UNIVERSIDADE FEDERAL DO CEARÁ

Lista 3 – Estrutura de Dados 2018.2

1 – Considere um conjunto c de n elementos inteiros positivos tal que c = {1, 2, 3, …, n} e uma uma Tabela Hash de tamanho m. Onde n e m devem ser dados pelo usuário. Implemente uma função hash de sua escolha. Apresente os seguintes métodos para sua Tabela Hash:

a) Adicionar um elemento na hash;
b) Buscar um elemento na hash
c) Remover um elemento da hash;
d) Sua implemetações tem muitas colisões? Justifique os motivos.

2 - Faça um método de dispersão (hash) que tenha como chave um string com os nomes de países, de até 32 caracteres. A tabela hash deve ter 513 elementos.

3 - Considere que você tenha sido contratado para construir um sistema para reconhecimento de estradas entre um conjunto de cidades. Ao todo serão **n** cidades que estão interligadas por uma série de estradas. Seu sistema inicialmente deve responder a questões simples, como:

(a) Dado uma cidade **k** qualquer, determinar quantas estradas saem e quantas chegam à cidade **k**.
(b) Qual das cidades chega o maior número de estradas?
(c) Dado **k**, verificar se todas as estradas diretas entre a cidade **k** e outras são de mão dupla.
(d) Relacionar as cidades que possuem estradas diretas para a cidade **k**.

(d) Qual estrutura de dados você utilizaria para criar este sistema. Por que?
(e) Dado sua escolha, dê um pequeno exemplo (no mínimo 5 cidades e 10 estradas) e descreva como seria a solução de cada pergunta acima efetuando operações sobre a estrutura de dados escolhida.

4 - O professor *inteligentus* afirmou para seus alunos que se você tem uma árvore com 687 nós **não-terminais**, tal que cada um destes nós tem exatamente **dois descentes** então está árvore possui 1374 arestas. Você concorda com o professor inteligentus? Se sim, **prove** o porque. Senão dê um **contra-exemplo**.

5 - Quantos *antecedentes* tem um nó no nível n em uma **árvore binária**?

6 - Escreva algoritmos recursivos e não-recursivos para determinar:

a) O número de nós em uma **árvore binária.**
b) A soma dos conteúdos de todos os nós em uma árvore binária, considerando que cada nó contém um inteiro.
c) A profundidade de uma **árvore binária**.

7 - Represente a sequência abaixo na forma de **árvores binárias** de alturas mínima e máxima.

S = { 3, 5, 9, 12, 14, 6, 7, 15 }

8- Dados os percursos abaixo, reconstruir a árvore original:
pré-ordem: 1, 2, 3, 6, 8, 4, 9, 10, 12, 11, 5, 7, I
simétrica (in-ordem): 6, 3, 8, 2, 4, 9, 12, 10, 11, 1, I, 7, 5

9 - Suponha que por causa da sua aplicação, os nós de uma **árvore binária de busca** agora também deverão armazenar o **nó pai de cada nó**. Efetue as modificações necessárias no arquivo no.h e na função inserir para que esse dado seja armazenado.

10 - Escreva uma função que receba como parâmetro uma chave, percorra uma **árvore binária de busca** e retorne o nível que se encontra o nó com está chave (caso ela exista).

11 - Desenvolva uma função que imprima (***In Ordem***) apenas as chaves das ***folhas*** de uma **árvore binária de busca**.

12 - Dois algoritmos A e B possuem *complexidade* n^5 e 2^n , respectivamente. Você utilizaria o algoritmo B ao invés do A. Em qual caso? Exemplifique.

13 – Um algoritmo A utiliza 100n operações enquanto o algoritmo B usa $3n^2$ operações. Determine c e n para o qual A é melhor do que B.

14 - Ordene as funções a seguir pela ordem de complexidade: n^2 , n , log n , 2^n , n log n , n^3

15 - (a) Considere A um arranjo (array) ordenado de n número inteiros. Desenvolva um algoritmo que receba A e um número inteiro n e retorne a posição do arranjo em que n se encontra
(b) Qual o número de operações do seu algoritmo no pior caso? E no melhor?
(c) Qual a ordem de complexidade do algoritmo? Este algoritmo é bom?

16 – Análise os algoritmos utilizando notação de ordem O:

```
Algoritmo A(a,n):
        Entrada dois inteiros, a e n
        Saída: ?
  k ← 0
  b ← 1
  enquanto (k < n) faça
     k ← k +1
```

```
Algoritmo A(a,n):
        Entrada dois inteiros, a e n
        Saída: ?
 k ← n
 b ← 1
 c ← a
 enquanto (k > 0) faça
   se (k % 2 == 0) então
     k ← k / 2
     c ← c * c
     b ← b * c
 retorne b
```