NOME:	**JOÃO PEDRO COSTA GAMEIRO**	**N.º MEC:**	**93097**

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}`	0	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?

A sequência {1, 3, 4, 5, 5, 6, 7, 7, 8, 9}.

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

As sequências {1, 2, 3, 4, 5, 6, 7, 8, 9, 9} e {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

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

O número de adições no caso médio depende de uma probabilidade p, que vamos considerar ½. Assim sendo obtemos que o número de adições efetuadas no caso médio para n=10 é 7
(ver análise formal do caso médio).

- Qual é a ordem de complexidade do algoritmo?

O algoritmo tem complexidade linear, *O(n)*.

- 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.

ANÁLISE FORMAL DO ALGORITMO

MELHOR CASO - B(N) = 1 $, \in O(1)$
Quando a primeira subtração não verifica a condição, não voltam a ser efetuadas novas subtrações e o algoritmo devolve 0.

PIOR CASO - W(N) = N-1 $, \in O(n)$
Ocorre quando a última subtração (a[n-1] - a[n-2] ≠ 1) não verifica a condição (tendo em conta que todas as anteriores verificaram) ou quando todas as subtrações efetuadas verificam a condição (a[i+1] - a[i] = 1).

CASO MÉDIO - A(N) =
Para calcular o caso médio temos de analisar todos os casos possíveis e as suas respetivas probabilidades.

Casos Possíveis	Nº Subtrações	Nº Casos	Probabilidades
a[1] – a[0] ≠ 1	1	Casos Insucesso - n-1 casos	Insucesso (p) (p / n-1) – por cada caso
a[2] – a[1] ≠ 1	2		
a[i+1] – a[i] ≠ 1	i+1		
a[n+1] – a[n-2] ≠ 1	n-1		
a[n+1] – a[n-2] = 1	n-1	Caso Sucesso - 1 caso	Sucesso (1-p)

$$A_c(n) = \left(\sum_{i=0}^{n-2}\left(\frac{p}{n-1}\right)\cdot(i+1)\right) + (1-p)\cdot(n-1) =$$
$$= \left(\frac{p}{n-1}\right)\cdot\left(\frac{1+n-1}{2}\cdot(n-1)\right) + (1-p)\cdot(n-1) =$$
$$= p\cdot\frac{n}{2} + (1-p)(n-1) \qquad , \in O(n)$$

E obtemos uma expressão para o caso médio em função de uma probabilidade p, na qual 1-p representa a probabilidade de sucesso e p de insucesso.

*__NOTA:__ Na contabilização dos casos possíveis presume-se que se o algoritmo chegou à subtração referida é porque todas as outras cumpriam a condição a[i+1]-a[i] = 1;

- Calcule o valor das expressões para n = 10 e compare-os com os resultados obtidos experimentalmente.

MELHOR CASO - B(10) = 1, tal como o valor obtido para a primeira sequência

PIOR CASO - W(10) = 10-1 = 9 tal como o valor obtido para as duas últimas sequências em que têm de ser efetuadas todas as Subtrações

CASO MÉDIO - A(10) – Considerando as probabilidades de sucesso (1-p) e insucesso (p) iguais (1/2), podemos obter A(n) = $\frac{1}{2}\cdot\frac{n}{2}+\frac{1}{2}\cdot(n-1) = \frac{n}{4}+\frac{n-1}{2}$ e concluímos assim que:

A(10) = $\frac{10}{4}+\left(\frac{10-1}{2}\right) = \frac{10}{4}-\frac{9}{2} = 7$

APRESENTAÇÃO DO ALGORITMO

```
int isArithmeticProgression1(int array[], int n)
{
    assert(n>1);
    int i=0;
    numSubs = 0;
    for(i=0;i<n-1;i++){
        numSubs++;            //contagem do número de subtrações
        if((array[i+1]-array[i])!=1)
            return 0;
    }
    return 1;
}
```

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	2
Final:	1	2								

N.º de comparações: 9

N.º de Cópias: 36

Justifique a sua resposta:

Este representa o melhor caso do número de comparações, pois só ocorre uma iteração, em que se compara o primeiro elemento com todos os outros. Como temos um ciclo while com a condição "while(array[i]==array[j])", e o último elemento é diferente do primeiro, não é efetuada nenhuma comparação adicional. Caso isso não acontecesse (ou seja, sequência com os números todos iguais), no final, após as 9 comparações era realizada ainda mais uma para a saída do ciclo while (em que array[i]≠array[j]).

