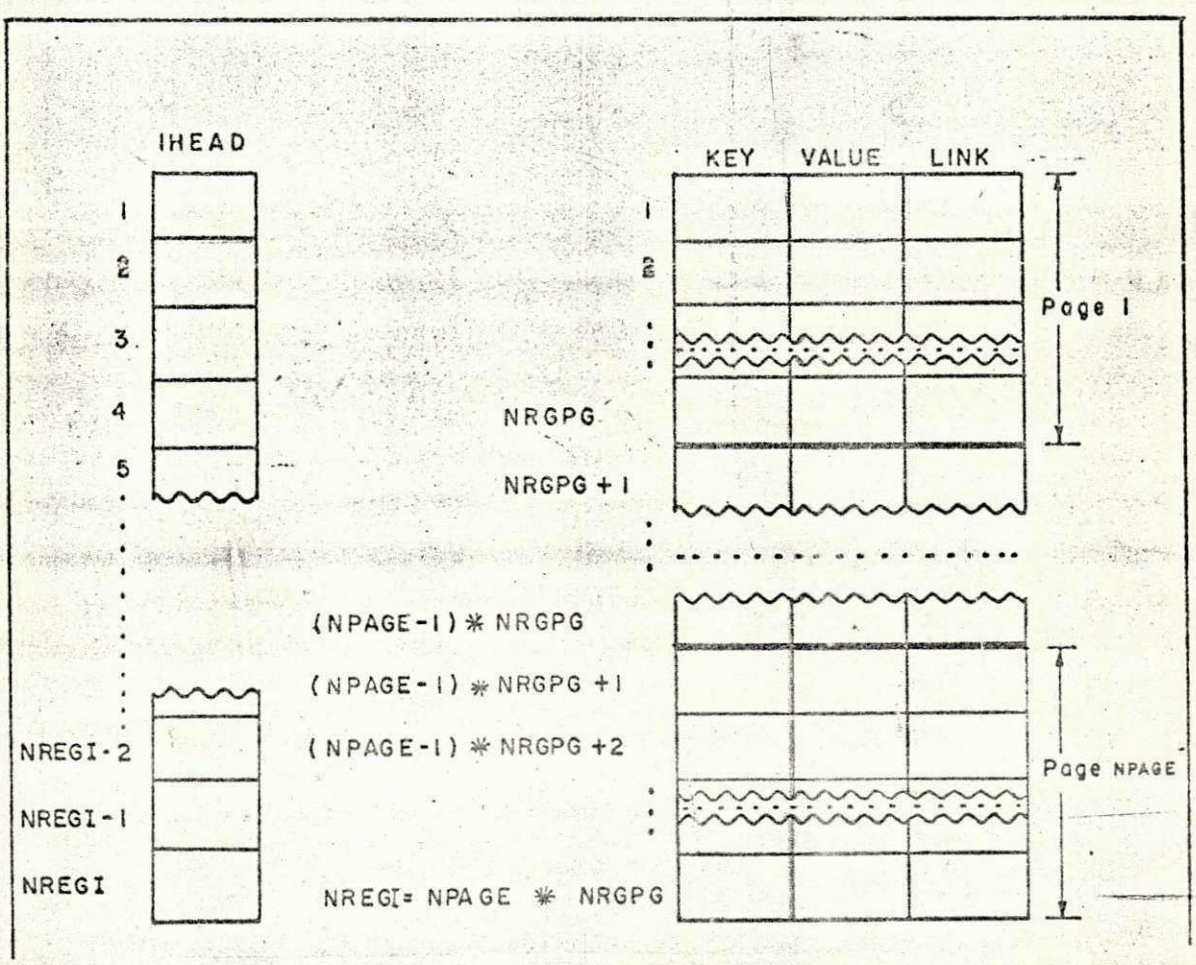
---

* EM UM TRABALHO PRATICO, A SOLUCAO COM USO DE LISTAS COALESCENTES IRA TALVEZ SER PREFERIVEL POIS TERA O MESMO NUMERO DE ACESSOS A PAGINAS E NAO NECESSITARA DO VETOR IHEAD, QUE O ATUAL ALGORITMO NECESSITA. NO ENTANTO, DEIXAMOS O TRABALHO DE ADAPTACAO DESTE ALGORITMO PARA O CASO DE LISTAS COALESCENTES A CARGO DO LEITOR.

TEM CERTA SEMELHANCA COM O USO DUM INDEX. RESSALTAMOS AQUI QUE, NA 2A PARTE DO NOSSO ESTUDO NO APENDICE, OU SEJA, NO ESTUDO DA EFICIENCIA EM MEMORIA, NAO LEVAMOS EM CONSIDERACAO O ESPACO NECESSARIO PARA O VETOR IHEAD DA SOLUCAO PARTICULAR REPRESENTADA POR ESTE ALGORITMO. ENFATIZAMOS NOVAMENTE QUE , EM CERTAS APLICACOES PRATICAS, O USO DE LISTAS COALESCENTES OU DE LISTAS SEPARADAS, POREM SEM O VETOR IHEAD, PODE SER MAIS ATRAENTE.

- VARIAVEIS:
  LINK = CAMPO PARA OS APONTADORES NOS REGISTROS;
  IHASH = VALOR DO "HASH" SOBRE A CHAVE K;
  IULTI = APONTADOR PARA O ULTIMO NODO DUMA LISTA DE SINONIMOS;

AS OUTRAS VARIAVEIS TEM O MESMO SIGNIFICADO QUE NO ALGORITMO ANTERIOR;



```
K.0 - (CONDICOES PREVIAS):
   UMA UNICA VEZ EM TODA A EXISTENCIA DO ARQUIVO, OU, MAIS
PRECISAMENTE, QUANDO DO SEU NASCIMENTO, PARA 1 <= I <= NREGI,
FAZEMOS:
        IHEAD (I) = 0,
        KEY (I) = 0,
        LINK (I) = 0;
K.1 - (FAZ O "HASH):
        IHASH = K (MOD NREGI) +1,
        IPOSI = IHASH;
K.2 - (AINDA NAO EXISTE UMA LISTA DE SINONIMOS APONTADA POR
        IHEAD (IHASH)):
        SE IHEAD (IHASH) ≠ 0 VA PARA K.3;
K.2.1 - (GUARDA AS POSICOES INICIAIS E COMECA A PESQUISA DE
         UM REGISTRO VAZIO):
         INICR = IPOSI,
         KPAGE = (IPOSI-1) / NRGPG + 1 (ARITMETICA INTEIRA),
         INICP = KPAGE;
K.2.2 - (SE O NOVO REGISTRO ESTA VAZIO, INSERE):
         SE  KEY (IPOSI) = 0, INSIRA NA POSICAO IPOSI,
             FOUND = ".FALSE.",
             LINK (IPOSI) = 0;
             SE IHEAD(IHASH) = 0, IHEAD(IHASH) = IPOSI;
             CASO CONTRARIO, LINK (IULT) = IPOSI;
             EM QUALQUER CASO, FIM.
K.2.3 - (AS CHAVES COINCIDEM ?):
         KEY (IPOSI) = K ?
K.2.3.1- SE SIM: DETERMINE VALUE, FACA FOUND = ".TRUE." E
         FIM.
K.2.3.2- SE NAO: PROSSIGA PARA K.2.4;
K.2.4 - (AINDA NAO COMPLETOU O CIRCULO NA PAGINA ?):
         ICIRC = (IPOSI + 1) - IPOSI / NRGPG * NRGPG + (KPAGE
                 - 1) * NRGPG (ARITMETICA INTEIRA);
         SE ICIRC ≠ INICR, IPOSI = INICR, VA PARA K.2.2;
K.2.5 - (JA COMPLETOU O CIRCULO NO ARQUIVO ?):
         KPAGE = KPAGE + 1,
         SE KPAGE = INICP, FOUND = ".FALSE.", ASSINALE "OVER-
             FLOW" DE ARQUIVO E FIM.
K.2.6 - (ESTA E A ULTIMA PAGINA (LOGICA) DO ARQUIVO ?):
         SE KPAGE > NPAGE,  KPAGE = 1,
                            IPOSI = 1,
                            INICR = IPOSI,
                            VA PARA K.2.2;
         CASO CONTRARIO:    IPOSI = (KPAGE -1) * NRGPG + 1,
                            INICR = IPOSI,
                            VA PARA K.2.2;
K.3 - (JA EXISTE LISTA DE SINONIMOS. FAZ A PESQUISA NA MES-
      MA.):
K.3.1 - (AS CHAVES COINCIDEM ?):
```

