NOME:	João Diogo Videira Oliveira	**N.º MEC:**	93295

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?

A primeira sequência: {1, 3, 4, 5, 5, 6, 7, 7, 8, 9} é o melhor caso, pois apenas é efetuada uma operação, em que o resultado da subtração do segundo número da sequência com o primeiro é diferente de 1.

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

As 2 últimas sequências (9ª e 10ª): {1, 2, 3, 4, 5, 6, 7, 8, 9, 9} e {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, respetivamente, são os piores casos, pois é onde o array é todo percorrido, efetuando o maior número de operações (9). Na 8ª sequência não se verifica a propriedade devido ao último elemento. Já na 9ª sequência a condição não falha, percorrendo da mesma forma todos os elementos do array.

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

$$\frac{1+2+3+4+5+6+7+8+9+9}{10} = \frac{54}{10} = 5{,}4$$

- Qual é a ordem de complexidade do algoritmo?

O(n), 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.

ANÁLISE FORMAL DO ALGORITMO

MELHOR CASO - B(N) = 1

PIOR CASO - W(N) =

$$1 + \sum_{i=2}^{n-1} 1 = \; 1 + (n-2) = n-1$$

CASO MÉDIO - A(N) =

$$\frac{1}{n}\left[\left(\sum_{i=0}^{n-2}(i+1)\right) + n - 1\right] = \frac{1}{n}\left[\frac{n-1}{2}(1+n-1) + n - 1\right] = \frac{n^2-n}{2n} + \frac{n-1}{n} = \frac{n-1}{2} + \frac{n-1}{n}$$

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

$B(10) = 1$

$W(10) = 10 - 1 = 9$

$A(10) = \frac{10-1}{2} + \frac{10-1}{10} = \frac{9}{2} + \frac{9}{10} = 5{,}4$

Valor Teórico = Valor experimental

APRESENTAÇÃO DO ALGORITMO

```
int isValid(int* a, int n) {

    assert(n > 1);

    for(int i=0; i<n-1; i++) {
        nOper++;
        if(a[i+1] - a[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:	3	3	3	3	3	3	3	3	3	3	N.º de comparações:	9
Final:	3										N.º de Cópias:	36

Justifique a sua resposta:

O melhor caso do número de comparações efetuadas corresponde a uma sequência com os elementos todos iguais, visto que neste caso é efetuado o número mínimo de comparações para 10 elementos, ou seja 9 comparações.
O melhor caso do número de comparações corresponde ao pior caso do número de cópias (36), pois como os elementos são todos iguais apenas um vai ficar na sequência final.

- 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:	3	5	8	9	6	4	1	7	0	2	N.º de comparações:	45
Final:	3	5	8	9	6	4	1	7	0	2	N.º de Cópias:	0

Justifique a sua resposta:

O pior caso do número de comparações efetuadas corresponde a uma sequência com os elementos todos diferentes, visto que neste caso é efetuado o número máximo de comparações para 10 elementos, ou seja 45 comparações. Isto também acontece quando os dois últimos elementos do array são iguais, sendo também o pior caso.
O pior caso do número de comparações corresponde ao melhor caso do número de cópias (0), pois como os elementos são todos diferentes, a sequência final vai ser igual à inicial.

- 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) =

$$\sum_{i=0}^{0}\left(\sum_{j=1}^{n-1} 1\right) = n-1$$

PIOR CASO - W(N) =

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

ANÁLISE FORMAL DO ALGORITMO - NÚMERO DE DESLOCAMENTOS DE ELEMENTOS

MELHOR CASO - B(N) = 0

PIOR CASO - W(N) =

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

APRESENTAÇÃO DO ALGORITMO

```
void removeDuplicates(int* a, int* arraySize) {

    int n = *arraySize;

    assert(n > 1);

    for(int i=0; i<n; i++) {
        for(int j=i+1; j<n; j++) {
            nComp++;
            if(a[i] == a[j]) {
                for(int k=j; k<n-1; k++) {
                    nCopias++;
                    a[k] = a[k+1];
                }
                j--;
                n--;
            }
        }
    }
```

```
    *arraySize = n;
}
```