AEDS 1: Tabela Hash

Hércules Teixeira 18.2.8072

[1] Universidade Federal de Ouro Preto - Campus ICEA

Resumo. *O objetivo deste trabalho é analisar a estrutura de dados Hash com encadeamento separado (separete chaining).*

1. Introdução

O trabalho foi dividido nas seguintes partes: Leitura de dados, implementação uma tabela hash com encadeamento separado, análise dos dados obtidos e execução por parâmetros obtidos a partir da linha de comando.

2. Leitura e armazenamento dos dados

Nos foi dado um arquivo TXT com o rendimento de 30228 escolas no Enem, o qual contém os seguintes tópicos: ”id, estado, município, rede, media ciências da natureza, media ciências humanas, media linguagem, media matemática e media redação. Após ser feito a leitura dos dados deve-se armazena-los em uma tabela hash.

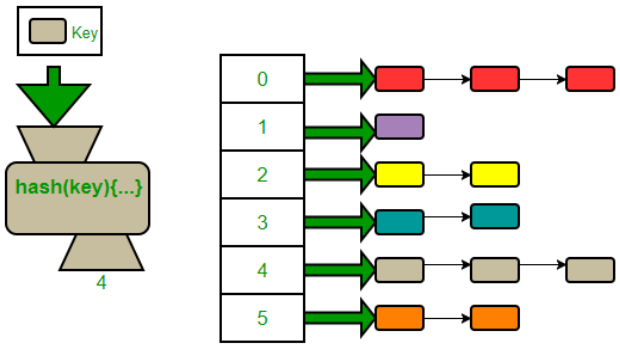

Figura 1. Tabela hash

O funcionamento de tal algoritmo consiste em um array de tamanho M (valor passado por linha de comando) no qual cada posição do mesmo contém um ponteiro para uma lista ligada; e para definir a posição do array no qual os dados vão ser armazenados é utilizado uma função hash-code que retorna o resto da divisão do ID pelo N (valor passado por linha de comando). Após ser calculado a posição correta deve ser testado se já existe um conjunto de dados com mesmo ID na lista que está ligada a essa posição.

3. Execução

O seu programa deverá imprimir informações na seguinte ordem: número de slots (espaços) vazios na Tabela Hash; tamanho médio das listas nos slots, isto é, soma do tamanho das listas pelo número possível de listas; número de colisões; tempo para inserir todos os dados.

O programa deve ser executado a partir de parâmetros passados pela linha de comando. Os parâmetros do programa foram definidos assim:

• -b: Este comando executará a busca pelos IDs presentes no arquivo que deve ser digitado logo após esse comando. Caso não contenha esse comando deverá ser digitado um ID em especifico;

• -d: Este comando indica que a seguir será passado um endereço de um arquivo, e a partir dele salvar os dados em uma tabela hash, onde depois será executado a busca seja a partir de um ID ou de um arquivo contendo IDs;

• -M: Este comando indica que a seguir será passado um número inteiro que será utilizado para definir o tamanho da tabela hash a ser implementada;

• -mod: Este comando indica que a seguir será passado um número inteiro que será utilizado na função hash-code.

4. Análise dos dados

A seguir segue a tabela com os valores dos testes com o valor de M = NUM.

M = NUM	1	30	302	3022	15114	22671	30228
Slots vazios	0	0	0	0	0	2	2
Tamanho médio das listas	30288	1007,6	100,09	10	2	1,33	1
Número de colisoes	30227	30198	29926	27206	15114	7559	2
Tempo para inserir	4,777 s	0,405 s	0,123 s	0,097 s	0,096 s	0,093 s	0,089 s
Qui-quadradro	1	0,999043	0,990247	0,916743	0,750045	0,75005	0,666718

Figura 2. Dados obtidos

Podemos observar que esse é o melhor caso dessa tabela, pois se o NUM for menor do que o valor de M, o número de slots vazios será proporcional à diferença entre os dois valores; e isso acarretaria em um desperdício de memória.

Analisando os dados podemos observar os seguintes fatos: se o valor de M for muito baixo, o valor do teste qui-quadrado se aproxima muito dos valores ideias, porém o seu tempo de inserção e de busca na tabela se aproxima muito próximo à O(n). Já quando temos o inverso, ou seja, um valor de M muito alto, podemos observar que o teste qui-quadrado fica distante dos valores ideais, e o tempo de busca e de inserção fica próximo à O(1), porem temos um alto custo de memória.

Após a análise dos dados podemos observar que existe um fator de relação entre o tamanho da tabela hash e o número de elementos inseridos para que a busca seja eficiente, com isso podemos dizer que os melhores casos são quando M se aproxima aos 10 por cento do valor total de elementos inseridos.

5. Estudo de complexidade

	Complexidade
printNode	O(1)
createNode	O(1)
searchInList	O(n)
hashCode	O(1)
insertToHash	O(1)
searInHash	O(1)
display	O(1)
slotsVazios	O(n)
introduzirDados	O(n)
inicio	O(1)
tamMedListas	O(n)

Figura 3. Análise de complexidade

No geral conclui-se que o programa completo tem uma complexidade de O(n).

6. Conclusão

Com esse trabalho prático pode-se obter vários conhecimentos sobre o assunto, dos quais a maioria se obtém apenas na prática. Um dos pontos a ser destacado é a reutilização de código; e isso traz várias barreiras, como o fato de entender toda a mecânica do algoritmo já pronto e saber lidar com isso.

Além de ter um conhecimento melhor sobre a utilização da tabela hash; já que depois de termos analisado vários dados agora temos um conhecimento melhor de suas vantagens.

Referências

OVERLEAF. Overleaf: Online latex editor, 2019. Disponível em: https://www.overleaf.com/. Acesso em: 28 de nov. de 2019.

GOOGLE. Goggle planilhas, 2019. Disponível em: https://docs.google.com/spreadsheets/. Acesso em: 28 de nov. de 2019

LUCIDCHART. Lucidchart: Software online de diagramas e comunicação visual, 2019. Disponível em: https://www.lucidchart.com/. Acesso em: 28 de nov. de 2019

GEEKS FOR GEEKS. Geeks for Geeks: a computer science portal for geeks, 2019. Disponível em: https://www.geeksforgeeks.org/. Acesso em: 28 de nov. de 2019.

STACKOVERFLOW. Stack Overflow: Where developers learn, share, and build carrers , 2019. Disponível em: https://stackoverflow.com/. Acesso em: 28 de nov. de 2019.