Lista 9 – Tabelas Hash

ESTRUTURA DE DADOS I – Pedro Nuno Moura
Monitor: Celio Ferreira Camara Junior

Para as questões abaixo, considere as implementações de Tabela Hash vistas em sala de aula e disponíveis no Moodle.

1) Insira as chaves E A S Y Q U T I O N, nessa ordem, em uma tabela inicial vazia de $M = 5$ listas, utilizando o método *Separate Chaining*. Utilize uma função de hash 11 * k % M para transformar a k-ésima letra do alfabeto em um índice da tabela.

2) Desenvolva um programa que encontre os valores de a e M, com M tão pequeno quanto possível, de tal forma que a função (a * k) % M para transformar a k-ésima letra do alfabeto em um índice da tabela produza valores distintos para cada k (isto é, em que não ocorram colisões) para as chaves S E A R C H X M P L. Uma função que possua tal propriedade é conhecida como **função de hash perfeita**.

3) A seguinte implementação do método `hashCode()` é legal (permitida na linguagem Java)?

```
public int hashCode()
{   return 17;   }
```

Caso seja legal, descreva o efeito de utilizá-la; caso contrário, explique por quê.

4) Mostre o conteúdo de uma tabela hash que segue a abordagem linear-probing quando se inserem as chaves E A S Y Q U T I O N, nessa ordem, em uma tabela inicialmente vazia de tamanho inicial $M = 4$ que é expandida pelo dobro do seu tamanho quando está cheia pela metade. Utilize a função de hash (11 * k) % M para transformar a k-ésima letra do alfabeto em um índice da tabela.

5) Modifique a classe `SeparateChainingHashST` para usar uma segunda função de hash e escolher a menor das duas listas obtidas ao se aplicarem ambas as funções para uma determinada chave. Mostre o processo de inserir as chaves E A S Y Q U T I O N, nessa ordem, em uma tabela inicialmente vazia de tamanho $M = 3$, usando a função (11 * k) % M como a primeira função de hash e a função (17 * k) % M como a segunda função de hash, ambas para a k-ésima letra. Forneça o número médio de comparações ao realizar uma busca bem sucedida.

6) Modifique a classe `SeparateChainingHashST` para usar uma árvore rubro-negra em vez de uma lista encadeada para armazenar os elementos. Realize experimentos que demonstrem claramente os prós e contras dessa mudança (por exemplo: calcular quantidade de chaves inseridas, tempo médio de inserção, tempo médio de deleção e tempo médio de busca).