**Nome: Afonso Pinheiro Teixeira**

**Nº mec:93170**

Aula 4 - Análise da Complexidade de Algoritmos

**1 –** Considere uma sequência (*array*) de n elementos inteiros, ordenada por **ordem não decrescente**. Pretende-se determinar se a sequência é uma **progressão aritmética de razão 1**, i.e., a[i+1] – a[i] = 1.

* Implemente uma função **eficiente** (utilize um algoritmo em lógica negativa) e **eficaz** que verifique se uma sequência com n elementos (n > 1) define uma sequência contínua de números. A função deverá devolver 1 ou 0, consoante a sequência verificar ou não essa propriedade.

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

* Determine experimentalmente a **ordem de** **complexidade do número de adições/subtrações** 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, se satisfaz a propriedade e qual o número de operações de adição/subtração efetuadas pelo algoritmo.

|  |  |  |
| --- | --- | --- |
| **Sequência** | **Resultado** | **N.º de operações** |
| {1, 3, 4, 5, 5, 6, 7, 7, 8, 9} | 0 | 1 |
| {1, 2, 4, 5, 5, 6, 7, 8, 8, 9} | 0 | 2 |
| {1, 2, 3, 6, 8, 8, 8, 9, 9, 9} | 0 | 3 |
| {1, 2, 3, 4, 6, 7, 7, 8, 8, 9} | 0 | 4 |
| {1, 2, 3, 4, 5, 7, 7, 8, 8, 9} | 0 | 5 |
| {1, 2, 3, 4, 5, 6, 8, 8, 9, 9} | 0 | 6 |
| {1, 2, 3, 4, 5, 6, 7, 9, 9, 9} | 0 | 7 |
| {1, 2, 3, 4, 5, 6, 7, 8, 8, 9} | 0 | 8 |
| {1, 2, 3, 4, 5, 6, 7, 8, 9, 9} | 0 | 9 |
| {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} | 1 | 9 |

**Depois da execução do algoritmo responda às seguintes questões:**

* Qual é a sequência (ou as sequências) que corresponde(m) ao melhor caso do algoritmo?

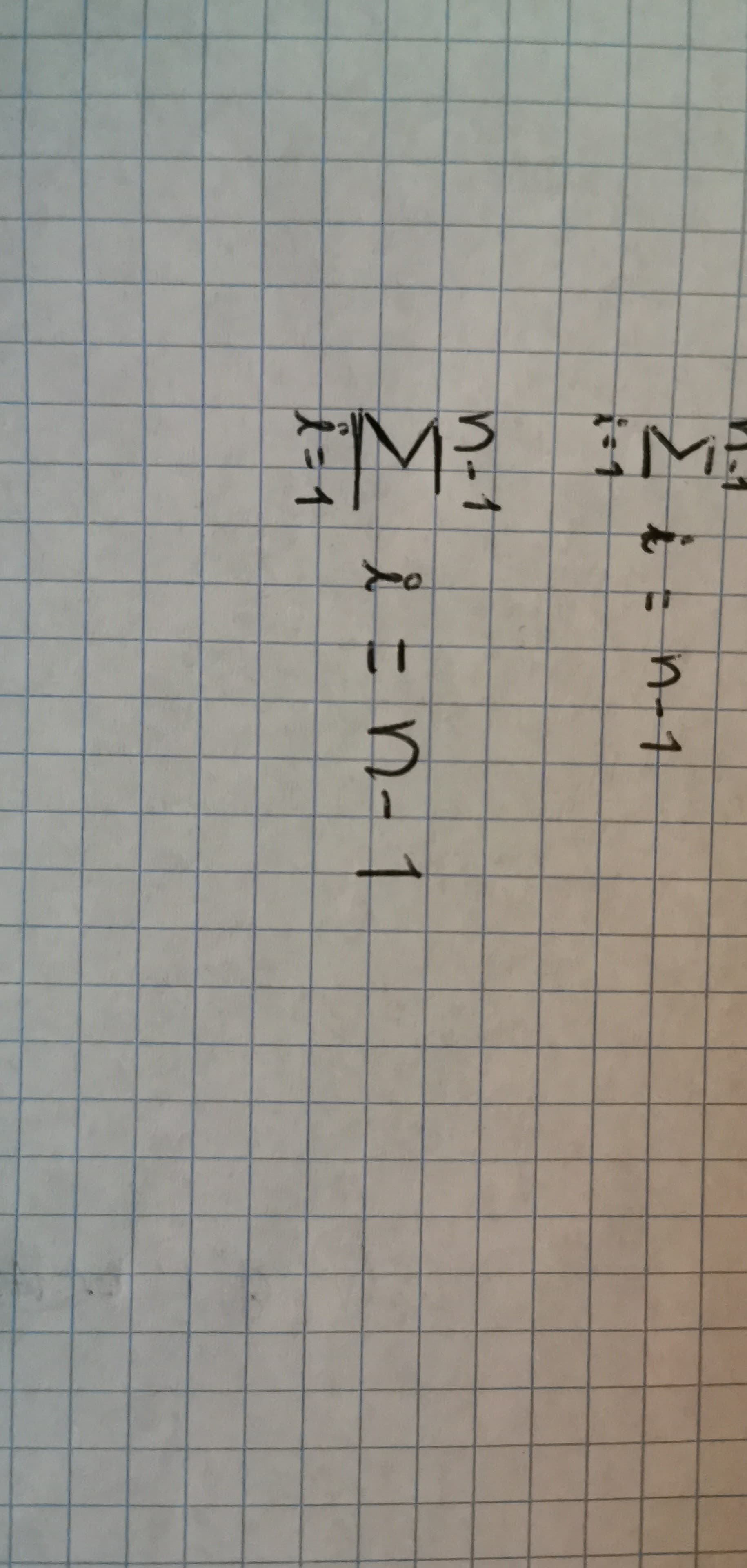
R: O melhor caso acontece na primeira sequência, pois entre o primeiro valor e o segundo da sequência a diferença é diferente de 1, logo só se vai realizar uma comparação.