COLISOES; SE NAO, PODEREMOS PROCEDER SEMELHANTEMENTE AO QUE VIMOS EM 4.2, GUARDANDO NO INDEX SOMENTE AS CHAVES AUXILIARES.

4.3.5 - DELECOES EM MAQUINAS COM MEMORIA VIRTUAL PAGINADA:

"MUTATIS MUTANDIS", TUDO AQUILO QUE FOI DITO EM 3.4 SOBRE O PROBLEMA GENERICO DAS DELECOES QUANDO USAMOS O "HASHING", APLICA-SE TAMBEM QUANDO TEMOS UMA "SCATTER TABLE" EM UMA MAQUINA COM MEMORIA VIRTUAL PAGINADA.

EM PARTICULAR, SE TIVERMOS USADO O ALGORITMO J PARA FAZERMOS INSERCOES NUMA "SCATTER TABLE" EM UMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, UMA VEZ QUE ESSE ALGORITMO E SEMELHANTE AO C, O LEITOR PODERIA CHEGAR A UM ALGORITMO PARA DELECAO, SEMELHANTE AO F.

4.4 - CONCLUSOES SOBRE O USO DE "SCATTER INDEX TABLES", "VIRTUAL SCATTER TABLES" E "SCATTER TABLES" EM MAQUINAS COM MEMORIA VIRTUAL PAGINADA

4.4.1 - USO DE "SCATTER INDEX TABLES":

INICIALMENTE, NA SECCAO 4.1, ESTUDAMOS O USO DE "SCATTER INDEX TABLES", VERIFICANDO QUE O MESMO APRESENTA AS SEGUINTES VANTAGENS, AO MENOS NOS GRANDES ARQUIVOS COMERCIAIS TIPICOS, MANTIDOS EM MEMORIA SECUNDARIA:

1. - INDEPENDENCIA ENTRE A LOCALIZACAO FISICA DO REGISTRO E SEU IDENTIFICADOR;
2. - REDUCAO NA UTILIZACAO DE MEMORIA QUANDO O REGISTRO E RELATIVAMENTE GRANDE (DIGAMOS, MAIS DE 50 VEZES O COMPRIMENTO NECESSARIO PARA UM APONTADOR);
3. - REDUCAO NO TEMPO NECESSARIO PARA ACESSO A UM REGISTRO;
4. - FACILIDADE PARA LIDAR COM REGISTROS DE TAMANHO VARIAVEL;
5. - FACILIDADE PARA SE USAR MULTIPLOS CAMPOS COMO CHAVE;
6. - REDUCAO DO NUMERO DE COLISOES.

APRESENTAMOS ALGORITMO PARA PESQUISA (G), INSERCAO (H) E DELECAO (I) DE REGISTROS.

4.4.2 - USO DE "VIRTUAL SCATTER TABLES":

NA SECCAO 4.2, ESTUDAMOS O USO DE "VIRTUAL SCATTER TABLES". VERIFICAMOS QUE:

1. - SE PUDERMOS DESPREZAR O TEMPO DE EXAME DA CHAVE AUXILIAR FRENTE AO TEMPO DE PROVA DA CHAVE PROPRIAMENTE DITA, A "SCATTER TABLE" REAL PODE TER UM DESEMPENHO TAL E QUAL COMO SE FOSSE REALMENTE DO TAMANHO BEM MAIOR DA "VIRTUAL SCATTER TABLE" COM O BAIXISSIMO FATOR DE CARGA CORRESPONDENTE;
2. - O USO DE UMA "VIRTUAL SCATTER TABLE" PERMITE QUE SEU TAMANHO SEJA AUMENTADO DE UM MODO SIMPLES E EFICIENTE;
3. - SE A "VIRTUAL SCATTER TABLE" TIVER SEU TAMANHO DUMA ORDEM DE MAGNITUDE MUITISSIMO MAIOR QUE O TAMANHO REAL DA TABELA, EM MUITAS APLICACOES PRATICAS AS CHAVES PODEM DEIXAR COMPLETAMENTE DE SER EXAMINADAS, OU MESMO ARMAZENADAS.

4.4.3 - USO DE "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA:

NA SECCAO 4.3 ESTUDAMOS O USO DE "SCATTER TABLES" EM MAQUINAS COM MEMORIA VIRTUAL PAGINADA, PERCEBENDO-SE QUE REFERENCIAS CONSECUTIVAS A MEMORIA (NUMA CADEIA DE SINONIMOS) DEVEM INCIDIR TANTO QUANTO POSSIVEL NUMA MESMA PAGINA, OU EM PAGINAS QUE PROVAVELMENTE JA ESTEJAM NA MEMORIA PRINCIPAL.

4.4.3.1 - QUANTO AO USO DE INDEX, TEMOS DUAS CLASSES DE SOLUCOES:

1. - SOLUCAO SEM USO DE UM INDEX: PRESTA-SE MELHOR QUANDO OS REGISTROS SAO RELATIVAMENTE PEQUENOS (DIGAMOS, MENOS DE 50 VEZES O COMPRIMENTO NECESSARIO PARA UM APONTADOR), CABENDO UM RAZOAVEL NUMERO DE REGISTROS POR PAGINA (DIGAMOS, 20 OU MAIS). O FATOR DE CARGA NAO DEVE SER MUITO ALTO (DIGAMOS, NAO DEVE EXCEDER 0.90).
2. - SOLUCAO COM USO DE UM INDEX: PRESTA-SE MELHOR QUANDO OS REGISTROS SAO RELATIVAMENTE GRANDES (DIGAMOS, MAIS DE 100 VEZES O COMPRIMENTO NECESSARIO PARA UM APONTADOR), CABENDO UM PEQUENO NUMERO DE REGISTROS POR PAGINA (DIGAMOS, 10 OU MENOS) E, PODENDO-SE MANTER O INDEX (RELATIVAMENTE PEQUENO) SEMPRE NA MEMORIA PRINCIPAL, NECESSITAMOS DAS VANTAGENS APONTADAS EM 4.4.1. O FATOR DE CARGA REAL DO INDEX FICA GERALMENTE NA FAIXA DE 0.3 A 0.7.

