



## Lista de Exercícios sobre Função de Complexidade

- 1) Apresente a função de complexidade (no pior e melhor caso e no caso médio) para os programas abaixo, fazendo as considerações que considerar pertinente. Lembre-se que a função de complexidade quando não mencionado o caso refere-se ao pior caso.

a)

```
int Max(int A[n])
{
    int i, Temp;

    Temp = A[0];
    for (i = 1; i < n; i++)
        if (Temp < A[i])
            Temp = A[i];
    return Temp;
}
```

b)

```
void MaxMin1(int A[n], int* pMax, int* pMin)
{
    int i;

    *pMax = A[0];
    *pMin = A[0];
    for (i = 1; i < n; i++)
    {
        if (A[i] > *pMax) *pMax = A[i];
        if (A[i] < *pMin) *pMin = A[i];
    }
}
```

c)

```
void MaxMin2(int A[n], int* pMax, int* pMin)
{
    *pMax = A[0];
    *pMin = A[0];
    for (i = 1; i < n; i++)
    {
        if (A[i] > *pMax) *pMax = A[i];
        else if (A[i] < *pMin) *pMin = A[i];
    }
}
```



```
d)
void MaxMin3(Vetor A, int* pMax, int* pMin)
{
    int i, FimDoAnel;

    if ((n % 2) > 0)
    {
        A[n] = A[n - 1];
        FimDoAnel = n;
    }
    else FimDoAnel = n - 1;

    if (A[0] > A[1]) { *pMax = A[0]; *pMin = A[1]; }
    else             { *pMax = A[1]; *pMin = A[0]; }

    i = 3;
    while (i <= FimDoAnel)
    {
        if (A[i - 1] > A[i])
        {
            if (A[i - 1] > *pMax) *pMax = A[i - 1];
            if (A[i] < *pMin) *pMin = A[i];
        }
        else
        {
            if (A[i - 1] < *pMin) *pMin = A[i - 1];
            if (A[i] > *pMax) *pMax = A[i];
        }
        i += 2;
    }
}
```



```
e)
void bubblesort( int A[n], int n)
{
    int i, j;
    int aux;

    for( j = 0; j < n; j++ )
    {
        for( i = 0; i < n - 1; i++ )
        {
            if( A[i] > A[i+1] )
            {
                aux      = A[i];
                A[i]     = A[i+1];
                A[i+1]   = aux;
            }
        }
    }
}
```

```
f)
void bubblesort2( int A[n], int n)
{
    int i, j, troca;
    int aux;

    do
    {
        troca = 0;
        for ( i = 0 ; i <= n-1 ; i++ )
        {
            if ( A[i] > A[i+1] )
            {
                aux      = A[i];
                A[i]     = A[i+1];
                A[i+1]   = aux;
                troca = 1;
            }
        }
    } while (troca);
}
```



```
g)
void selectsort ( int A[n] , int n)
{
    int i,j,min;
    int aux;

    for ( i = 0 ; i < n - 1 ; i++ )
    {
        min = i;
        for ( j = i + 1 ; j < n ; j++ )
            if ( A[j] < A[min] )
                min = j;

        aux = A[min];
        A[min] = A[i];
        A[i] = aux;
    }
}
```

```
h)
void insertsort(int A[n], int n )
{
    int j;
    for (int i = 1; i < n; i++)
    {
        aux = A[i];
        j = i - 1;

        while ( ( j >= 0 ) && ( aux < v[j] ) )
        {
            v[j + 1] = v[j];
            j--;
        }
        v[j + 1] = aux;
    }
}
```



```
i)
void FazAlgo( int n )
{
    int i,j,k;

    x = 0;
    for( i = 1 ; i <= n - 1 ; i++ )
    {
        for( j = i + 1 ; j <= n ; j++ )
        {
            for( k = 1 ; k <= j ; k++ )
                x = x + 1;
        }
    }
}

j)
void FazAlgo2( int n)
{
    int i,j,k,x;
    x = 0;
    for( i = 1 ; i <= n ; i ++ )
        for( j = i + 1 ; j <= n - 1 ; j++ )
            for( k = 1 ; k <= j ; k++ )
                x = x + 1;
}
```



- 2) Considere o algoritmo abaixo. O que ele faz? Qual é a função de complexidade do número de comparações no melhor caso e no pior caso? Que configuração do vetor de entrada **A** leva a essas duas situações? **Explique** como você chegou a esses resultados.

```
Type Vetor = array [0..MaxTam] of integer;
...
procedure FazAlgo(var A: Vetor; n: integer);
var i, j, x: integer;
begin
  for i := 2 to n do begin
    x := A[i];
    j := i - 1;
    A[0] := x; { sentinela para terminar o loop qdo j = 0
  }
  while x < A[j] do begin
    A[j + 1] := A[j];
    j := j - 1;
  end;
  A[j + 1] := x;
end;
end;
```