* Qual é a sequência (ou as sequências) que corresponde(m) ao pior caso do algoritmo?

R: O pior caso acontece nas duas últimas sequências, pois é onde se realiza o maior número de comparações.

* Determine o número de adições efetuadas no caso médio do algoritmo (**para n = 10**).

R: A(10) = 5,4.



* Qual é a ordem de complexidade do algoritmo?

R: Ordem linear.­

* Determine formalmente a ordem de complexidade do algoritmo nas situações do melhor caso, do pior caso e do caso médio, considerando uma sequência de tamanho n. Tenha em atenção que deve obter expressões matemáticas exatas e simplificadas. **Faça as análises no verso da folha.**
* Calcule o valor das expressões para n = 10 e compare-os com os resultados obtidos experimentalmente.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Apresentação do Algoritmo**

int globalCount=0; // contador global de comparações

int funct(int\* arr, int n){

assert (n>1);

globalCount=0;

int res=1;

for(int i=0;i<n-1;i++){

globalCount++;

if(arr[i+1]-arr[i]!=1){

res=0;

break;

}

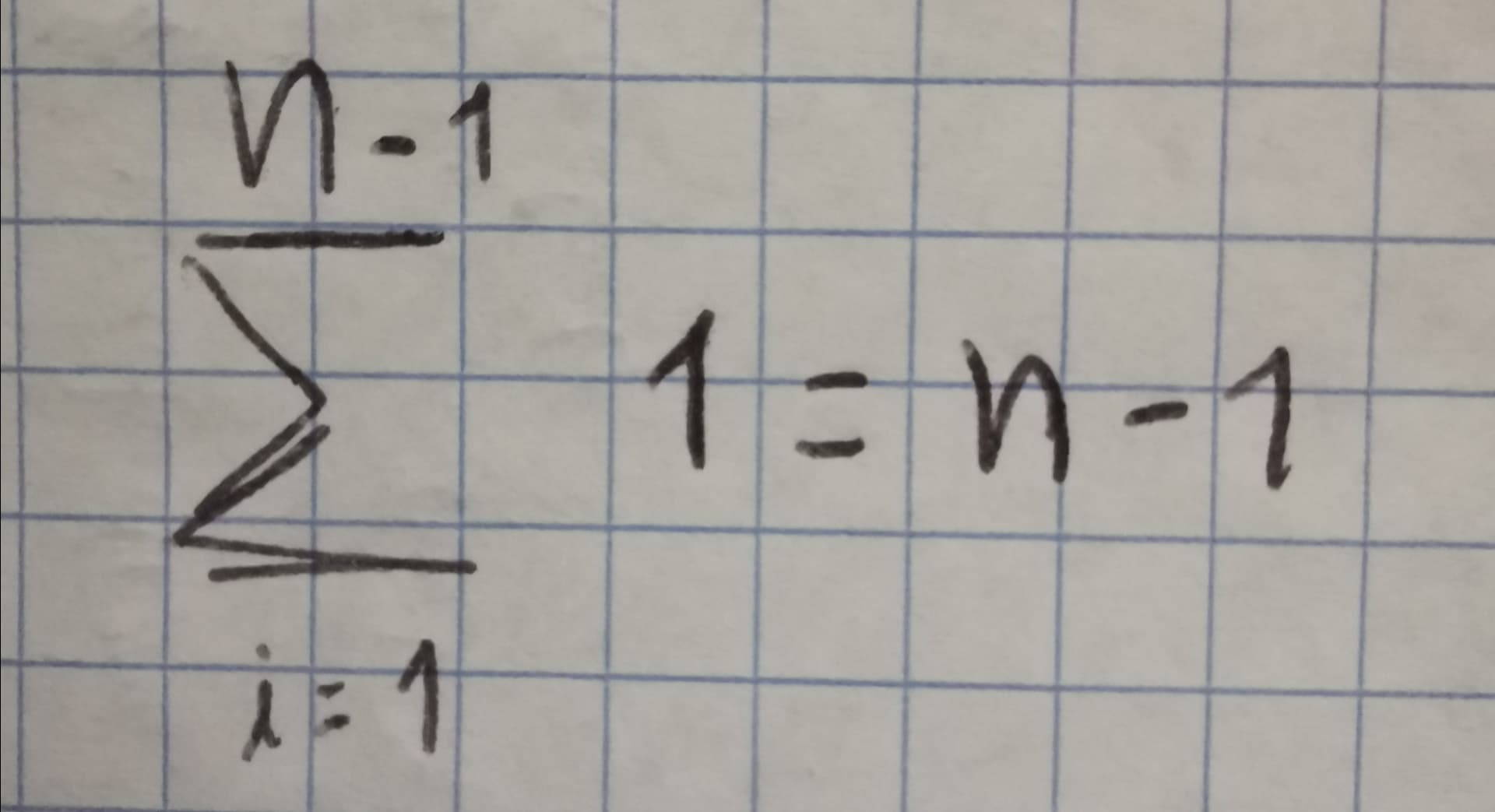
}

return res;

