Estrutura Avançada de Dados - Aula 4

Aluno: Klaus Becker

Exercício 1

Qual a complexidade de pior caso e de melhor caso para a inserção em HashTable com separate chaining?

Resposta

A complexidade do pior caso para a inserção em HashTable com separate chaining é O(n), onde todos os elementos são inseridos no mesmo índice da tabela hash, formando uma lista encadeada. Já a complexidade do melhor caso é O(1), onde temos cada elemento inserido em um índice diferente da tabela hash, sendo possível acessar diretamente o elemento sem a necessidade de percorrer a lista encadeada.

Exercício 2

Qual a complexidade de pior caso e de melhor caso para deleção em em HashTable com separate chaining?

Resposta

Assim como na inserção, a complexidade do pior caso para a deleção em HashTable com separate chaining é O(n), onde o elemento a ser deletado está no final do bucket, sendo necessário percorrer toda a lista encadeada. Já a complexidade do melhor caso é O(1), onde o elemento a ser deletado está no início do bucket, sendo possível acessar diretamente o elemento sem a necessidade de percorrer a lista encadeada.

Exercício 3

Qual a principal desvantagem da HashFunction de divisão?

Resposta

Ao utilizar uma HashFunction, é importante que ela distribua os elementos de forma uniforme na tabela hash, evitando colisões. A principal desvantagem da HashFunction de divisão é que ela pode gerar colisões com frequência, tendo em vista que ela utiliza o resto da divisão para calcular o índice da tabela hash, o que pode resultar em vários elementos sendo inseridos no mesmo índice, formando uma lista encadeada e prejudicando o desempenho da estrutura de dados.

Exercício 4

Descreva, pelo menos, 3 HashFunctions distintas e dê exemplos de suas vantagens e desvantagens.

Resposta

1. **Método da divisão**: como descrito na questão anterior, a HashFunction de divisão utiliza o resto da divisão para calcular o índice da tabela hash. Uma de suas vantagens é a simplicidade de

implementação, porém, como desvantagem, pode gerar colisões com frequência, prejudicando o desempenho da estrutura de dados.

2. **Método da multiplicação**: essa HashFunction utiliza-se de uma constante *A* (um valor entre 0 e 1) e de uma constante *m* (quantidade de buckets) para calcular o índice da tabela hash. A fórmula utilizada é *h(k) = floor(m * (k * A mod 1))*. A sua principal vantagem é a distribuição uniforme dos elementos na tabela hash, reduzindo as colisões. Entretanto, a desvantagem é que a escolha de *A* pode ser um desafio, visto que um valor inadequado pode resultar em um maior número de colisões.

3. **Método por dobramento**: nesse método, a chave é dividida em partes menores e somadas ou aplicadas a operação de XOR para gerar o índice da tabela hash. Por exemplo, se a chave for 123456, ela pode ser dividida em 12 e 34, e a soma desses valores (12 + 34 = 46) seria utilizado para calcular o índice da tabela. Esse modelo tem como vantagem o ótimo funcionamento com chaves númericas grandes, como CPFs e CNPJs. No entanto, a desvantagem é que ele pode gerar colisões em chaves que possuem partes iguais.