|  |  |  |  |
| --- | --- | --- | --- |
| **NMec:** |  | **Nome:** |  |

**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] ≠ array [i–1], 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?

R: Não conseguimos distinguir qualquer variação na execução do algoritmo, logo estamos perante um caso sistemático.

* Qual é a ordem de complexidade do algoritmo?

R: Ordem de complexidade linear.

* 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.

R: Quando temos n=10 vai ser:

10-1 = 9, ou seja, o número de comparações para n=10 é 9.

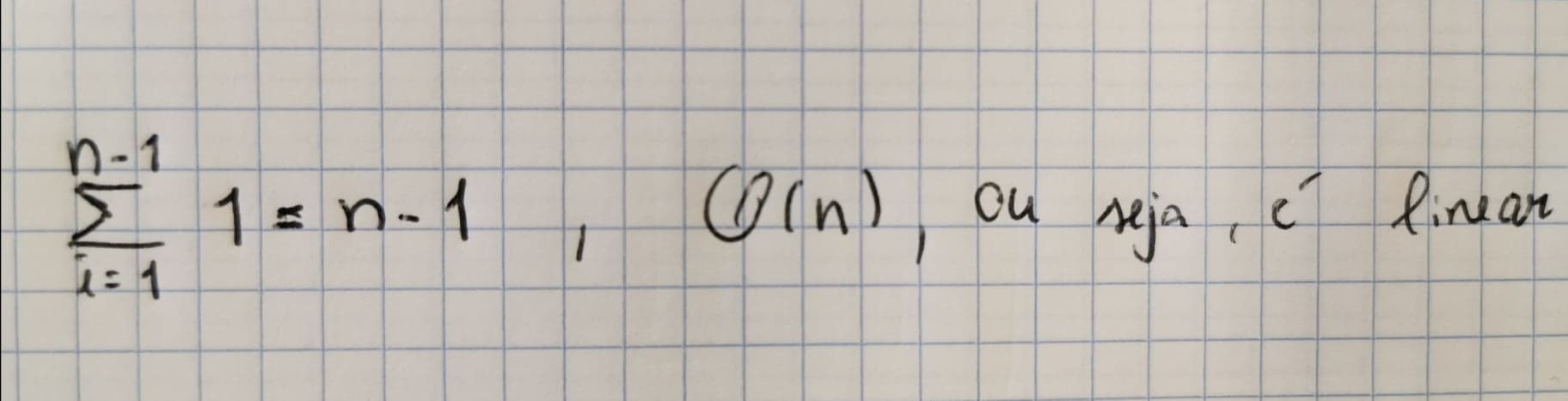
Se compararmos com os resultados experimentais podemos ver que em ambos o número de comparações é 9.

**Apresentação do Algoritmo**

|  |
| --- |
| int funct(int n){    assert (n>1);  int count=0;  int comp=0;  int arr[n];  for(int i=0; i<n; i++){  scanf(“%d”, &arr[i]);  }  for(int i=1; i<n; i++){  if(arr[i]!=arr[i-1]){  count++;  }  comp++;  }    printf(“Comparações : %d\n”, comp);  return count;  } |

**Análise Formal do Algoritmo**

**E(n) =**

****

**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?

R: Não conseguimos distinguir qualquer variação na execução do algoritmo, logo estamos perante um caso sistemático.

* Qual é a ordem de complexidade do algoritmo?

R: Ordem de complexidade quadrática.

* 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.

R: Quando temos n=10 vai ser:

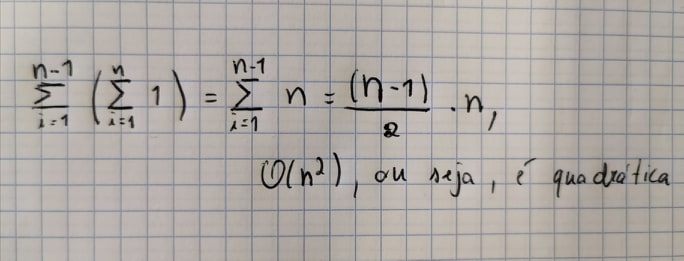
10\*((10-1)/2) = 45, ou seja, o número de comparações para n=10 é 45.

Se compararmos com os resultados experimentais podemos ver que em ambos o número de comparações é 45.

**Apresentação do Algoritmo**

|  |
| --- |
| int funct(int n){    assert (n>1);  int count=0;  int comp=0;  int max=0;  int index=-1;  int arr[n];  for(int i=0; i<n; i++){  scanf(“%d”, &arr[i]);  }  for(int i=1; i<n; i++){  count=0;  for(int j=0; j<i; j++){  if(arr[i]>arr[j]){  count++;  }  comp++;  }  if(max<count){  index=i;  max=count;  }  }    printf(“Comparações : %d\n”, comp);  return index;  } |

**Análise Formal do Algoritmo**

**E(n) =**