Pontifícia Universidade Católica de Goiás
Escola de Ciências Exatas e da Computação
CMP1099 - Estrutura de Dados II
Max Gontijo de Oliveira
1º Desafio - Tabela Hash - Dispersão e Colisões

1 Resumo

O professor irá disponibilizar um arquivo contendo uma quantidade determinada de palavras distintas e o aluno deverá inserir todas essas palavras como chaves em uma tabela hash. O vetor dessa tabela deverá ter um tamanho determinado pelo professor para o conjunto de palavras a ser inserido. Com base nisso, dois desafios são lançados: melhor dispersão das chaves ao final das inserções; e menor quantidade de colisões.

Cada desafio será explicado melhor adiante.

Desde já, fica estabelecido que a tabela hash deverá ser implementada com endereçamento aberto, ou seja, deverá tratar as colisões utilizando sondagem.

2 Desafios

O aluno poderá decidir ser deseja participar de apenas um dos desafios ou de ambos. Para qualquer um dos desafios, o aluno deverá ser capaz de construir uma aplicação que seja capaz de ler o arquivo de entrada contendo as palavras e inserir essas palavras em uma tabela hash criada pelo aluno com o tamanho do vetor determinado no próprio arquivo. Em ambos os desafios, o objeto de disputa será a função hash.

2.1 Melhor dispersão das chaves

Uma das evidências que pode ser observada em uma tabela hash com uma boa função hash é justamente a dispersão das chaves pelo vetor. Nesse desafio, o objetivo é obter a melhor dispersão possível das chaves do arquivo de entrada inseridas na tabela. Após a inserção de todas as chaves do arquivo, a dispersão da tabela pode ser analisada pelos segmentos contíguos no vetor que tem apenas posições ocupadas por chaves ou apenas posições desocupadas. Como serão diversos segmentos, cada segmento deve ser analisado. Nesse desafio, a forma de análise será pela soma dos quadrados dos tamanhos de todos esses segmentos, de modo que quanto menor for a soma total, melhor será considerada a dispersão das chaves. Assim, após a inserção de todas as chaves, o valor que representará a qualidade q da dispersão realizada é dada por:

$$q = \sum_{i=1}^{m} d_i^2$$

onde m é a quantidade de segmentos e d_i é o tamanho (quantidade de posições no vetor) de cada segmento.

Por exemplo, suponha que seis chaves A, B, C, D, E e , F sejam inseridas em uma tabela hash com vetor de tamanho 13. Suponha que após a inserção dessas chaves, os registros estejam distribuídos dessa forma:

A	F	C			B				E	D		
0	1	2	3	4	5	6	7	8	9	10	11	12

Nesse vetor, podemos identificar seis segmentos:

- Da posição 0 à 2, temos um segmento de tamanho 3 de posições ocupadas no vetor
- Da posição 3 à 4, temos um segmento de tamanho 2 de posições desocupadas no vetor
- Da posição 5 à 5, temos um segmento de tamanho 1 de posições ocupadas no vetor
- Da posição 6 à 8, temos um segmento de tamanho 3 de posições desocupadas no vetor
- Da posição 9 à 10, temos um segmento de tamanho 2 de posições ocupadas no vetor
- Da posição 11 à 12, temos um segmento de tamanho 2 de posições desocupadas no vetor

Assim, temos que:

$$q = 3^2 + 2^2 + 1^2 + 3^2 + 2^2 + 2^2 = 31$$

Uma função hash diferente poderia ser utilizada no mesmo conjunto de entrada e gerar uma nova dispersão, como por exemplo:

		B	C			A	F				E	D
0	1	2	3	4	5	6	7	8	9	10	11	12

Nesse exemplo, por acaso, novamente temos seis segmentos, de modo que a qualidade da dispersão é dada por:

$$q = 2^2 + 2^2 + 2^2 + 2^2 + 3^2 + 2^2 = 29$$

Portanto, essa segunda dispersão é melhor do que a primeira.
Uma outra função hash poderia ser utilizada e gerar uma dispersão ainda mais diferente:

	E		C		D		B		A	F		
0	1	2	3	4	5	6	7	8	9	10	11	12

Nesse novo exemplo temos 11 segmentos, de modo que a qualidade da dispersão é dada por:

$$q = 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 2^2 + 2^2 = 17$$

Como podemos ver, quanto mais dispersas as chaves, menor é o valor da soma, o que indica uma melhor dispersão.

