Instituto de Computação
Departamento de Ciência da Computação
Universidade Federal da Bahia (UFBA) - Salvador, BA - Brasil

MATA54 - Estrutura de Dados II
Prof. Flávio Morais de Assis Silva

Alunos grupo 1: João Lucas Lima de Melo (joaollm@ufba.br); Luca Assis Argolo (luca.argolo@ufba.br)

Relatório de Implementação Hashing Perfeito em Dois Níveis

Sumário.

Introdução.

O segundo trabalho prático da disciplina Estrutura de Dados e Algoritmos II propõe, como visto em sala, a implementação de uma estrutura de hashing perfeito em dois níveis. Nos baseamos nos conteúdos abordados em aula e no livro *Algoritmos: Teoria e Prática* para o desenvolvimento do projeto.

A nossa estrutura foi implementada com o uso de três componentes: Um arquivo contendo a tabela de nível um, o número primo e os valores *a* e *b*; Um conjunto de arquivos auxiliares contendo apenas os registros alocados por índice (usado apenas durante o período de construção); E um conjunto de arquivos contendo os registros já alocados em suas devidas posições (arquivo final usado para as consultas O(1)).

A tabela de nível um comporta uma quantidade *m* (entrada do usuário) de *Células* (estrutura criada para armazenar a quantidade de elementos no índice, e os valores de *a* e *b* para cada índice), e três inteiros *p* (o número primo global), *a e b* (valores para função hash de primeiro nível)

Uma vez inseridos os registros, eles são inicialmente alocados em um arquivo de tamanho *ìndice_célula.m* de *Registros*. Cada estrutura *Registro* armazena os valores *chave, nome* e *idade*, sua posição *hash* (em função do *a* e *b* usados para o primeiro nível) e uma flag

ocupado (indica se o registro é vazio ou não). Registros alocados na posição *i* da tabela de nível um serão inseridos no arquivo *i.dat*.

Sabendo quais registros pertencem ao índice *i* da tabela de nível um, agrupados no arquivo *i*, é criado um arquivo *[i]aloc.dat* (0aloc.dat, 2aloc.dat, etc), de tamanho *ìndice_célula.m* * *ìndice_célula.m* de *Registros*. O arquivo é ocupado por registros vazios (cujo valor de *ocupado* é atribuído a 0), e então itera-se sobre os registros do arquivo *i* e os aloca, usando os valores de *a* e *b* atribuídos na célula de índice *i* da tabela de nível um, em suas respectivas posições usando a função de hash. Se houver colisão, os valores de *a* e *b* são atualizados e o processo de alocação é reiniciado.

Os registros e células sempre foram manipulados por meio do uso de funções gerenciadoras de arquivos, não havendo alocação em memória principal. Os arquivos foram escritos de tal forma que possam ser consultados uma vez escritos e o programa ter sido finalizado.

Os Tipos de Estruturas.

Decidimos criar três principais estruturas que serão iteradas, inseridas e manipuladas pela aplicação: *Célula, Registro* e *Dados*.

Célula: Estrutura utilizada na tabela de nível um. Comporta os valores *mtab* (referente à quantidade de registros no índice da célula em questão)*, a* (referente ao parâmetro *a* a ser utilizado na função hash dos registros no índice da célula em questão) e *b* (referente ao parâmetro *b* a ser utilizado na função hash dos registros no índice da célula em questão).

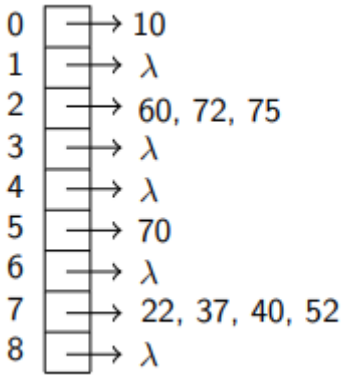

Figura 1. (Retirado do slide de Hashing Universal e Hashing Perfeito)

Tomando como exemplo a tabela de nível um da *Figura 1*. O bloco de índice 7 seria um *Célula* cujo valor *mtab* = 4, e os valores de *a* e *b* atribuídos à célula 7 seriam os utilizados para a função de hash de segundo nível para os registros de chave 22, 37, 40 e 52.

Dados: Estrutura utilizada para comportar os valores *chave, nome* e *idade* de um registro.

Registro: Estrutura utilizada nas tabelas de segundo nível. Comporta os valores de *Dados* atribuídos a um registro, uma flag *ocupado* e sua posição *hash* (varia entre primeiro e segundo nível a depender da operação na aplicação).

Tomando ainda como exemplo a tabela de nível um da Figura 1, o número 22 estaria representando um registro de *chave* 22, sem atribuição visível de *nome* e *idade*, cujo valor *ocupado* = 1 e *hash* = 7 (em função do primeiro nível).

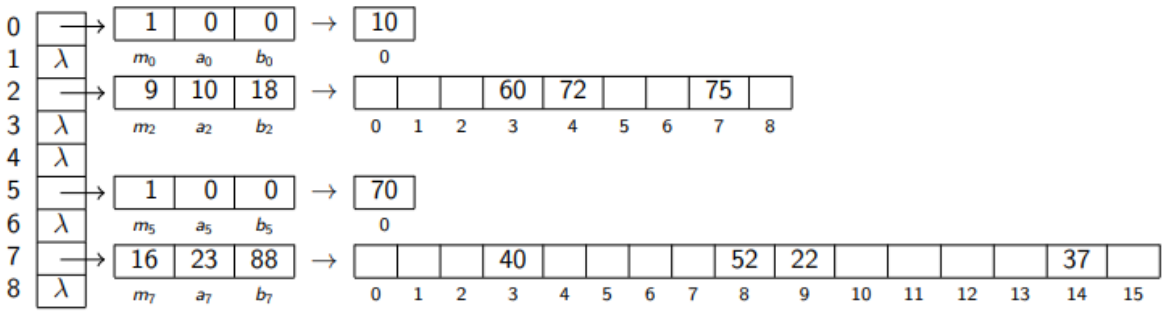

Figura 2. (Retirado do slide de Hashing Universal e Hashing Perfeito)

Tomemos agora como referência a Figura 2. Uma vez implementadas as tabelas de nível dois, analisando ainda o registro de *chave* 22, todos os valores seguem inalterados, com exceção de *hash*, agora atribuído para 9.

