NMEC: 93097 **NOME:** JOÃO PEDRO COSTA GAMEIRO

AULA 3 – ANÁLISE DA COMPLEXIDADE DE ALGORITMOS

1 - Seja uma dada sequência (*array*) de n elementos inteiros. Pretende-se determinar quantos elementos da sequência são diferentes do seu elemento anterior. Ou seja:

$$\text{array [i]} \neq \text{array [i–1]}, \text{ para i} > 0$$

- Implemente uma **função eficiente** e **eficaz** que determine quantos elementos (resultado da função) de uma sequência com n elementos (sendo n > 1) respeitam esta propriedade.

  Depois de validar o algoritmo apresente-o no verso da folha.

- Determine experimentalmente a **ordem de complexidade do número de comparações** efetuadas pelo algoritmo e envolvendo elementos da sequência. Considere as seguintes 10 sequências de 10 elementos inteiros, todas diferentes, e que cobrem as distintas situações possíveis de execução do algoritmo. Determine, para cada uma delas, o número de elementos que obedecem à condição e o número de comparações efetuadas.

Sequência	Resultado	N.º de operações
{3, 3, 3, 3, 3, 3, 3, 3, 3, 3}	0	9
{4, 3, 3, 3, 3, 3, 3, 3, 3, 3}	1	9
{4, 5, 3, 3, 3, 3, 3, 3, 3, 3}	2	9
{4, 5, 1, 3, 3, 3, 3, 3, 3, 3}	3	9
{4, 5, 1, 2, 3, 3, 3, 3, 3, 3}	4	9
{4, 5, 1, 2, 6, 3, 3, 3, 3, 3}	5	9
{4, 5, 1, 2, 6, 8, 3, 3, 3, 3}	6	9
{4, 5, 1, 2, 6, 8, 7, 3, 3, 3}	7	9
{4, 5, 1, 2, 6, 8, 7, 9, 3, 3}	8	9
{4, 5, 1, 2, 6, 8, 7, 9, 3, 0}	9	9

Depois da execução do algoritmo responda às seguintes questões:

- Em termos do número de comparações efetuadas, podemos distinguir alguma variação na execução do algoritmo? Ou seja, existe a situação de melhor caso e de pior caso, ou estamos perante um algoritmo com caso sistemático?

  O algoritmo para cada valor de *n* compara todos os elementos do array com o seu anterior, ou seja, faz sempre *n-1* comparações. Logo podemos afirmar que não existe a situação de melhor nem de pior caso e que estamos na presença de um algoritmo com caso sistemático.

  Podemos comprovar esta situação através dos resultados obtidos anteriormente aonde é possível observar que apesar de ocorrer uma variação no número de elementos que são diferentes do seu anterior, o número de comparações em cada caso se mantém sempre igual (9 neste caso).

- Qual é a ordem de complexidade do algoritmo?

  A ordem de complexidade do algoritmo é linear, $O(n)$.

- Determine formalmente a ordem de complexidade do algoritmo. Tenha em atenção que deve obter uma expressão matemática exata e simplificada. **Faça a análise no verso da folha.**

- Calcule o valor da expressão para N = 10 e compare-o com os resultados obtidos experimentalmente.

APRESENTAÇÃO DO ALGORITMO

```
int compareElements (int array[], int n)
{
    int i, result=0;
    for (i=1; i<n; i++) {
        if(array[i]!=array[i-1]) {
            result++;
        }
    }
    return result;
}
```

ANÁLISE FORMAL DO ALGORITMO

E(N) =

$$\sum_{i=1}^{n-1} 1 = n - 1 \cong n$$

Logo concluímos que a complexidade do algoritmo é linear, $O(n)$.

Para n=10 elementos obtemos o seguinte número de comparações:

$$E(10) = \sum_{i=1}^{10-1} 1 = 10 - 1 = 9$$

Podemos verificar que nos resultados obtidos anteriormente, o número de comparações obtido foi sempre 9 para um total de 10 elementos.

É importante referir que como estamos na presença de um algoritmo com caso sistemático podemos afirmar que:

$$B_c(n) = W_c(n) = A_c(n) = n - 1$$

2 - Seja uma dada sequência (*array*) de n elementos inteiros e não ordenada. Pretende-se determinar qual é o primeiro elemento da sequência que tem mais elementos menores do que ele atrás de si, e indicar a posição (índice do *array*) onde esse elemento se encontra.