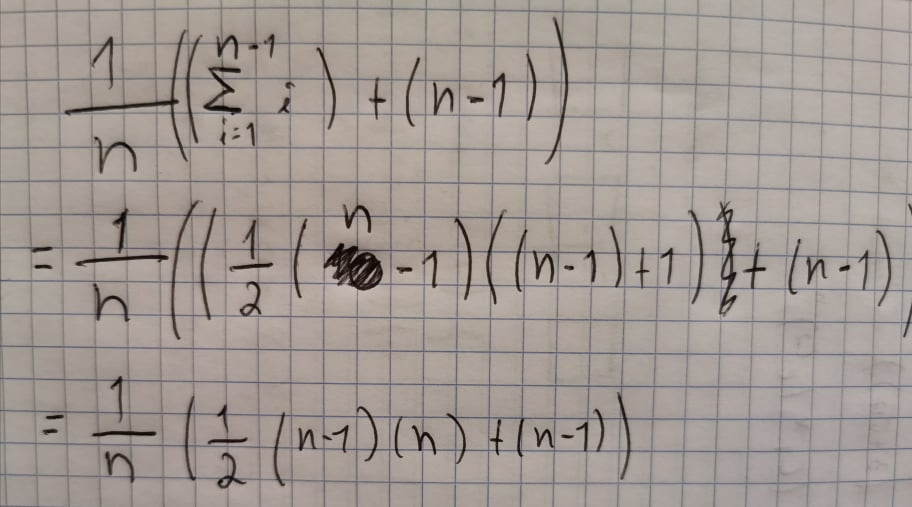
}

Análise Formal do Algoritmo

Melhor Caso - B(n) = 1



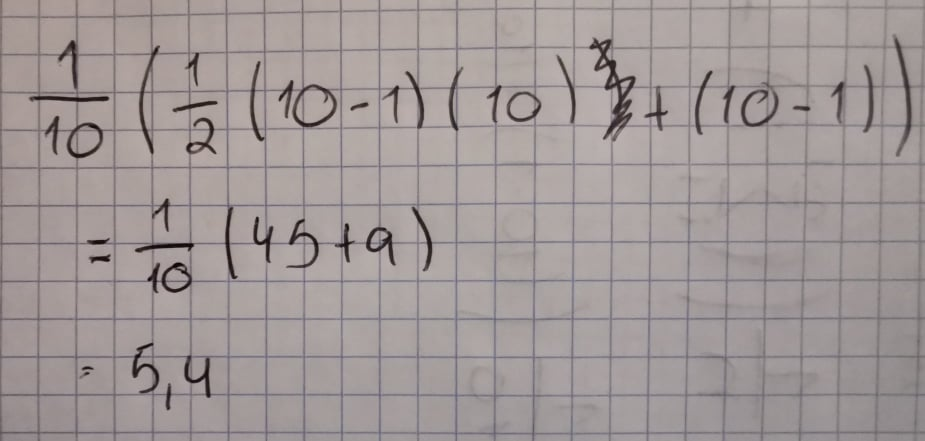
Pior Caso - W(n) =



Caso Médio - A(n) =

Melhor caso – b(10) = 1, igual ao obtido experimentalmente

Pior Caso – W(10) = 10 -1 = 9, igual ao obtido experimentalmente



Caso médio -A(10) =

**2 –** Considere uma sequência (array) não ordenada de n elementos inteiros. Pretende-se eliminar os elementos repetidos existentes na sequência, sem fazer uma pré-ordenação e sem alterar a posição relativa dos elementos. Por exemplo, a sequência { 1, 2, 2, 2, 3, 3, 4, 5, 8, 8 } com 10 elementos será transformada na sequência { 1, 2, 3, 4, 5, 8 } com apenas 6 elementos. Por exemplo, a sequência { 1, 2, 2, 2, 3, 3, 3, 3, 8, 8 } com 10 elementos será transformada na sequência { 1, 2, 3, 8 } com apenas 4 elementos. Por exemplo, a sequência { 1, 2, 3, 2, 1, 3, 4 } com 7 elementos será transformada na sequência { 1, 2, 3, 4 } com apenas 4 elementos. Mas, a sequência { 1, 2, 5, 4, 7, 0, 3, 9, 6, 8 } permanece inalterada.

* Implemente uma função **eficiente** e **eficaz** que elimina os elementos repetidos numa sequência com n elementos (n > 1). A função deverá ser *void* e alterar o valor do parâmetro indicador do número de elementos efetivamente armazenados na sequência (que deve ser passado por referência).

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

* Determine experimentalmente a **ordem de** **complexidade do número de comparações** e **do número de deslocamentos** envolvendo elementos da sequência. Considere as sequências anteriormente indicadas de 10 elementos e outras à sua escolha. Determine, para cada uma delas, a sua configuração final, bem como o número de comparações e de deslocamentos efetuados.

**Depois da execução do algoritmo responda às seguintes questões:**

* Indique uma sequência inicial com 10 elementos que conduza ao **melhor caso do número de comparações** efetuadas. Qual é a sequência final obtida? Qual é o número de comparações efetuadas? Qual é o número de deslocamentos (i.e., cópias) de elementos efetuados?

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inicial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  | Nº Comparações: 9 |
| Final | 1 |  |  |  |  |  |  |  |  |  |  | Nº Cópias: 36 |

Justifique a sua resposta: Pois neste caso como todos os números da sequência são iguais logo só será necessário comparar uma vez.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Indique uma sequência inicial com 10 elementos que conduza ao **pior caso do número de comparações** efetuadas. Qual é a sequência final obtida? Qual é o número de comparações efetuadas? Qual é o número de deslocamentos (i.e., cópias) de elementos efetuados?

