NOME: SOFIA TEIXEIRA VAZ **Nº MEC: 92968**

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.

Array	Resultado	Número 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 tipo de sequência na qual estamos perante o melhor caso é aquela na qual o segundo elemento do array não é igual ao primeiro elemento incrementado 1 (primeiro array da tabela), uma vez que o ciclo é quebrado após uma comparação.

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

R: O pior caso do algoritmo ocorre quando os primeiros (size-1) elementos obedecem à progressão aritmética, sendo irrelevante se o último pertence ou não (últimos dois arrays da tabela), uma vez que ambos os ciclos são feitos até ao fim.

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

R: 5,4

- Qual é a ordem de complexidade do algoritmo?

R: O algoritmo é 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. **Faça as análises no verso da folha.**

R: Melhor caso: 1, pior caso: n-1, caso médio: $\frac{N-1}{2}+\frac{N-1}{N}$

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

R: 5,4 (que é igual ao valor obtido experimentalmente)

APRESENTAÇÃO DO ALGORITMO

```
int checkSequence(int *pInt, int size) {
    assert(size > 1);
    for (int i = 1; i < size; ++i) {
        numberOfSums++;
        if (pInt[i] - pInt[i - 1] != 1) {
            return 0;
        }
    }
```

ANÁLISE FORMAL DO ALGORITMO

MELHOR CASO - B(N) = 1

PIOR CASO - W(N) = N-1

CASO MÉDIO - A(N) = $\frac{N-1}{2}+\frac{N-1}{N}$

Cálculo:

(foi assumido que todos os casos são equiprováveis, isto é, é igualmente provável que cada valor faça parte da sequência ou não)

$$\frac{N-1+\sum_{i=1}^{N-1} i}{N}=\frac{N-1+\frac{(N-1)\times(N-1+1)}{2}}{N}=\frac{N-1+\frac{(N-1)\times N}{2}}{N}=\frac{N-1}{2}+\frac{N-1}{N}$$

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}
  Final: {1}
  Número de comparações: 9
  Número de cópias: 36

Justifique a sua resposta:

R: Neste caso, as únicas comparações serão entre o primeiro elemento e os n-1 outros. Assim, este caso constitui um melhor caso.

- 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: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
  Final: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
  Número de comparações: 45
  Número de cópias: 0

Justifique a sua resposta:

R: Neste caso, cada elemento tem de ser comparado com todos os que estão numa posição mais acima do array. Com isso, este é um dos piores casos do número de comparações.

- 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.**

Comparações: B(n)=n-1, W(n) $= \frac{n^2-n}{2}$

Deslocações: B(n)= 0 ,W(n)= $\frac{(n-1)(n-2)}{2}$

APRESENTAÇÃO DO ALGORITMO

```
void checkSequence(int arr[], int *size) {
    int n = *size;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; ++j) {
            comps++;
            if (arr[i] == arr[j]) {
                for (int k = j; k < n - 1; ++k) {
                    moves++;
                    arr[k] = arr[k + 1];
                }
                n--;
                j--;
            }
        }
    }
    *size = n;
}
```

ANÁLISE FORMAL DO ALGORITMO

Nº DE COMPARAÇÕES

MELHOR CASO - B(N) = N-1

Quando estamos perante o melhor caso de comparações, iremos comparar o primeiro elemento com todos os outros.

PIOR CASO - W(N) = $\frac{n^2-n}{2}$

Quando estamos perante o pior caso de comparações, iremos comparar todos os elementos com todos os que estão numa posição acima no array.

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

Nº DE DESLOCAMENTOS DE ELEMENTOS

MELHOR CASO - B(N) = 0

No melhor caso não haverá deslocação de elementos.

PIOR CASO - W(N) = $\frac{(n-1)(n-2)}{2}$

Um exemplo de pior caso é aquele no qual o array tem todos os elementos iguais. Na primeira iteração, é necessário transformar um array de tamanho n num de tamanho n-1, movendo os últimos n-2 elementos. O número de elementos a mover vai diminuir por 1 até à última iteração. Nessa, temos um array de 2 elementos e comprimimo-lo para passar a ser unitário, fazendo uma deslocação.

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