NMEC: 92996 **NOME:** Joni Luis Costa

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:

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

- 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.

3	3	3	3	3	3	3	3	3	3	Resultado 0	Nº de operações
4	3	3	3	3	3	3	3	3	3	Resultado 1	Nº de operações
4	5	3	3	3	3	3	3	3	3	Resultado 2	Nº de operações
4	5	1	3	3	3	3	3	3	3	Resultado 3	Nº de operações
4	5	1	2	3	3	3	3	3	3	Resultado 4	Nº de operações
4	5	1	2	6	3	3	3	3	3	Resultado 5	Nº de operações
4	5	1	2	6	8	3	3	3	3	Resultado 6	Nº de operações
4	5	1	2	6	8	7	3	3	3	Resultado 7	Nº de operações
4	5	1	2	6	8	7	9	3	3	Resultado 8	Nº de operações
4	5	1	2	6	8	7	9	3	0	Resultado 9	Nº de operações

Todos
n - 1

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? Caso Sistemático

- Qual é a ordem de complexidade do algoritmo?

  $O(n)$

- 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.

  No verso da folha

APRESENTAÇÃO DO ALGORITMO

```c
#include <stdio.h>

int check(int prev, int actual){
    if(prev == actual){
        return 0;
    }
    return 1;
}

int main()
{
    int list[] = {0,1,2,3,4,5,6,7,8,9};
    int size = 10;
    int i;
    int resp = 0;
    int counter = 0;
    for(i = 1; i < size;i++){
        resp += check(list[i-1],list[i]);
        counter++;
    }
    printf("N exe: %d\n",counter);
    printf("Val %d\n",resp);
    return 0;
}
```

ANÁLISE FORMAL DO ALGORITMO

E(N) =

$$\sum_{i=1}^{n-1} (1) = n-1$$

Para $n = 10$, As comparações são $n-1$, logo

$10-1 = 9$

Igual aos resultados Experimentais

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.

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? Melhor Caso: O último número do índice é maior que todos os outros, n-1 Comparações

  Pior Caso: Todos os numeros iguais ou por ordem descrescente
- Qual é a ordem de complexidade do algoritmo?

  $O\left(\frac{n}{2}(n-1)\right)$
- 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.

  No Verso da folha

APRESENTAÇÃO DO ALGORITMO

```c
#include <stdio.h>
int main()
{
    int list[] = {3, 4, 1, 5, 2, 1 , 3 , 4 , 9 , 10};
    int size = 10;
    int i;
    int j;
    int resp = 0;
    int counter = 0;
    int counterComp = 0;
    int max = 0;
    int index = -1;
    for (i = size-1; i >= 0;i--)
    {
        if(i <= max){
            break;
        }
        for (j = 0; j < i; j++)
        {
            counterComp++;
            if (list[j] < list[i])
            {
                counter++;
            }
        }
        if (counter > max)
        {
            max = counter;
            index = i;
        }
        counter = 0;
    }
    printf("Indice: %d\n", index);
    printf("N de menores %d\n", max);
    printf("Comp %d\n",counterComp);
    return 0;
}
```

ANÁLISE FORMAL DO ALGORITMO

Pior Caso:

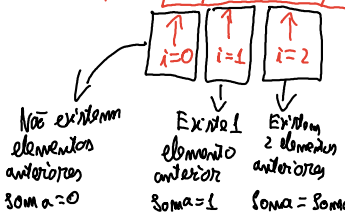
$$E(N) = \sum_{i=0}^{n-1} (i) = \frac{n}{2}(n-1)$$

-> Último Índice do array

-> Primeiro Índice do array

-> Para cada valor de I a ser iterado somamos o nº de elementos do array anterior a ele:

Exemplo:  Array de 5 elementos



(...) Neste caso o Pior vai ser 10 comparações

Melhor Caso:

Como neste algoritmo o for inicial vai iterar do fim para o início do array, o melhor caso ocorre quando todos os números anteriores ao último número do array são menores que este, sendo assim o índice com o maior número de casos possíveis.

$$\sum_{i=1}^{n-1} (1) = n-1 \text{ comparações}$$

-> Resolvendo do somatório, ao substituir o n, temos o número de comparações máximo.

Para n = 10, experimentalmente o pior caso é 45, igual a $\frac{10}{2}(10-1) = 45$