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inicial | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Nº Comparações: 45 |
| Final | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Nº Cópias: 0 |

Justifique a sua resposta: Como todos os números são diferentes o algoritmo tem de comparar todos os números para ver se há números repetidos.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Determine formalmente a ordem de complexidade do algoritmo nas situações do **melhor caso** e do **pior caso**, considerando uma sequência de tamanho n. Tenha em atenção que deve obter expressões matemáticas exatas e simplificadas. **Faça as análises no verso da folha.**

Apresentação do Algoritmo

int copias=0; // contador de shifts

int comp=0; // contador de comparações

int size=10;

void funct(int\* arr, int\* size) {

assert (size>0);

for(int i=0;i<\*size;i++){

for(int j = i+1; j<\*size; j++){

comp++;

if(arr[i] == arr[j] ){

(\*size)--;

for(int k=j; k<\*size; k++){

arr[k] = arr[k+1];

copias++; }

j--;

}

}

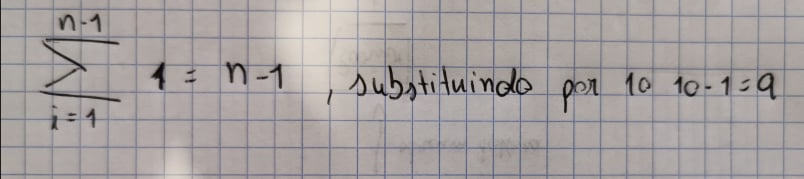
}

}

Análise Formal do Algoritmo

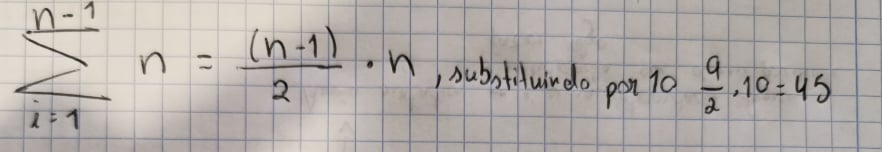
Nº de Comparações

Melhor Caso - B(n) =



Ordem linear

Pior Caso - W(n) =

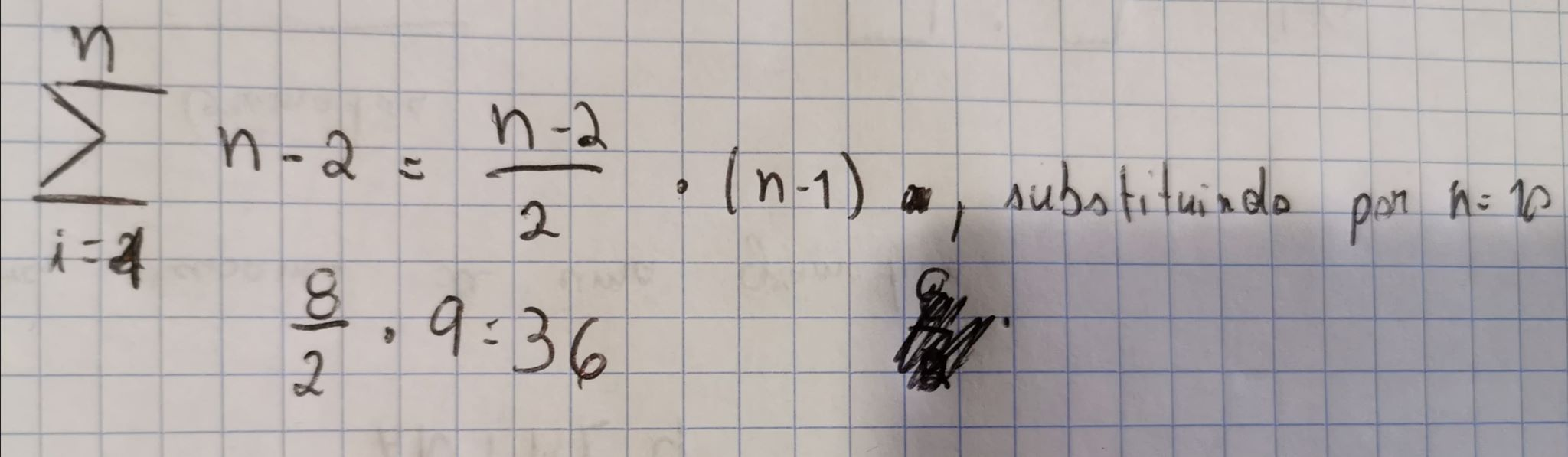


Ordem quadrática

Em ambos os casos a análise formal e a análise experimental deram resultados iguais.

Nº de Deslocamentos de Elementos

Melhor Caso - B(n) = 0, para N=10, o resultado será 0. Que coincide com o obtido experimentalmente.

Pior Caso - W(n) =

Ordem Quadrática