4.4.3.2 - QUANTO AO USO DE VIRTUALIDADE (CONFORME VISTO EM 4.2) NA "SCATTER TABLE" TEMOS DUAS CLASSES DE SOLUCOES:

1. - SOLUCAO SEM USO DE VIRTUALIDADE NASCATTER TABLE" : PRESTA-SE MELHOR QUANDO A COMPACAO DAS CHAVES E DE UMA FACILIDADE NORMAL.
2. - SOLUCAO COM USO DE VIRTUALIDADE NASCATTER TABLE" : PRESTA-SE MELHOR QUANDO A COMPACAO DAS CHAVES E SOBREMODO COMPLICADA E DISPENDIOSA EM TEMPO.

4.4.4 - "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA, SEM USO DE INDEX E SEM USO DE VIRTUALIDADE (NO CONCEITO DA SECCAO 4.2):

PARA RESOLVERMOS AS COLISOES NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA SEM USO DE INDEX, E SEUSO DE VIRTUALIDADE NA "SCATTER TABLE", APRESENTAMOS (EM 4.3.4) METODOS E FIZEMOS EXPERIMENTOS PRATICOS PARA COMPARACAO DOSEUS DESEMPENHOS . DE UM MODO GERAL, PARA AS HIPOTESES DE 4.3.1 E 4.4.3.2.1 , NA ORDEM DECRESCENTE DE PREFERIBILIDADEESSES METODOS SAO :

1. - "ENCAPAGE": "OVERFLOW" ENCADEADO DENTRO DUMA MESMA PAGINA, (SOLUCAO OTIMA EM TEMPO, MAS SOMISSIVEL SE A UTILIZACAO ADICIONAL DE MEMORIA PARA APONTADORES FOR RELEGADA A SEGUNDO PLANO);
2. - "ABCOMCIR": "OVERFLOW" ABERTO (PESQUISA LINEAR), COM CIRCULARIDADE DE ENDERECOS DENTRO DUMA MESMA PAGINA;
3. - "ABSEMCIR": "OVERFLOW" ABERTO (PESQUISA LINEAR) SEM CIRCULARIDADE DE ENDERECOS DENTRO DUMA MESMA PAGINA; VER ALGORITMO C, EM 3.2.1, ADAPTADO PARA B = 1;
4. - "BUCKPAGE": USO DE "BUCKETS" COINCIDENTES COM AS PAGINAS; VER TAMBEM O ALGORITMO C, EM 3.2.1.

APRESENTAMOS ALGORITMOS PARA PESQUISE INSERCAO USANDO OS METODOS "ENCAPAGE" (K) E "ABCOMCIR" (J) DEIXAMOS OS OUTROS A CARGO DO LEITOR.

SE O NUMERO DE REGISTROS POR PAGINA FOR RAZOAVELMENTE GRANDE (DIGAMOS, MAIOR QUE 20) E O FATOR DE CARGA NAO FOR MUITO ALTO (DIGAMOS, NAO EXCEDER 0.90) NA PRATICA O METODO "ABSEMCIR" PODE QUASE SEMPRE SER O PREFERIDO, TENDO UMA BOA PERFORMANCE ALIADA A FACILIDADE E SIMPLICIDADE DE IMPLEMENTACAO. PODERIA TAMBEM SER VANTAJOSO O USO DE QUALQUER OUTRO METODO DE RESOLUCAO DE COLISOES, DESDE QUE ADOTASSEMOS AREAS ESPECIFICAS PARA "OVERFLOWS", DISTRIBUIDAS UMA AREA POR CADA PAGINA.

EM UMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, E DESACONSELHAVEL A RESOLUCAO DE COLISOES POR PESQUISA QUADRATICA, POR DUPLO "HASH", RANDOMICA, ENCADEADA OU BRYSSOTSKY, ONDE A RESOLUCAO DAS COLISOES E FEITA LIVREMENTE ATRAVES DE TODA A MEMORIA.

4.4.5 - "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA, SEM USO DE INDEX MAS COM USO DE VIRTUALIDADE (NO CONCEITO DA SECCAO 4.2):

PARA RESOLVERMOS AS COLISOES NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, SEM USO DE INDEX MAS COM USO DE VIRTUALIDADE NA "SCATTER TABLE", VER 4.3.3.

4.4.6 - "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA, COM USO DE INDEX:

PARA RESOLVERMOS AS COLISOES NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, COM USO DE INDEX, VER 4.3.4.

5 - CONCLUSOES, SUGESTOES PARA FUTUROS ESTUDOS

5.1 - CONCLUSOES

JULGAMOS TER FEITO UM RAZOAVEL ESTUDO GERAL SOBRE TODOS OS DETALHES QUE CONSEGUIMOS OBTER A RESPEITO DO "HASHING". EM PARTICULAR, FIZEMOS NO CAPITULO 4 UMA ANALISE DE "SCATTER INDEX TABLES", "VIRTUAL SCATTER TABLES" E "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA. NESTE ULTIMO CASO, FIZEMOS UM ESTUTUDO EXPERIMENTAL-COMPARATIVO DE 4 SOLUCOES ALTERNATIVAS. SE BEM QUE ESSAS ALTERNATIVAS JA TIVESSEM SIDO VISUALIZADAS PELOS ESTUDIOSOS, FAZIA-SE CONVENIENTE UM ESTUDO DA MANEIRA QUE FIZEMOS E QUE, AO MENOS NA BIBLIOGRAFIA CONSULTADA, AINDA NAO EXISTIA. APRESENTAMOS DIVERSOS ALGORITMOS, A MAIOR PARTE DELES SENDO CONSTITUIDA DE GENERALIZACOES DE ALGORITMOS JA DETALHADOS MAS SO PARA "BUCKET SIZE" IGUAL A 1, E O RESTANTE CONSTITUINDO ADAPTACOES, EXTENSOES E DETALHAMENTOS DE IDEIAS JA EXISTENTES.

5.1.1-QUANDO APLICAR O "HASHING":

O "HASHING" PRESTA-SE MUITO BEM PARA PESQUISAS EM "DIRECT ACCESS STORAGE DEVICES", QUANDO AS CHAVES A SEREM PESQUISADAS SURGEM NUMA SEQUENCIA ALEATORIA. O MAIOR PROBLEMA DO "HASHING" E A TOTAL FALTA DE ORDENACAO NUMA TABELA "HASH". VER 1.1, 1.2.16, 1.2.17 E 1.3.

5.1.2-ESCOLHA DA FUNCAO DE "HASH":

TOMADAS AS DEVIDAS PRECAUCOES (APONTADAS EM 2.4.4), O ME -

TODO DA DIVISAO E O ESCOLHIDO NA GRANDE MAIORIA DOS CASOS , COM O DIVISOR M PRIMO (OU NAO FATORAVEL POR NUMEROS MENORES QUE 20). VER 2.4, PARA COMPARACAO ENTRE AS FUNCOES DE "HASH", E 2.3.1, PARA DETALHES SOBRE O METODO DA DIVISAO.