A Tabela de Nível Um.

De acordo com a especificação, a primeira função do programa será a de inserir registros, onde a primeira entrada consiste do caractere "i", informando a operação a ser realizada e um inteiro *m*, o número de registros a serem inseridos na estrutura.

Recebido *m*, é invocada a função *insereRegistros()*, que recebe como parâmetro o tamanho da tabela de nível um e um ponteiro para *a*, *b*, e *p* globais. Essa função irá receber os dados a serem salvos nos registros e irá salvar esses registros por indice nos arquivos auxiliares. Ao receber o primeiro conjunto de dados, o valor de *p* será gerado com base no valor da chave inserida, e os valores de *a* e *b* serão gerados aleatoriamente e modularizados para *p*. Em seguida a função *criaArquivoNivelUm()* será chamada e a mesma cria o arquivo *nivelUm.dat* contendo *m Células* e os valores de *p*, *a* e *b*.

Enquanto o arquivo é criado, novas células são atribuídas, seus valores *mtab* são atribuídos a 0, já que ainda não existem registros para o índice correspondente. Os valores de *a* e *b* da célula são gerados aleatoriamente, mas podem assumir valores fixos caso necessário.

A fim de controle e teste, definimos os valores de *a* e *b* dos índices 0, 2, 5, e 7 para os mesmos da Figura 2, exemplo dado em sala de aula. Para as demais células, são atribuídos valores aleatórios.

Ao término da inserção de células, o arquivo *nivelUm.dat* foi inicializado com *m* células.

As Tabelas de Nível Dois.

Uma vez criado o arquivo *nivelUm.dat* (a tabela de nível um), continuaremos na função *insereRegistros()* onde ao recebermos individualmente os dados da entrada, usamos um registro auxiliar *r* para receber os valores *chave*, *nome* e *idade* do registro a ser inserido.

Usamos o primo *p*, valores globais (de primeiro nível) de *a* e *b*, quantidade de células *m* e *chave* para calcular, a partir da função hash, a posição do registro na tabela de primeiro nível. Tentamos abrir um arquivo de nome *i,* índice da posição do registro. Caso não exista, criamos o arquivo. Uma vez aberto, inserimos *r* no arquivo e, na célula de índice *i*, incrementamos em 1 o valor de *celula.mtab*.

Realizamos esse processo para todos os registros a serem inseridos. No final da execução, usando como referência a Figura 1, teremos 4 arquivos de nome:
0. guardando o registro de chave 10,
2. guardando os registros de chave 60, 72 e 75,
5. guardando o registro de chave 70,
7. guardando o registro de chave 22, 37, 40, 52

Para isso, iteramos sobre a tabela de nível um, conferindo se na posição iterada possui um registro a ser realocado (*celula.mtab > 0*). Nesse caso, abrimos o arquivo de nome *indice_celula* e criamos um arquivo *[indice_celula]aloc.dat* de tamanho *celula.mtab*celula.mtab* registros.

[indice_celula]aloc.dat é iniciado contendo apenas registros vazios (*ocupado = 0*). Após a inicialização, iteramos sobre o arquivo *indice_celula* e, para cada registro, tentamos aloca-lo em *[indice_celula]aloc.dat* em função da sua posição hash utilizando os valores de *celula_indice_i.a*, *celula_indice_i.b*, o primo *p*, *celula_i.mtab * celula_i.mtab* e a chave do registro.

Se um registro for alocado para uma posição já ocupada (checado a partir dos valores de *ocupado* dos registros), os valores *celula_indice_i.a* e *celula_indice_i.b* são atualizados aleatoriamente e o processo de alocação reinicia.

Uma vez que os registros dos arquivos *[ìndice_célula].dat* forem alocados, os referidos arquivos são excluídos, uma vez que são usados exclusivamente para a gerência dos registros antes de sua alocação por hash de nível dois.

Tomando como referência a Figura 2, as tabelas de segundo nível seriam representadas nos arquivos *0aloc.dat*, *2aloc.dat*, *5aloc.dat* e *7aloc.dat*.

Estruturação do Código e Compilação.

Para melhor organização do código, separamos nossa implementação em multiplos arquivos:

main.c: Contém a lógica de entrada basica e chama as outras funções a depender da operação escolhida.

base.c: Contém a função de criação do arquivo *nivelUm.dat* e a função de hash usada em todo o programa.
base.h: Contém a estrutura da célula e as declarações das funções implementadas em *base.c*.

registros.c: Contém as funções de inserção, realocação e consulta de registros.
registros.h: Contém a estrutura dos dados e do registro e as declarações das funções implementadas em *registros.c*.

impressao.c: Contém as funções de impressão do nível um, impressão por índice do nível dois e impressão de todos os índices de tamanho maior que 0 do nível dois.
impressao.h: Contém as declarações das funções implementadas em *impressao.c*.

Para compilar nosso código, utilizamos o *gcc* na versão *gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0* e executamos o seguinte comando: `gcc main.c base.c impressao.c registros.c`