UNIVERSIDADE DE BRASÍLIA
FACULDADE DO GAMA

CURSO:	**ENGENHARIAS**		
DISCIPLINA:	**Estruturas de Dados e Algoritmos**	**CÓDIGO:**	**193704**
CARGA HORÁRIA:	**60 h**	**CRÉDITOS:**	**04**
PROFESSOR:	**Dr. Nilton Correia da Silva**		

RESOLUÇÃO DE TRABALHO PRÁTICO
TEMA: ANÁLISE DE COMPLEXIDADE DE CÓDIGOS

1. 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[i]> max
                        max=v[i];
                fimse
                se v[i]< min
                        min=v[i];
                fimse
        fimpara;
fim.
```

$\boxed{\emptyset(n)}$

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

Não. As duas primeiras linhas apresentam comandos de atribuição simples ($\emptyset(1)$, cada uma delas). O restante do código é um laço de repetição que tem ordem de complexidade $\emptyset(n)$ pois possui um esforço (quantidade de repetições) proporcional a n. Logo, a complexidade deste código é: $\emptyset(n)$.

3. Considere o problema de inserir um novo elemento em um conjunto ordenado de dados: a1> a2>a3>....>an. Apresente um limite inferior (Ω) para este problema e exemplifique.

   Este problema consiste nas seguintes fases:
   a. Encontrar o local correto da inserção
      i. Busca sequencial →Melhor caso: primeiro elemento: $\Omega(1)$. Pior caso: no último elemento: $\Omega(n)$. Caso médio com equiprobabilidade: $\Omega(n)$.
   b. Inserção no local encontrado:
      i. Deslocar elementos maiores para o final do arranjo: Pior caso: inserir no início do arranjo: $\Omega(n)$. Melhor caso: no último elemento: $\Omega(1)$. Caso médio com equiprobabilidade: $\Omega(n)$.

   Das análises de a e b, temos que o problema é $\Omega(n) + \Omega(n) = \Omega(n)$.

4. Dizemos que um vetor P[1..m] ocorre em um vetor T[1..n] se P[1..m] = T[s+1..s+m] para algum s. O valor de um tal s é um deslocamento válido. Faça um programa para encontrar todos os deslocamentos válidos de um vetor e analise sua complexidade.

   Este problema consiste em varrer as posições i = 0,.., (n-m) de T. Ou seja, (n-m+1) iterações. Para cada i, varrer as próximas m posições j comparando se os elementos T[i] são iguais a P[j]. Temos então:

   $$f(n,m) = (n - m + 1)(m)$$

   Neste problema podemos ter duas situações: Uma quando P é muito pequeno ($m \cong 1$). Outra quando P se aproxima do tamanho de T ($m \cong n$). Analisemos estes dois casos:

   $$\begin{cases} f(n,1) = (n)1 \rightarrow O(n) \\ f(n,n) = (1)n \rightarrow O(n) \end{cases}$$

   Concluímos então que: $f(n) = O(n)$

5. Seja A={a(1) < .... < a(n)} uma lista de números reais. A proximidade entre a(i) e a(j) é definida como |a(i)-a(j)|. Fáca um programa que leia os inteiros j e k, encontre os k elementos de A mais próximos de A[j] em O(k).

```c
//Exercício 6 da lista de complexidade

void Exe6(float *V, int m, int j, int k)
{
  int pmenor, pdir, pesq;
  pesq = j - 1;
  pdir = j + 1;
  while (k>0)
  {
    if((pesq>=0)&(pdir<m))
    {
      if(fabs(V[pesq]-V[j]) < fabs(V[pdir]-V[j]))
      {
        pmenor = pesq;
        pesq--;
      }
      else
      {
        pmenor = pdir;
        pdir++;
      }
    }
    else if (pesq>=0) //Caso em que pdir chegou no final do vetor
    {
      pmenor = pesq;
      pesq--;
    }
    else if (pdir<m) //Caso em que pesq chegou no início do vetor
    {
      pmenor = pdir;
      pdir++;
    }
    else //Caso em que k>m. Não há k elementos a serem impressos
    {
      break;
    }
//Imprimir o k-ésimo valor mais próximo a V[j]:
    printf("%f",V[pmenor]);
    k--;
  }
  return;
}
```

6. Os subvetores de um vetor de valores V[1..m], são todos os vetores U_i[V[1], ..., V[i]], para todo $1 \leq i \leq m$. Faça um programa para calcular i tal que a soma dos elementos de U_i seja máxima. Dê a complexidade da solução encontrada.

Para cada $i = 1..n$ teremos que efetuar $(\sum_{j=1}^{i} V[i])$ e testar se é máximo. Como $\sum_{j=1}^{i} V[i]$ é O(n), pois i → n e teremos que fazer este somatório n vezes, $f(n) = O(n^2)$.

7. Seja A[1..n] um vetor com n números positivos e seja S[i]=A[1]+...+A[i]. Encontre o índice i que minimiza | S[i] – (A[i+1] + ... +A[n]) |. Dê a complexidade da solução encontrada.

Veja que testar uma possibilidade de S[i] consiste em fazer n somas (de 1 até i e depois de i+1 até n). Ou seja O(n).
Mas teremos que testar todas as possibilidades de S[i]: para cada $1 \leq i \leq n$ teremos que efetuar ($S[i] = \sum_{j=1}^{i} A[i]$) e testar se é menor que a soma dos elementos restantes do vetor ($\sum_{j=i+1}^{n} A[i]$). Ou seja, temos n testes: O(n).
Logo: $f(n) = O(n).O(n) = O(n^2)$