5.1.3-ESCOLHA DO METODO DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS":

1. - EM MEMORIA PRINCIPAL, COM "BUCKET SIZE" IGUAL A 1 E FATOR DE CARGA CRESCENDO, OS MELHORES METODOS PARECEM SER GERALMENTE, EM ORDEM:
   A) PESQUISA ENCADEADA COM LISTAS SEPARADAS, SE O DISPENDIO COM APONTADORES NAO FOR SIGNIFICATIVO.
   B) PESQUISA ENCADEADA COM LISTAS COALESCENTES, SE O DISPENDIO COM APONTADORES NAO FOR SIGNIFICATIVO.
   C) PESQUISA LINEAR, VARIACAO PROPOSTA POR KRUTAR (VER 3.2.1).
   D) PESQUISA LINEAR, ALGORITMO C DE 3.2.1.
   E) PESQUISA POR DUPLO "HASH", OU DE VYSSOTSKY, OU QUADRATICA OU RANDOMICA.
2. - EM MEMORIA SECUNDARIA, COM "BUCKET SIZE" IGUAL A 1 E FATOR DE CARGA CRESCENDO, OS MELHORES METODOS PARECEM SER, EM ORDEM:
   A) PESQUISA ENCADEADA, COM LISTAS SEPARADAS (ALGORITMO D DE 3.2.7.1).
   B) PESQUISA ENCADEADA, COM LISTAS COALESCENTES (ALGORITMO E DE 3.2.7.2).
   C) PESQUISA LINEAR, VARIACAO PROPOSTA POR KRUTAR (VER 3.2.1).
   D) PESQUISA POR DUPLO "HASH", VARIACAO PROPOSTA POR BRENT (VER 3.2.3).
   E) PESQUISA POR DUPLO "HASH" (VER 3.2.2).
   F) PESQUISA DE VYSSOTSKY, QUADRATICA E RANDOMICA (VER 3.2.6, 3.2.5 E 3.2.4).
   G) PESQUISA LINEAR (ALGORITMO C DE 3.2.1).
3. - EM MEMORIA SECUNDARIA, COM "BUCKET SIZE" CRESCENDO E FATOR DE CARGA TAMBEM, OS MELHORES METODOS PARECEM SER , EM ORDEM:
   A) PESQUISA LINEAR E SUAS VARIACOES, DESDE QUE O FATOR DE CARGA NAO SEJA MUITO ALTO (DIGAMOS, P < = 0.85) E O "BUCKET SIZE" JA SEJA RAZOAVELMENTE GRANDE (DIGAMOS , B> = 10, OU MESMO B> = 20).
   B) PESQUISA ENCADEADA COM LISTAS SEPARADAS E AREA DE "OVERFLOW" EM SEPARADO, TENDO O "BUCKET" DE "OVERFLOW " CAPACIDADE IGUAL A 1 SE O FATOR DE CARGA NAO FOR MUITO ALTO, E CAPACIDADE MAIOR QUE 1 EM CASO CONTRARIO.
   C) PESQUISA ENCADEADA COM LISTAS COALESCENTES, TENDO O "BUCKET" DE "OVERFLOW" SUA CAPACIDADE IGUAL A 1 SE O

FATOR DE CARGA NAO FOR MUITO ALTO, E MAIOR QUE 1 EM CASO CONTRARIO.

D) OS OUTROS METODOS DE PESQUISA, OU SEJA, POR DUPLO "HASH" (SIMPLES E VARIACAO DE BRENT), PESQUISA QUADRATICA, RANDOMICA E DE VYSSOTSKY, SAO POUCO USADOS, E QUANDO O SAO O "BUCKET SIZE" TAMBEM DEVE SER MAIOR QUE 10 OU 20, E O FATOR DE CARGA DEVE SER NAO MUITO ALTO (DIGAMOS, P < = 0.85), QUANDO OS "BUCKETS" DE "OVERFLOW" PODERAO TER CAPACIDADE IGUAL A 1. EM CASO CONTRARIO, OS "BUCKETS" DE "OVERFLOW" DEVERAO TER CAPACIDADE MAIOR QUE 1.

4. - DE UM MODO GERAL, O USO DE AREA EM SEPARADO PARA "OVERFLOW" TRAZ CERTAS VANTAGENS. COM "BUCKET SIZE" PEQUENO (DIGAMOS, B< = 10) E USANDO-SE PESQUISA LINEAR, A RESOLUCAO DE "OVERFLOWS" NUMA AREA EM SEPARADO E MAIS EFICIENTE.

5.1.4-USO DE "SCATTER INDEX TABLES":

EM 4.1 ESTUDAMOS O USO DE "SCATTER INDEX TABLES", VERIFICANDO QUE O MESMO APRESENTA VARIAS VANTAGENS NOS GRANDES ARQUIVOS COMERCIAIS TIPICOS, MANTIDOS NA MEMORIA SECUNDARIA:

1. - INDEPENDENCIA ENTRE A LOCALIZACAO FISICA DO REGISTRO E SEU IDENTIFICADOR;
2. - REDUCAO NA UTILIZACAO DE MEMORIA QUANDO O REGISTRO E RELATIVAMENTE GRANDE (DIGAMOS, MAIS DE 50 VEZES O COMPRIMENTO NECESSARIO PARA UM APONTADOR);
3. - REDUCAO NO TEMPO NECESSARIO PARA ACESSO A UM REGISTRO;
4. - FACILIDADE PARA LIDAR COM REGISTROS DE TAMANHO VARIAVEL;
5. - FACILIDADE PARA SE USAR MULTIPLOS CAMPOS COMO CHAVE;
6. - REDUCAO DO NUMERO DE COLISOES.

5.1.5-USO DE "VIRTUAL SCATTER TABLES":

EM 4.2 ESTUDAMOS O USO DE "VIRTUAL SCATTER TABLES", VERIFICANDO QUE:

1. - SE PUDERMOS DESPREZAR O TEMPO DE EXAME DA CHAVE AUXILIAR FRENTE AO TEMPO DE PROVA DA CHAVE PROPRIAMENTE DITA, A "SCATTER TABLE" REAL PODE TER UM DESEMPENHO TAL E QUAL COMO SE FOSSE REALMENTE DO TAMANHO BEM MAIOR DA "VIRTUAL SCATTER TABLE" COM O BAIXISSIMO FATOR DE CARGA CORRESPONDENTE;
2. - O USO DE UMA "VIRTUAL SCATTER TABLE" PERMITE QUE SEU TAMANHO SEJA AUMENTADO DE UM MODO SIMPLES E EFICIENTE;
3. - SE A "VIRTUAL SCATTER TABLE" TIVER SEU TAMANHO DUMA OR-

DEM DE MAGNITUDE MUITISSIMO MAIOR QUE O TAMANHO REAL DA TABELA, EM MUITAS APLICACOES PRATICAS AS CHAVES PODEM DEIXAR COMPLETAMENTE DE SER EXAMINADAS, OU MESMO ARMAZENADAS.

5.1.6-USO DE "SCATTER TABLES" EM MEMORIA VIRTUAL PAGINADA:

PARA RESOLVERMOS AS COLISOES NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA SEM USO DE INDEX E SEM USO DE VIRTUALIDADE (NO CONCEITO DE 4.2) NA "SCATTER TABLE", APRESENTAMOS (EM 4.3.2) 4 METODOS E FIZEMOS EXPERIMENTOS PRATICOS PARA COMPARACAO DOS SEUS DESEMPENHOS. SE O NUMERO DE REGISTROS POR PAGINA FOR RAZOAVELMENTE GRANDE (DIGAMOS, MAIOR QUE 20) E O FATOR DE CARGA NAO FOR MUITO ALTO (DIGAMOS, NAO EXCEDER 0.90), O METODO .... "ABSEMCIR" ("OVERFLOW" ABERTO SEM CIRCULARIDADE DE ENDERECOS DENTRO DUMA MESMA PAGINA) PODE SER O PREFERIDO, NA PRATICA . VER 4.4.3. SE O NUMERO DE REGISTROS POR PAGINAS FOR BEM PEQUENO E O FATOR DE CARGA FOR MUITO ALTO, PODERA SER PREFERIDO O METODO "ENCAPAGE" ("OVERFLOW" ENCADEADO DENTRO DUMA MESMA PAGINA). EM AMBAS AS HIPOTESES, PODERIA TAMBEM SER ADOTADO UM OUTRO QUALQUER METODO DE RESOLUCAO DE COLISOES, DESDE QUE TIVESSEMOS AREAS ESPECIFICAS PARA "OVERFLOWS", DISTRIBUIDAS UMA AREA PARA CADA PAGINA.

