Projeto e Análise de Algoritmos
Exercícios Resolvidos

1) Hoje, dos algoritmo de ordenação mais utilizados, o que possui menor complexidade é o inserção, considerando o seu melhor caso, O(n). Podemos dizer que nenhum outro algoritmo poderá atingir uma complexidade melhor do que esta? Justifique.

2) 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.

3) Considerando, que as chaves inseridas em um Hashing não provocaram colisão. Independente do algoritmo, podemos dizer que a pesquisa por uma chave na tabela será de ordem de complexidade constante, ou seja, O(1)? Podemos dizer que este algoritmo é ótimo? Justigique.

4) Para duas funções g(n) e f(n) temos que f(n)=”teta”(n) se somente se f(n)= Ω(g(n)) e f(n)=O(g(n)). Explique o teorema acima.

5) Podemos definir o seguinte algoritmo para calcular a ordem de complexidade de algoritmos não recursivos:
1- Escolher o parâmetro que indica o tamanho da entrada
2- Identificar a operação básica (comparação, atribuição)
3- Estabeleça uma soma que indique quantas vezes sua operação básica foi executada (pio caso)
4- Utilize regras para manipulação de soma e fórmulas definindo uma função de complexidade
5- Encontre a ordem de complexidade

a) Baseando-se no algoritmo acima determine a ordem de complexidade do algoritmo abaixo:

```
MaxMin(vetor v)
        max=v[1];
        min=v[1];
        para i=2 até n faça
                se v[1]> max então max=v[1]; fimse
                se v[1]< min então  min=v[1]; fimse
        fimpara;
fim.
```

b) Podemos dizer que o algoritmo acima é $O(n^2)$? Justifique.

6) Uma outra métrica muito utilizada para avaliar algoritmos é a Métrica Empírica. Essa consiste em escolher uma métrica (tempo , número de instruções executadas, etc), propor entradas diferenciadas (geralmente com alguma característica pré-definida), ou seja, amostras. Finalmente executar o algoritmo com as entradas e analisar os resultados. Esta medida é muito utilizada para comparar dois algoritmos. Critique a métrica.

7) Considere o problema de inserir um novo elemento em um conjunto

ordenado de dados: a1> a2>a3>....>an. Apresente um limite inferior para este problema e exemplifique.

8) Por muitas vezes damos atenção apenas ao pior caso dos algoritmos. Explique o porque.

9) Podemos dizer que uma um algoritmo com complexidade f(n) =O(f(n/2))? Justifique.

Respostas

1) Sim. Para resolver o problema é necessário que todos os valores pertencentes a entrada sejam avaliados, ou seja, podemos dizer que qualquer algoritmo proposto será no mínimo Ω(n).

2) Sim. Apesar do algoritmo ser exponencial, quando o valor de n é pequeno, esta função produz um tempo de complexidade menor do que a função do algoritmo A. Por exemplo para valores de n iguais 2 ... 10 20

3)Sim. O algoritmo é ótimo, pois o acesso à chave é direto.

4) A notação teta indica que a função f(n) está entre um limite superior e um limite inferior. Se f(n) é teta de g(n) que dizer que f(n) é tem limite superior em g(n) se somente se f(n)<= g(n)* c1, onde c1 é uma constante qualquer para n>m, ou seja f(n)= O(g(n)). O mesmo é válido para notação Ω.

5) A escolha do parâmetro é n, que indica a quantidade de elementos do vetor. A operação básica é a comparação, pois esta domina a o loop interno e a complexidade é dada por 3(n-1). O algoritmo é da ordem de complexidade O(n). Como n é $O(n^2)$, podemos dizer que o algoritmo é $O(n^2)$, entretanto estamos interessados sempre no limite mínimo superior, logo é melhor dizer que o algoritmo é O(n).

6) O problema desta medida é a definição da amostra e como serão avaliados os parâmetros. Por exemplo, se utilizarmos a função time para calcular o tempo, podemos obter valores diferentes para sistemas operacionais diferentes. Ainda, deve-se pensar que o cálculo do tempo não pode incluir o tempo em que o processo não esteve escalonado. Porém, este método é muito interessante quando se deseja comparar dois ou mais algoritmo, quando se deseja saber o comportamento do algoritmo para determinadas bases de dados.

7) Utilizando a estrutura de árvores binária balanceadas podemos incluir qualquer elemento a uma complexidade logn +1.

8) Porque normalmente desejamos saber qual o limite máximo gasto para executar o um determinado algoritmo.