Universidade federal do Mato Grosso-UFMT

Estrutura de Dados II

Guilber Domingues Leal

Arthur Gonçalves Soares

**Trabalho 2**

**Tabela Hash**

2021

**Introdução**

O objetivo desse trabalho é comparar duas funções hash para três arquivos de diferentes tamanhos (50, 100 e 150) e alguns com restrição sendo par e outros sem restrições.

Contém dois algoritmos, o primeiro para gerar os arquivos que contém números e letras diferentes e o segundo para manipular e comparar a tabela Hash.

Para realização dos testes foram utilizadas duas funções hash. A primeira chamada de “HASH1” utiliza o método da divisão para mapear onde o registro será colocado.

A segunda função é chamada de “HASH2” e utiliza do método da multiplicação, onde nesse exemplo foi utilizado a função (chave \* 0.09) para mapeamento de onde alocar o registro.

Foram utilizados três tipos de tabelas contendo chaves de inteiros até 1023 e um valor char[3]. Os três tipos são parecidos, mudando basicamente a quantidade de valores inseridos neles, que são 50, 100 e 150 respectivamente.

Os arquivos se encontram em anexo com este relatório.

De acordo com os testes as funções que obtiveram a menor média de colisões foram bem variadas. Comparando a função “HASH1” com as tabelas de 150 registros, percebemos que, em função da “HASH2” a média de colisões foi menor, tanto nas chaves pares quanto nas sem restrições.

Analisando dessa forma chegamos à conclusão de que a função “HASH2” é mais eficiente para armazenamento de registros de maior quantidade, gerando uma diferença grande quando comparado com um número maior de registros.

|  |  |  |  |
| --- | --- | --- | --- |
| Usando a função Hash TIPO 1, essa é média de colisões: | | | |
| REGISTROS | 50 | 100 | 150 |
| PAR | 12,8 | 30,2 | 36 |
| Aleatório | 9 | 26 | 40,6 |

|  |  |  |  |
| --- | --- | --- | --- |
| Usando a função Hash TIPO 2, essa é média de colisões: | | | |
| REGISTROS | 50 | 100 | 150 |
| PAR | 2.2 | 5,6 | 5,4 |
| Aleatório | 2,6 | 4,2 | 6 |

**Desenvolvimento**

No primeiro algoritmo que gera números aleatórios diferentes e a chave de caracteres diferentes, foi utilizado a função srand(time(NULL)) que inicializa o gerador, garantindo que a cada execução o valor será diferente e depois é gravado esses valores em arquivos com o nome tamanho dos valores(50, 100 e 150), se é \_Ale\_ (sem restrições) e \_Par\_(só valores pares) que vai de 1 até 5 arquivos gerados.

No segundo algoritmo foi implementado as funções de manipulação da estrutura: insere, inicializa, imprime, lê arquivos, colisões (que calcula o numero de colisões), libera e aloca.

Foi utilizado duas funções para comparação de desempenho, a primeira:

int hash(int mat, int tam){ return mat%tam; }

É o método da divisão, consiste em pegar o resto da divisão entre a chave (mat) e o tamanho da estrutura (tam).

A segunda Função:

int hashDois(int mat, int tam){ return floor( tam\*((0.61\*mat)mod 1)) }

É o método da multiplicação, consiste em pegar o piso (floor) da expressão do tamanho da matriz (tam) multiplicado por 0.61(é uma constante entre 0 e 1, esse valor 0.61 é sugerido por Knuth que vem do resultado da função ((√5-1)/2)), multiplicado pela chave (mat) e o resto da divisão por 1.