PARA RESOLVERMOS AS COLISOES NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, SEM USO DE INDEX, MAS COM USO DE VIRTUALIDADE NA "SCATTER TABLE", VER 4.3.3.

PARA RESOLVERMOS AS COLISOES NUMA MAQUINA COM MEMORIA VIRTUAL PAGINADA, COM USO DE INDEX, VER 4.3.4.

5.2 - SUGESTOES PARA FUTUROS ESTUDOS

1.- FAZER UM GRANDE TRABALHO EXPERIMENTAL, SEMELHANTE AO DE LUM, YUEN & DODD (24) E CERTAMENTE MUITO MAIS TRABALHOSO , AGORA, POREM, PARA ESTABELECER EMPIRICAMENTE QUAL O COMPORTAMENTO DAS MAIS VARIADAS POSSIVEIS COMBINACOES DE METODOS PARA CALCULO DA FUNCAO DE "HASH" COM OS VARIOS METODOS POSSIVEIS PARA RESOLUCAO DE "BUCKET OVERFLOWS". ESTE ESTUDO DEVERA SER FEITO PARA MUITOS, GRANDES, TIPICOS E DIVERSIFICADOS ARQUIVOS PRE-EXISTENTES, NA MEMORIA SECUNDARIA E COM "BUCKET SIZE" VARIANDO, DIGAMOS, DE 1 A 100. ESTE E, DE

LONGE, O TRABALHO DE MAIOR NECESSIDADE ATUALMENTE. NUNCA DEVEMOS ESQUECER O ASPECTO OBSERVACIONAL DA CIENCIA DE COMPUTACAO: OS RESULTADOS MATEMATICOS OBTIDOS NESTA AREA SAO SEMPRE PASSIVEIS DE DISCUSSAO A RESPEITO DA VALIDADE DAS HIPOTESES, MODELOS E SIMPLIFICACOES ADOTADAS. VER ULLMAN... (46). JA SERIA MUITO DESEJAVEL QUE ESSE ESTUDO FOSSE FEITO AO MENOS QUE APENAS PARA O METODO DA DIVISAO.

2.- ANALISAR TEORICO-COMPARATIVAMENTE, PARA "BUCKET SIZES" MAIORES QUE 1, TODOS OS METODOS DE PESQUISA POR "HASH" (COM SUAS MUITAS VARIANTES E COMBINACOES POSSIVEIS) E QUE AINDA NAO FORAM ANALISADOS, PRINCIPALMENTE OS SUGERIDOS EM 3.6.2, SUB-ITEM 3.

3.- ESTUDAR, TEORICA E/OU EXPERIMENTALMENTE, QUAIS OS MELHORES METODOS PARA CALCULO DA FUNCAO DE "HASH" E RESOLUCAO DE COLISOES, AGORA, POREM, PARA CONJUNTOS DE CHAVES COM DISTRIBUICAO NAO UNIFORME (DIGAMOS, COM DISTRIBUICAO NORMAL, OU EXPONENCIAL, ETC.). ISTO E UMA EXTENSAO DOS TRABALHOS DE LUM, YUEN, DODD & [illegible] 13 E 26). INICIALMENTE O TRABALHO PODERIA SE [illegible] "BUCKET SIZE" IGUAL A 1, SENDO DEPOIS GENE[illegible] "BUCKET SIZES" MAIORES QUE 1.

4.- DESENVOLVER ESTUDOS COMPARATIVOS GENERICOS PARA OS TEMPOS DE CALCULO DAS DIVERSAS FUNCOES DE "HASH". PARTICULARIZAR PARA AS MAQUINAS MAIS REPRESENTATIVAS.

5.- DESENVOLVER ESTUDOS COMPARATIVOS GENERICOS, TEORICO E EXPERIMENTAIS, PARA O CUSTO GLOBAL DOS DIVERSOS:

1-METODOS DE CALCULO DA FUNCAO DE "HASH";

2-METODOS DE RESOLUCAO DE COLISOES E/OU "BUCKET OVERFLOWS";

3-COMBINACOES POSSIVEIS ENTRE ESSAS DUAS CLASSES DE METODOS ACIMA.

PARTICULARIZAR PARA AS MAQUINAS MAIS REPRESENTATIVAS.

6.- FAZER A DEMONSTRACAO MATEMATICA RIGOROSA DA VALIDADE DE TODOS OS ALGORITMOS APRESENTADOS NESTE TRABALHO, ESPECIALMENTE:

1-DAQUELES QUE SE REFEREM A "BUCKET SIZES" MAIORES QUE 1, OS QUAIS FORAM GENERALIZACOES FEITAS POR NOS DE ALGORITMOS JA EXISTENTES (MAS PARTICULARIZADOS PARA "BUCKET SIZE" IGUAL A 1).

2-DAQUELES QUE SE REFEREM AO USO DE "SCATTER TABLES" EM MAQUINAS COM MEMORIA VIRTUAL PAGINADA, ALGORITMOS ESTES QUE, AO QUE SAIBAMOS, NAO ESTAVAM EXPLICITAMENTE DETALHADOS NA LITERATURA SOBRE "HASHING".

DEVEMOS NOTAR QUE NAO FIZEMOS NENHUMA TENTATIVA DE DEMONSTRACAO MATEMATICA DA VALIDADE DESSES ALGORITMOS. LIMITAMO-NOS A TESTAR CADA UM DELES PARA CERCA DE 2,000 CHAVES "PERFEITAMENTE" RANDOMICAS (E NAO PSEUDO-RANDOMICAS) E PARA DIVERSOS FATORES DE CARGA, "BUCKET SIZES", TAMANHOS DE PAGINAS, ETC.

7.- OTIMIZAR OS ALGORITMOS APRESENTADOS, DE MODO QUE NAO SOMENTE FUNCIONEM, MAS O FACAM DE MANEIRA OTIMIZADA. ESTUDAR VARIAS VARIANTES PARA CADA ALGORITMO APRESENTADO, ESCOLHENDO A MAIS EFICIENTE, SOB UM ASPECTO GLOBAL.

8.- PROVER A PROXIMA GERACAO DE COMPUTADORES COM INSTRUCOES QUE EXECUTEM O "HASHING" DIRETAMENTE. PRESENTEMENTE, JA TEMOS INSTRUCOES EM MUITAS UNIDADES DE DISCO E TAMBOR QUE ENCONTRARAO UM REGISTRO EM UM "BUCKET", DESDE QUE SEJAM DADOS O ENDERECO DO "BUCKET" E A CHAVE DO REGISTRO, E QUE TRANSFERIRAO AUTOMATICAMENTE ESSE REGISTRO PARA A MEMORIA PRINCIPAL. DEVER-SE-AO BREVEMENTE DESENVOLVER INSTRUCOES QUE CALCULEM O "HOME ADDRESS" DUMA CHAVE (E OS ENDERECOS SUBSEQUENTES PARA RESOLUCAO DOS EVENTUAIS "BUCKET OVERFLOWS"), A NIVEL DE "HARDWARE" OU DE MICROPROGRAMACAO. COM ESSAS NOVAS INSTRUCOES, O PROGRAMADOR FARA O "HASHING" EM UM NIVEL BASTANTE ALTO E EFICIENTE.

```
***********************************************************************
*                         A P E N D I C E                             *
*                         ---------------                             *
***********************************************************************
```

ESTUDO COMPARATIVO DO DESEMPENHO DOS QUATRO-METODOS APRESENTADOS EM 4.3 PARA RESOLUCAO DE SINONIMOS NUMA MAQUINA COM MEMORIA VIRTUAL (PAGINACAO POR DEMANDA)

```
*****************************************************************************
```

-SIGLAS DOS METODOS DE RESOLUCAO DE SINONIMOS..