2.2 Menor quantidade de colisões

O segundo desafio se trata da simples minimização da quantidade de colisões que as inserções irão gerar. Esse desafio não está necessariamente ligado ao desafio apresentado na Seção 2.1. O objetivo aqui é analisar a complexidade de inserção de chaves na tabela. Na contagem final, deverão ser contabilizados cada cálculo adicional de posição realizado por conta de uma colisão durante a inserção.

Vamos tomar um exemplo. Considere a inserção das chaves 32, 7, 15, 28 e 9 em uma tabela hash. Considere que tais chaves sejam mapeadas da seguinte maneira pela função hash:

- Chave 32 → 8
- Chave 7 → 45
- Chave 15 → 8 (colisão) → 90
- Chave 28 → 17
- Chave 9 → 90 (colisão) → 45 (colisão) → 29

Para esse conjunto de inserções, seriam contabilizadas três colisões: uma com a chave 15 e duas com a chave 9.

3 Como participar do desafio

Para participar do desafio, o aluno deverá simplesmente enviar ao e-mail do professor até a data limite, a função hash utilizada e um número que representa o indicador de qualidade da dispersão gerada pela função sobre o arquivo de entrada fornecido (desafio descrito na Seção 2.1) e/ou o número de colisões gerada após a inserção das chaves do dito arquivo (desafio descrito na Seção 2.2).

Será considerado o vencedor do desafio aquele aluno que fornecer a melhor função hash para o primeiro desafio ou para o segundo desafio. Caso hajam alunos que empatem, será considerado o vencedor aquele que enviou a solução antes.

Observação: O professor irá rodar uma aplicação utilizando a função do aluno vencedor para conferir se essa função realmente gera o valor do indicador de qualidade de dispersão (primeiro desafio) ou número colisões (segundo desafio) informado pelo aluno vencedor. Caso o resultado não se repita, o aluno é desclassificado. Portanto, é fundamental que a função hash seja determinística. Em outras palavras, a função hash não poderá ter nenhum fator randômico.

O aluno pode participar de um dos desafios ou de ambos. Para participar apenas do primeiro, deve enviar a função hash e o indicador de qualidade, cujo cálculo foi descrito na Seção 2.1. Para participar apenas do segundo, deve enviar a função hash e o número de colisões ocorridas ao inserir as chaves do arquivo. Para participar de ambos, precisa enviar a função hash e as informações já citadas.

3.1 Leitura do arquivo e inserção das chaves na tabela

O arquivo fornecido pelo professor trata-se de um arquivo de texto e terá a seguinte estrutura:

- A primeira linha é um número e se refere ao tamanho que o vetor da tabela hash deve ter.
- A segunda linha é outro número e indica apenas quantas palavras (chaves) existem naquele arquivo.
- As linhas seguintes são palavras (chaves) compostas de caracteres alfanuméricos que devem ser inseridas na tabela hash.

Nenhuma palavra no arquivo se repete, de modo que, ao final das inserções, a tabela hash deverá conter exatamente a quantidade de chaves indicadas na segunda linha do arquivo.

3.2 Função hash

A função hash que o aluno deverá apresentar ao professor precisa NECESSARIAMENTE obedecer à seguinte assinatura:

```
INTEIRO hash(TEXTO k, INTEIRO i)
```

onde k se refere à uma chave e i ao contador de sondagem, que será 0 (zero) na primeira tentativa de inserção de cada chave. A função deve retornar a posição mapeada para aquela chave k na tentativa de sondagem i.

Em C++, por exemplo, essa função ficaria assim:

```
int hash(string k, int i)
```

ou assim:

```
int hash(char* k, int i)
```

Em Java, o método ficaria assim:

```
int hash(String k, int i)
```

Em Octave/MATLAB, a função ficaria assim:

```
function [pos] = hash(k, i)
```

Essa assinatura é obrigatória e necessária para que o professor possa realizar os testes necessários e confirmar as informações prestadas pelo aluno sobre a qualidade da dispersão ou número de colisões.

Obviamente, o aluno pode usar outras funções internamente. Mas deverá provê-las também.

Nesse trabalho, a conversão da chave k em um número inteiro faz parte do processo de mapeamento da chave para uma posição.

4 Premiação

Poderão haver até dois vencedores, um para cada desafio. Entretanto, se um mesmo aluno participar dos dois desafios e vencer ambos, haverá somente um vencedor.

Cada vencedor irá receber uma nota extra combinada em sala de aula. Se um mesmo aluno vencer os dois desafios, a nota extra atribuída será relativa apenas a de um desafio.