

## ALGORITMOS II

### 6ª LISTA DE EXERCÍCIOS COMPLEXIDADE DE ALGORITMOS

- 1 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.
- 2 Suponha dois algoritmos A e B com funções de complexidade de tempo dadas por  $a(n) = n^2 - n + 549$  e  $b(n) = 49n + 49$  respectivamente. Determine quais valores de  $n$  no conjunto dos números naturais para os quais A leva menos tempo para executar que B.
- 3 Ordene as seguintes funções em termos da taxa de crescimento (complexidade), considerando o melhor e o pior caso.

$n$ ,  $\sqrt{n}$ ,  $\log(n)$ ,  $\log(\log(n))$ ,  $\log(n)^2$ ,  $n/\log(n)$ ,  $\sqrt{n}\log(n)^2$ ,  $(1/3)n$ ,  $(3/2)n$ , 17.

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

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

```
int MaxMin(int n, int *v) {  
    max = v[0];  
    min = v[0];  
    for (int i=1; i<n; i++) {  
        if (v[i] > max) max=v[i];  
        if (v[i] < min) min=v[i];  
    }  
}
```

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

- 5 Por muitas vezes damos atenção apenas ao pior caso dos algoritmos. Explique o porque.
- 6 Perdido em uma terra muito distante, você se encontra em frente a um muro de comprimento infinito para os dois lados (esquerda e direita). Em meio a uma escuridão total, você carrega um lampião que lhe possibilita ver apenas a porção do muro que se encontra exatamente à sua frente (o campo de visão que o lampião lhe proporciona equivale exatamente ao tamanho de um passo seu). Existe uma porta no muro que você deseja atravessar. Supondo que a mesma esteja a  $n$  passos de sua posição inicial (não se sabe se à direita ou à esquerda), elabore um algoritmo para caminhar ao longo do muro que encontre a porta em  $O(n)$  passos. Considere que  $n$  é um valor desconhecido (informação pertencente à instância). Considere que a ação composta por dar um passo e verificar a posição do muro correspondente custa

$O(1)$ .

7 Analise a complexidade do algoritmo abaixo:

```
int x = 0;
for (int i = 1; i <= n ; i++)
    for (int j = i+1; j <= n; j++)
        for (int k=1; k <= j-1; k++)
            x++;
```

8 Dois programas A e B foram analisados e os respectivos limites de pior caso foram determinados como sendo  $150n \cdot \log(n)$  e  $n^2$ . Se possível, responda às seguintes perguntas:

- Qual dos programas tem melhor garantia de desempenho para valores grandes de  $n$  ( $n > 10000$ )?
- Qual dos programas tem melhor garantia de desempenho para valores pequenos de  $n$  ( $n < 100$ )?
- Qual programa executará mais rápido na media para  $n = 1000$ ?
- É possível ao programa B executar mais rápido que A para todas as entradas possíveis?

9 Apresente a complexidade de pior caso para os algoritmos recursivos vistos em sala de aula: Torres de Hanói, Sequência de Fibonacci e Função de Ackerman.

10 Desenvolva uma função para determinar se um dado número é primo ou não. A função deve receber o número lido e retornar um valor lógico, indicando se o número é primo ou não.

Lembre-se que:

- Um número é primo se não for divisível por qualquer número exceto um ou ele próprio.
- Um número é divisível por outro se o resto da divisão inteira for zero.

Teste com os seguintes valores:

Entrada	Saída
2	Primo
3	Primo
4	Não-primo
5	Primo
6	Não-primo

Calcule o número de divisões (o módulo é equivalente a uma divisão) que são necessárias para determinar se um número  $n$  é primo através do algoritmo que desenvolveu. Com base nesse número de divisões, qual é a complexidade do algoritmo que desenvolveu?

11 Qual é a ordem de complexidade das seguintes funções (utilize a notação  $O$ ).

- $f(n) = n^2 + 2$
- $g(n) = 503$
- $g(n) = 2 \log n + n$
- $g(n) = 10 \cdot 2^n$
- $f(n) = n \log n + \log n^2$ .

Qual dessas funções possui a maior ordem de complexidade?

12 Arranje as seguintes expressões de acordo com a taxa de crescimento (da menor para a maior):  $4n^2$ ,  $n!$ ,  $\log 3n$ ,  $3^n$ ,  $20n$ ,  $2$ ,  $\log 2n$ .

**13** Determine o pior caso dos seguintes procedimentos em função de n:

```
void Misterio (int n) {
    int i, j, k, l;
    for (i = 1; i <= n-1; i++)
        for (j = i+1; j <= n; j++) {
            for (k = 1; k <= j; k++)
                // Operação de complexidade O(1));
            for (k = 1; k <= sqrt(n); k++)
                // Operação de complexidade O(1));
        }
}

int Recursiva (int n) {
    if (n <= 1)
        return 1;
    return Recursiva(n-1) + Recursiva(n-1);
}
```

OBS.: Para a função *Misterio*, considere dois casos para a complexidade da função *sqrt(x)* (parte inteira da raiz quadrada):

$O(\text{sqrt}(x)) = O(x(0.5))$  – sqrt tem complexidade igual à raiz quadrada

$O(\text{sqrt}(x)) = O(1)$  – sqrt tem complexidade constante.