- ABSEMCIR.. USO DE 'OVERFLOW' ABERTO SEM CIRCULARIDADE DE ENDERECOS DENTRO DA MESMA PAGINA.
- BUCKPAGE.. USO DE 'BUCKETS' COM COMPRIMENTO IGUAL AO DA PAGINA, RESOLVENDO-SE OS SINONIMOS POR 'OVERFLOW' ABERTO, POREM PARTINDO-SE SEMPRE DO INICIO DA PAGINA.
- ABCOMCIR.. USO DE 'OVERFLOW' ABERTO COM CIRCULARIDADE DE ENDERECOS DENTRO DA MESMA PAGINA.
- ENCAPAGE.. USO DE 'OVERFLOW' ENCADEADO, POREM DENTRO DA MESMA PAGINA. AS LISTAS DE SINONIMOS SAO SEPARADAS (NAO COALESCENTES).

```
*****************************************************************************
```

1A. PARTE.. ESTUDO DA EFICIENCIA EM TEMPO.

-PARA CADA METODO A 1A. COLUNA FORNECE O NUMERO MEDIO DE ACESSOS A REGISTROS E A 2A. COLUNA O NUMERO MEDIO DE ACESSOS A PAGINAS QUE SE TORNARAM NECESSARIOS

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 1

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.460	1.460	1.460	1.460	1.460	1.237	1.237
0.55	1.550	1.550	1.550	1.550	1.550	1.550	1.254	1.254
0.60	1.698	1.698	1.698	1.698	1.698	1.698	1.281	1.281
0.65	1.818	1.818	1.818	1.818	1.818	1.818	1.303	1.303
0.70	1.990	1.990	1.990	1.990	1.990	1.990	1.326	1.326
0.75	2.227	2.227	2.227	2.227	2.227	2.227	1.345	1.345
0.80	2.768	2.768	2.768	2.768	2.768	2.768	1.371	1.371
0.85	3.253	3.253	3.253	3.253	3.253	3.253	1.395	1.395
0.90	4.208	4.208	4.208	4.208	4.208	4.208	1.420	1.420
0.95	8.263	8.263	8.263	8.263	8.263	8.263	1.447	1.447

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 2

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.231	1.660	1.164	1.501	1.164	1.237	1.088
0.55	1.550	1.278	1.764	1.206	1.593	1.206	1.254	1.100
0.60	1.698	1.350	1.921	1.274	1.745	1.274	1.281	1.120
0.65	1.818	1.404	2.059	1.332	1.875	1.332	1.303	1.132
0.70	1.990	1.493	2.243	1.417	2.054	1.417	1.326	1.149
0.75	2.227	1.615	2.499	1.537	2.306	1.537	1.345	1.167
0.80	2.768	1.886	3.052	1.807	2.855	1.807	1.371	1.191
0.85	3.253	2.128	3.557	2.053	3.357	2.053	1.395	1.218
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
0.95	8.263	4.630	8.638	4.577	8.439	4.577	1.447	1.271

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 5

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.096	2.261	1.018	1.453	1.018	1.237	1.012
0.55	1.550	1.117	2.423	1.035	1.560	1.035	1.254	1.022
0.60	1.698	1.148	2.651	1.062	1.743	1.062	1.281	1.033
0.65	1.818	1.169	2.819	1.078	1.865	1.078	1.303	1.038
0.70	1.990	1.206	3.046	1.104	2.053	1.104	1.326	1.048
0.75	2.227	1.249	3.338	1.146	2.309	1.146	1.345	1.061
0.80	2.768	1.363	3.899	1.243	2.827	1.243	1.371	1.081
0.85	3.253	1.457	4.430	1.334	3.343	1.334	1.395	1.102
0.90	4.208	1.645	5.304	1.492	4.205	1.492	1.420	1.123
0.95	8.263	2.460	9.599	2.334	8.474	2.334	1.447	1.146

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 10

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.054	3.465	1.002	1.488	1.002	1.237	1.001
0.55	1.550	1.066	3.715	1.005	1.606	1.005	1.254	1.002
0.60	1.698	1.082	4.018	1.011	1.744	1.011	1.281	1.007
0.65	1.818	1.092	4.313	1.018	1.891	1.018	1.303	1.011
0.70	1.990	1.113	4.665	1.030	2.092	1.030	1.326	1.015
0.75	2.227	1.135	5.041	1.047	2.348	1.047	1.345	1.023
0.80	2.768	1.188	5.658	1.089	2.874	1.089	1.371	1.039
0.85	3.253	1.234	6.299	1.134	3.453	1.134	1.395	1.055
0.90	4.208	1.330	7.268	1.213	4.382	1.213	1.420	1.073
0.95	8.263	1.735	11.233	1.590	8.305	1.590	1.447	1.096

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 20

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.030	5.884	1.000	1.479	1.000	1.237	1.000
0.55	1.550	1.034	6.369	1.000	1.569	1.000	1.254	1.000
0.60	1.698	1.040	6.903	1.001	1.711	1.001	1.281	1.000
0.65	1.818	1.046	7.481	1.003	1.838	1.003	1.303	1.003
0.70	1.990	1.055	8.014	1.008	2.042	1.008	1.326	1.004
0.75	2.227	1.066	8.604	1.014	2.293	1.014	1.345	1.007
0.80	2.768	1.092	9.369	1.029	2.844	1.029	1.371	1.016
0.85	3.253	1.115	10.230	1.050	3.496	1.050	1.395	1.025
0.90	4.208	1.163	11.493	1.091	4.648	1.091	1.420	1.041
0.95	8.263	1.367	15.859	1.288	8.905	1.288	1.447	1.062

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 50

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.010	13.413	1.000	1.467	1.000	1.237	1.000
0.55	1.550	1.015	14.587	1.000	1.551	1.000	1.254	1.000
0.60	1.698	1.018	15.844	1.000	1.684	1.000	1.281	1.000
0.65	1.818	1.019	17.062	1.000	1.804	1.000	1.303	1.000
0.70	1.990	1.022	18.312	1.000	1.964	1.000	1.326	1.000
0.75	2.227	1.025	19.559	1.000	2.173	1.000	1.345	1.000
0.80	2.768	1.036	20.951	1.003	2.702	1.003	1.371	1.002
0.85	3.253	1.043	22.313	1.006	3.238	1.006	1.395	1.004
0.90	4.208	1.059	24.080	1.017	4.414	1.017	1.420	1.012
0.95	8.263	1.143	28.153	1.075	8.058	1.075	1.447	1.027

