ED2 2025/1: Exercício sobre Dicionários (parte 1)
Estrutura de Dados II
Professor(es): Abrantes Araújo Silva Filho Entrega: 2025-05-13 10:55

As questões e exercícios de programação a seguir foram retiradas do livro "*Programming Abstractions in C: A Second Course in Computer Science*", de Eric S. Roberts (Addison-Wesley, 1998), Capítulo 11: *Symbol Tables*.

1 Questões de Revisão

Responda às questões abaixo, de forma **manuscrita**, em papel almaço. Entregue diretamente ao professor na data indicada.

(a) O que é o TAD Dicionário? Quais suas características?

(b) Sabe-se que os dois TADs mais comuns do tipo "dicionário" são o *dictionary* e a *symbol table*. Descreva as diferenças e as semelhanças entre eles.

(c) O que a **chave** representa no contexto de uma *symbol table*?

(d) Quais são os tipos de dados mais comumente utilizados para representar a **chave** e o **valor** em um TAD Dicionário?

(e) Pode-se afirmar que, se usarmos arrays ordenados pelas chaves para implementar um dicionário, um algoritmo de busca binária nos permitiria implementar as operações de busca e inserção em tempo $O(\log n)$? Por quê?

(f) Por que usar o valor ASCII da primeira letra da chave (que é uma string) como *hash code* é uma abordagem ruim?

(g) O que é uma *hash table*? O que é um *bucket*? O que é uma colisão?

(h) O que é uma *hash function*? O que significa *hashing*?

(i) Você implementou a seguinte função *hash*, que recebe a **chave** e a quantidade de *buckets* da *hash table*, e utiliza a função `inteiro_aleatorio` para retornar o *hash code*:

```
int hash (string chave, int buckets)
{
    return (inteiro_aleatorio(0, buckets - 1));
}
```

Essa implementação está correta ou errada? Por quê?

(j) Para evitar colisões, por vezes fazemos um *tradeoff* entre o tempo de acesso e o espaço utilizado por uma *hash table*. Explique essa compensação entre tempo-espaço.