Por exemplo, na sequência { 1, 9, 2, 8, 3, 4, 5, 3, 7, 2 } o elemento 7, que está na posição de índice 8 da sequência, é maior do que 6 elementos seus predecessores. Na sequência { 1, 7, 4, 6, 5, 2, 3, 2, 1, 0 } o elemento 6, que está na posição de índice 3 da sequência, é maior do que 2 elementos seus predecessores. Mas, na sequência { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 } nenhum elemento é maior do que qualquer um dos seus predecessores, pelo que deve ser devolvido -1 como resultado.

- Implemente uma **função eficiente** e **eficaz** que determine o índice do primeiro elemento (resultado da função) de uma sequência com n elementos (sendo n > 1) que tem o maior número de predecessores menores do que ele.

  Depois de validar o algoritmo apresente-o no verso da folha.

- Determine experimentalmente a **ordem de complexidade do número de comparações** efetuadas envolvendo elementos da sequência. Considere as sequências anteriormente indicadas de 10 elementos inteiros e outras sequências diferentes à sua escolha. Determine, para cada uma delas, o índice do elemento procurado e o número de comparações efetuadas.

  As comparações têm ordem de complexidade quadrática, $O(n^2)$.

  { 1, 9, 2, 8, 3, 4, 5, 3, 7, 2 } – Índice devolvido: 8, Nº de Comparações: 54;
  { 1, 7, 4, 6, 5, 2, 3, 2, 1, 0 } – Índice devolvido: 3, Nº de Comparações: 54;
  { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 } – Índice devolvido:-1, Nº de Comparações: 54;

Depois da execução do algoritmo responda às seguintes questões:

- Em termos do número de comparações efetuadas, podemos distinguir alguma variação na execução do algoritmo? Ou seja, existe a situação de melhor caso e de pior caso, ou estamos perante um algoritmo com caso sistemático?

  Estamos perante um algoritmo com caso sistemático, pois qualquer que seja o índice devolvido (quer seja a primeira ou última posição ou não encontrado), o número de comparações vai ser sempre o mesmo para um certo *n*.

  Esta situação pode ser observada pelos valores obtidos experimentalmente (para os três diferentes valores de índice devolvido, o número de comparações obtido foi de 54).

- Qual é a ordem de complexidade do algoritmo?

  O algoritmo tem uma ordem de complexidade quadrática, $O(n^2)$.

- Determine formalmente a ordem de complexidade do algoritmo. Tenha em atenção que deve obter uma expressão matemática exata e simplificada. **Faça a análise no verso da folha.**

- Calcule o valor da expressão para N = 10 e compare-o com os resultados obtidos experimentalmente.

APRESENTAÇÃO DO ALGORITMO

```
int getIndex (int array[], int n) {
    int i, j, maxi=-1, cntmax=0, cnt=0;
    for(i=1;i<n;i++){
        for(j=0;j<i;j++){
            if(array[j]<array[i])
                cnt++;
        }
        if(cnt>cntmax){
            cntmax = cnt;
            maxi = i;
        }
        cnt=0;
    }
    return maxi;
}
```

ANÁLISE FORMAL DO ALGORITMO

E(N) =

$$\left(\sum_{i=1}^{n-1}\left(\sum_{j=0}^{i}1\right)+1\right)=\sum_{i=1}^{n-1}i+1=\frac{2+n}{2}\times(n-1)=\frac{n^2+n-2}{2}\cong n^2$$

Logo concluímos que a complexidade do algoritmo é quadrática $O(n^2)$.

Para n = 10 elementos obtemos o seguinte número de comparações:

$$E(10)=\left(\sum_{i=1}^{10-1}\left(\sum_{j=0}^{i}1\right)+1\right)=\sum_{i=1}^{9}i+1=\frac{2+10}{2}\times 9=6\times 9=54$$

Este resultado também pode ser observado na segunda fórmula obtida e nos resultados experimentais:

$$E(10)=\frac{10^2+10-2}{2}=\frac{110-2}{2}=\frac{108}{2}=54$$

Como este algoritmo também é um algoritmo com caso sistemático podemos afirmar que:

$$B_c(n)=W_c(n)=A_c(n)=\frac{n^2+n-2}{2}$$