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 100

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.007	26.012	1.000	1.479	1.000	1.237	1.000
0.55	1.550	1.008	28.367	1.000	1.582	1.000	1.254	1.000
0.60	1.698	1.009	30.827	1.000	1.709	1.000	1.281	1.000
0.65	1.818	1.009	33.339	1.000	1.834	1.000	1.303	1.000
0.70	1.990	1.011	35.856	1.000	2.011	1.000	1.326	1.000
0.75	2.227	1.012	38.324	1.000	2.252	1.000	1.345	1.000
0.80	2.768	1.019	40.819	1.000	2.785	1.000	1.371	1.000
0.85	3.253	1.022	43.294	1.000	3.327	1.000	1.395	1.000
0.90	4.208	1.032	46.069	1.003	4.581	1.003	1.420	1.002
0.95	8.263	1.074	49.997	1.018	7.042	1.018	1.447	1.011

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 200

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.002	50.676	1.000	1.461	1.000	1.237	1.000
0.55	1.550	1.002	55.711	1.000	1.550	1.000	1.254	1.000
0.60	1.698	1.003	60.750	1.000	1.700	1.000	1.281	1.000
0.65	1.818	1.003	65.812	1.000	1.842	1.000	1.303	1.000
0.70	1.990	1.003	70.818	1.000	2.016	1.000	1.326	1.000
0.75	2.227	1.004	75.875	1.000	2.272	1.000	1.345	1.000
0.80	2.768	1.007	80.831	1.000	2.849	1.000	1.371	1.000
0.85	3.253	1.008	85.708	1.000	3.350	1.000	1.395	1.000
0.90	4.208	1.012	90.777	1.000	4.452	1.000	1.420	1.000
0.95	8.263	1.035	96.902	1.006	7.373	1.006	1.447	1.004

NUMERO DE REGISTROS POR PAGINA (BUCKET SIZE).. 500

FATOR DE CARGA	ABSEMCIR		BUCKPAGE		ABCOMCIR		ENCAPAGE	
0.50	1.460	1.000	125.725	1.000	1.460	1.000	1.237	1.000
0.55	1.550	1.001	138.093	1.000	1.550	1.000	1.254	1.000
0.60	1.698	1.001	150.556	1.000	1.698	1.000	1.281	1.000
0.65	1.818	1.001	163.058	1.000	1.818	1.000	1.303	1.000
0.70	1.990	1.001	175.640	1.000	1.990	1.000	1.326	1.000
0.75	2.227	1.001	188.157	1.000	2.227	1.000	1.345	1.000
0.80	2.768	1.001	200.777	1.000	2.754	1.000	1.371	1.000
0.85	3.253	1.001	213.169	1.000	3.239	1.000	1.395	1.000
0.90	4.208	1.006	225.667	1.000	4.200	1.000	1.420	1.000
0.95	8.263	1.016	238.518	1.001	7.384	1.001	1.447	1.000

****

2A. PARTE.. ESTUDO DA EFICIENCIA EM MEMORIA

-METODOS 'ABSEMCIR', 'BUCKPAGE', E 'ABCOMCIR'.. A PERCENTAGEM DE MEMORIA REALMENTE APROVEITADA PARA AS INFORMACOES COINCIDE COM O FATOR DE CARGA USADO, POIS NAO HA CONSUMO COM APONTADORES.

****

*-METODO 'ENCAPAGE'.. A PERCENTAGEM DE MEMORIA REALMENTE APRO- *
* VEITADA PARA AS INFORMACOES (EXCLUINDO A TECNICA DO USO DE*
* UM VETOR SO COM APONTADORES PARA O INICIO DAS LISTAS) EH..*

****

* FATOR *	RELACAO ENTRE O COMPRIMENTO TOTAL DAS INFORMACOES E O DO APONTADOR								
* DE CARGA *	1	2	5	10	20	50	100	200	500
0.50	0.250	0.333	0.417	0.455	0.476	0.490	0.495	0.498	0.499
0.55	0.275	0.367	0.458	0.500	0.524	0.539	0.545	0.547	0.549
0.60	0.300	0.400	0.500	0.546	0.572	0.588	0.594	0.597	0.599
0.65	0.325	0.433	0.542	0.591	0.619	0.637	0.644	0.647	0.649
0.70	0.350	0.467	0.583	0.636	0.667	0.686	0.693	0.697	0.699
0.75	0.375	0.500	0.625	0.682	0.714	0.735	0.743	0.746	0.749
0.80	0.400	0.533	0.667	0.727	0.762	0.784	0.792	0.796	0.799
0.85	0.425	0.567	0.708	0.773	0.810	0.833	0.842	0.846	0.848
0.90	0.450	0.600	0.750	0.818	0.857	0.882	0.891	0.896	0.898
0.95	0.475	0.633	0.792	0.864	0.905	0.931	0.941	0.945	0.948

BIBLIOGRAFIA

(01) ACKERMAN, A. F. "QUADRATIC SEARCH FOR HASH TABLES OF SIZE P**N", COMM. ACM 17, 3 (MARCH 1974), PG. 164.

(02) BATAGELJ, V. "THE QUADRATIC HASH METHOD WHEN THE TABLE SIZE IS NOT A PRIME NUMBER", COMM. ACM 18, 4 (APRIL 1975).

(03) BAYS, C. "THE REALLOCATION OF HASH-CODED TABLES", COMM . ACM 16, 1 (JAN. 1973), PG. 11-14.

(04) BELL, J. R. "THE QUADRATIC QUOTIENT METHOD: A HASH CODE ELIMINATING SECONDARY CLUSTERING", COMM. ACM 13, 2 (FEB. 1970), PG. 107-109.

(05) BELL, J. R.; AND KAMAN, C. H. "THE LINEAR QUOTIENT HASH CODE", COMM. ACM 13, 11 (NOV. 1970), PG. 675-677.

(06) BRENT, R. P. "REDUCING THE RETRIEVAL TIME OF SCATTER STORAGE TECHNIQUES", COMM. ACM 16, 2 (FEB. 1973), PG. 105-109.

(07) BUCHHOLZ, W. "FILE ORGANIZATION AND ADDRESSING", IBM SYSTEMS J. 2 (JUNE 1963), PG. 86-111.

(08) DAY, A. C. "FULL TABLE QUADRATIC SEARCHING FOR SCATTER STORAGE", COMM. ACM 13, 8 (AUGUST 1970), PG. 481-482.

(09) DENNING, P. J. "VIRTUAL MEMORY", COMPUTING SURVEYS 2, 3 (SEPT. 1970).

(10) DIPPEL, G. ; AND HOUSE, W. C. "INFORMATION SYSTEMS ", SCOTT, FORESMAN & CO., GLENVIEW, ILL., 1969.

(11) DUMMEY, A. I. "INDEXING FOR RAPID RANDOM-ACCESS MEMORY SYSTEMS", COMPUTERS AND AUTOMATION 5,12 (DEC. 1956), PG. 6-9.

(12) FOSTER, C. C. "A GENERALIZATION OF AVL TREES", COMM. ACM 16, 8 (AUGUST 1973).

(13) GHOSH, S. P.; AND LUM, V. Y. "AN ANALYSIS OF COLLISIONS WHEN HASHING BY DIVISION", TECH. REPORT RJ-1218, IBM , MAY 1973.

(14) HANAN, M. ; AND PALERMO, F. P. "AN APPLICATION OF CODING THEORY TO A FILE ADDRESS PROBLEM", IBM J. RES. & DEVELOPMENT 7, 2 (APRIL 1963), PG. 127-129.

(15) HEISING, W. P., "NOTE ON RANDOM ADDRESSING TECHNIQUES" , IBM SYSTEMS J. 2 (JUNE 1963), PG. 112-116.