- 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:	5	7	9	1	2	4	6	0	3	3
Final:	5	7	9	1	2	4	6	0	3	

N.º de comparações: 46

N.º de Cópias: 0

Justifique a sua resposta:

Este representa o pior caso do número de comparações, pois são efetuadas todas as comparações, ou seja, cada elemento é comparado com os seus precedentes. Nesta situação, para além de todas as comparações anteriormente referidas é ainda efetuada um adicional pelo facto de os dois últimos elementos serem iguais. A comparação adicional vai indicar a saída do ciclo while (array[n-2] ≠array[n-1]).

- 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.

ANÁLISE FORMAL DO ALGORITMO - NÚMERO DE COMPARAÇÕES
MELHOR CASO - B(N) = Para o melhor caso como já vimos anteriormente, temos de considerar que todos os elementos são iguais exceto o último para que não seja realizada nenhuma comparação adicional para a saída do ciclo while. Logo nesse caso vamos obter que $C_i = n - 1$ e assim ocorre apenas uma iteração de cada ciclo logo: $$B(N) = \sum_{i=0}^{0}\left(\sum_{j=1}^{1} n-1\right) = n-1 \quad ,\in O(n)$$
PIOR CASO - W(N) = Para o pior caso têm de ser efetuadas todas as comparações de cada elemento com os seus precedentes, mais a comparação adicional de saída do ciclo while. Assim sendo obtemos que $C_i = 1$ e com este valor obtemos todas as comparações dos elementos com os seus precedentes. No final adicionamos mais 1 que representa a comparação de saída do ciclo while (os dois últimos elementos são iguais). $$W(N) = \sum_{i=0}^{n-2}\left(\sum_{j=i+1}^{n-1} 1\right) + 1 = \sum_{i=0}^{n-2}\big(n-(i+1)\big) + 1 = \frac{n-1+\big(n-(n-1)\big)}{2}\cdot(n-1) + 1 = \frac{n^2-n}{2} + 1$$ $$,\in O(n^2)$$
NOTA: C_i representa as comparações do ciclo while que podem variar entre 1 (não se chega a entrar no ciclo) ou N.

ANÁLISE FORMAL DO ALGORITMO - NÚMERO DE DESLOCAMENTOS DE ELEMENTOS
MELHOR CASO - B(N) = 0 $,\in O(1)$ nunca é preciso deslocar elementos, ou seja, todos os elementos da sequência são diferentes
PIOR CASO - W(N) =   Número máximo de deslocamentos que é possível efetuar num array com N elementos ocorre quando todos os elementos são iguais exceto o último, ou seja, o ciclo for exterior (i), efetua apenas uma iteração, mas o ciclo interior (j) efetua todas as iterações e em cada iteração realiza o maior número de deslocamentos possíveis. $$W(N) = \sum_{i=0}^{0}\left(\sum_{j=1}^{n-1}(n-1)-j\right) = \frac{(n-2)+\big((n-1)-(n-1)\big)}{2}\cdot(n-1) = \frac{n^2-3n+2}{2} \quad ,\in O(n^2)$$
NOTA: Não é contabilizado o deslocamento em array[n-1] para um array de tamanho n porque se o conteúdo da última posição do array for igual a outro valor presente no array ocorre um deslocamento que não nos interessa pois o array irá diminuir de tamanho uma unidade logo esse valor não vai ser copiado.

APRESENTAÇÃO DO ALGORITMO

```
//função adicional para efectuar deslocamentos
void moveElements(int array[], int inicial, int final)
{
    for(int i=inicial;i<final;i++){
        array[i] = array[i+1];
        numShifts++;                //contar número de deslocamentos
    }
}


void eliminateRepeated(int array[], int *n)
{
    assert(*n > 1);
    numComp = 0;
    numShifts = 0;
    int k = *n;
    for(int i=0;i<k-1;i++) {
        for(int j=i+1;j<k;j++){
            numComp++;              //contar número de comparações
            while(array[i]==array[j]){
                moveElements(array, j, k);
                numShifts--;        //o último deslocamento não conta
                numComp++;          //contar número de comparações
                k--;
            }
        }
    }
    *n = k;
}
```