(16) HELLERMAN, H. "DIGITAL COMPUTER SYSTEM PRINCIPLES", MC-GRAW-HILL, NEW YORK, 1967.

(17) IBM "INTRODUTION TO IBM DIRECT ACCESS STORAGE DEVICES AND ORGANIZATION METHODS - STUDENT TEXT". 1973.

(18) JOHNSON, L. R. "AN INDIRECT CHAINING METHOD FOR ADDRESSING ON SECUNDARY KEYS", COMM. ACM 4,5 (MAY 1961), PG. 218-222.

(19) KNUTH, D. E. "THE ART OF COMPUTER PROGRAMMING", VOL. II: SEMINUMERICAL ALGORITHMS, ADDISON WESLEY, READING, MASS., 1969.

(20) KNUTH, D. E. "THE ART OF COMPUTER PROGRAMMING" VOL. III: SORTING AND SEARCHING, ADDISON-WESLEY, READING, MASS., 1973.

(21) KNUTH, D. E. "COMPUTER SCIENCE AND ITS RELATION TO MATHEMATICS", AMER. MATH. MONTHLY 81, 4 (APRIL 1974), PG. 323-343.

(22) LIN, A. D. "KEY ADDRESSING OF RANDOM ACCESS MEMORIES BY RADIX TRANSFORMATION", PROC. 1963 SPRING JOINT COMPUTER CONF. AFIPS VOL. 23, SPARTAN BOOKS, BALTIMORE, 1963, PG. 355-366.

(23) LUCCIO, F. "WEIGHTED INCREMENT LINEAR SEARCH FOR SCATTER TABLES", COMM. ACM 15, 12 (DEC. 1972), PG. 1045-1047.

(24) LUM, V. Y. ; YUEN, P. S. T. ; AND DODD, M. "KEY-TO-ADDRESS TRANSFORM TECHNIQUES: A FUNDAMENTAL PERFORMANCE STUDY ON LARGE EXISTING FORMATTED FILES", COMM. ACM 14, 4 (APRIL 1971), PG. 228-239.

(25) LUM, V. Y. ; AND YUEN, P. S. T. "ADDITIONAL RESULTS ON KEY-TO-ADDRESS TRANSFORM TECHNIQUES", COMM. ACM 15, 11 (NOV. 1972), PG. 996-997.

(26) LUM, V. Y. "GENERAL PERFORMANCE ANALYSIS OF KEY-TO-ADDRESS TRANSFORMATION METHODS USING AN ABSTRACT FILE CONCEPT", COMM. ACM 16, 10 (OCT. 1973), PG. 603-612.

(27) MARTIN, J. "COMPUTER DATA BASE ORGANIZATION", PRENTICE-HALL, ENGLEWOOD CLIFFS, N. J., 1975.

(28) MAURER, W. D. "AN IMPROVED HASH CODE FOR SCATTER STORAGE", COMM. ACM 11, 1 (JAN. 1968), PG. 35-38.

(29) MAURER, W. D. "PROGRAMMING", HOLDEN-DAY, SAN FRANCISCO, CALIF., 1968.

(30) MAURER, W. D. ; AND LEWIS, T. G. "HASH TABLE METHODS", COMPUTING SURVEYS 7, 1 (MARCH 1975), PG. 5-19.

(31) MAXWELL, W. L. ; AND SEVERANCE, D. G. "COMPARISON OF ALTERNATIVES FOR THE REPRESENTATION OF DATA ITEMS VALUES IN AN INFORMATION SYSTEM", CORNELL UNIVERSITY, 1974.

(32) MCILROY, M. D. "A VARIANT METHOD OF FILE SEARCHING", COMM. ACM 6, 3 (MARCH 1963), PG. 101.

(33) MORRIS, R. "SCATTER STORAGE TECHNIQUES", COMM. ACM 11, 1 (JAN. 1968), PG. 38-43.

(34) NAYLOR, T. H. ; BALINTFY, J. L. ; BURDICK, D. S. ; AND CHU, K. "TECNICAS DE SIMULACAO EM COMPUTADORES", EDITORA VOZES LTDA., PETROPOLIS, RJ., BRASIL, 1971.

(35) NIEVERGELT, J. "BINARY SEARCH TREES AND FILE ORGANIZATION", COMPUTING SURVEYS 6, 3 (SEPT. 1974), PG. 195-207.

(36) OLSON, C. A. "RANDOM ACCESS FILE ORGANIZATION FOR INDI - RECTLY ACCESSED RECORDS", PROC. ACM 24TH NATIONAL CONF., 1969, PG. 539-549.

(37) PETERSON, W. W. "ADDRESSING FOR RANDOM-ACCESS STORAGE", IBM J. RES. & DEVELOPMENT 1,2 (APRIL 1957), PG. 130-146.

(38) PETERSON, W. W. "ERROR CORRECTING CODES", M.I.T. PRESS, CAMBRIDGE, MASS., 1961.

(39) PRICE, C. E. "TABLE LOOKUP TECHNIQUES", COMPUTING SUR - VEYS 3, 2 (JUNE 1971), PG. 49-65.

(40) RADKE, C. E. "THE USE OF QUADRATIC RESIDUE RESEARCH" , COMM. ACM 13, 2 (FEB. 1970), PG. 103-105.

(41) SCHAY, G. ; AND RAVER, N. "A METHOD FOR KEY-TO-ADDRESS TRANSFORMATION", IBM J. RES. & DEVELOPMENT 7, 2 (APRIL 1963), PG. 121-126.

(42) SCHAY, G. ; AND SPRUTH, W. G. "ANALYSIS OF A FILE AD- DRESSING METHOD", COMM. ACM 5, 8 (AUGUST 1962), PG. 459- 462.

(43) SEVERANCE, D. G. "IDENTIFIER SEARCH MECHANISMS: A SURVEY AND GENERALIZED MODEL", COMPUTING SURVEYS 6, 3 (SEPT . 1974), PG. 175-194.

(44) SUSSENGUTH, E. H. JR. "USE OF TREE STRUCTURES FOR PRO - CESSING FILES", COMM. ACM 6, 5 (MAY 1963).

(45) TAINITER, M. "ADDRESSING FOR RANDOM-ACCESS STORAGE WITH MULTIPLE BUCKET CAPACITIES", J. ACM 10, 3 (JULY 1963) , PG. 307-315.

(46) ULLMAN, J. D. "A NOTE ON THE EFFICIENCY OF HASHING FUNC- TIONS", J. ACM 19, 3 (JULY 1972), PG. 569-575.

(47) VAN DER POOL, J. A. "OPTIMUM STORAGE ALLOCATION FOR INI- TIAL LOADING OF A FILE", IBM J. RES. & DEVELOPMENT 16, 6 (NOV. 1972), PG. 579-586.

(48) VAN DER POOL, J. A. "OPTIMUM STORAGE ALLOCATION FOR A FI- LE IN STEADY STATE", IBM J. RES. & DEVELOPMENT 17, 1 (JAN. 1973), PG. 27-38.

(49) WEBB, D. A. "THE DEVELOPMENT AND APPLICATION OF AN EVA - LUATION MODEL FOR HASH CODING SYSTEMS", PHD. THESIS, SY- RACUSE UNIVERSITY, SYRACUSE, N. Y., AUGUST 1972.

(50) WEGNER, P. "PROGRAMMING LANGUAGES, INFORMATION STRUCTU - RES AND MACHINE ORGANIZATION", MCGRAW-HILL, NEW YORK, 1968.