AULA 5 - ANÁLISE DA COMPLEXIDADE DE ALGORITMOS

***** Entregue, num ficheiro ZIP, este guião preenchido e o código desenvolvido *****

1 – Considere uma sequência (*array*) de **n valores reais**. Pretende-se determinar se os elementos da sequência são sucessivos termos de uma **progressão geométrica**:

$$\mathbf{r = a[\,1\,] \,/\, a[\,0\,]} \quad \text{e} \quad \mathbf{a[\,i\,] = r \times a[\,i-1\,],\ i > 1.}$$

- Implemente uma função **eficiente** (utilize um algoritmo em lógica negativa) e **eficaz** que verifique se os n elementos (n > 2) de uma sequência de valores reais são sucessivos termos de uma progressão geométrica. A função deverá devolver 1 ou 0, consoante a sequência verificar ou não essa propriedade. **Depois de validar o algoritmo apresente a função no verso da folha.**

- Pretende-se determinar experimentalmente a **ordem de complexidade do número de multiplicações e divisões** efetuadas pelo algoritmo e envolvendo elementos da sequência.

- Considere as seguintes sequências de 10 elementos, que cobrem as distintas situações possíveis de execução do algoritmo. Determine, para cada uma delas, se satisfazem a propriedade e qual o número de operações de multiplicação e de divisão efetuadas pelo algoritmo.

1	2	3	4	5	6	7	8	9	10
1	2	4	4	5	6	7	8	9	10
1	2	4	8	5	6	7	8	9	10
1	2	4	8	16	6	7	8	9	10
1	2	4	8	16	32	7	8	9	10
1	2	4	8	16	32	64	8	9	10
1	2	4	8	16	32	64	128	9	10
1	2	4	8	16	32	64	128	256	10
1	2	4	8	16	32	64	128	256	512

Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	1

Nº de operações	2
Nº de operações	3
Nº de operações	4
Nº de operações	5
Nº de operações	6
Nº de operações	7
Nº de operações	8
Nº de operações	9
Nº de operações	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 é a que corresponde ao melhor caso do algoritmo porque é a que tem um menor número de comparações. Só teve de comparar os dois primeiros elementos do array.

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

A primeira sequência é a que corresponde ao melhor caso do algoritmo porque é a que tem um maior número de comparações. Tem de comparar todos os elementos do array.

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

No caso medio do algoritmo para n = 10, são efetuadas 5,89 operações.

- Qual é a ordem de complexidade do algoritmo?

É um algoritmo 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.** Deve obter expressões matemáticas exatas e simplificadas. **Faça essas análises no verso da folha.**

FUNÇÃO

```
int func(int *arr, int size){
    assert(size > 2);
    NUM_COMP = 0;
    int res = 1;
    int r = arr[1]/arr[0];
    for(int i = 1; i< size; i++){
        int val = r*arr[i-1];
        NUM_COMP++;
        if(arr[i] != val){
            res = 0;
            break;
        }
    }
    return res;
}
```

ANÁLISE FORMAL DO ALGORITMO

Melhor caso (B(N)): 2

Pior caso (W(N)): N-1

Caso Médio (A(N)): $\frac{1}{n}((\sum_{i=1}^{n-1} i)(n-1)) = \frac{n^2+n-2}{2n}$

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

Calculando o valor das expressões para n = 10, obtemos como resultado, 5,89, que é igual ao resultado obtido experimentalmente.

2 – Considere uma sequência (array), possivelmente não ordenada, de n elementos inteiros e positivos. Pretende-se **eliminar os elementos da sequência que sejam iguais ou múltiplos ou submúltiplos de algum dos seus predecessores**, sem fazer a sua ordenação e sem alterar a posição relativa dos elementos.

Por exemplo, a sequência { 2, 2, 2, 3, 3, 4, 5, 8, 8, 9 } com 10 elementos será transformada na sequência { 2, 3, 5 } com 3 elementos; e a sequência { 7, 8, 2, 2, 3, 3, 3, 8, 8, 9 } com 10 elementos será transformada na sequência { 7, 8, 3, } com 3 elementos.

- Implemente uma função **eficiente** e **eficaz** que elimina os elementos iguais ou múltiplos ou submúltiplos de algum dos seus predecessores 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 a função no verso da folha.

- Pretende-se determinar experimentalmente a **ordem de complexidade do número de comparações** e **do número de deslocamentos** (i.e., cópias) efetuados pelo algoritmo e 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? Justifique.

Inicial:	1	1	1	1	1	1	1	1	1	1		Nº de comparações	9
Final:	1											Nº de cópias	36

Neste caso, assim como em todos os arrays com todos os elementos iguais, apenas será feita uma comparação por elemento pois, este será imediatamente eliminado.

- 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? Justifique.

Inicial:	3	5	7	11	13	17	23	29	31	33		Nº de comparações	45
Final:	3	5	7	11	13	17	23	29	31	33		Nº de cópias	0

Neste caso, um array constituído por números primos, todos os elementos terão de ser comparados com os seus predecessores, não sendo nenhum dos elementos eliminado durante o processo.

- Determine formalmente a ordem de complexidade do algoritmo nas situações do **melhor caso** e do **pior caso**, considerando uma sequência de tamanho n. Deve obter expressões matemáticas exatas e simplificadas. **Faça essas análises no verso da folha.**

FUNÇÃO

```
void func(int *arr, int *size){
    NUM_COMPS = 0;
    copias = 0;
    assert(*size>1);
    for (int i = 0; i < *size; i++){

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

            NUM_COMPS++;
            if (arr[i] == arr[h] || arr[i]%arr[h]==0 || arr[h]%arr[i]==0){

                (*size)--;
                for (int j = h; j < *size; j++){
                    copias++;
                    arr[j]=arr[j+1];
                }
                h--;
            }
        }
    }
}
```

ANÁLISE FORMAL DO ALGORITMO – COMPARAÇÕES – MELHOR CASO – PIOR CASO

Melhor caso B(N) = N-1

Pior caso W(N) = $\sum_{i=0}^{n-1}\left(\sum_{h=i+1}^{n-1} 1\right) = n^2 - \frac{n\,(n-1)}{2}$

ANÁLISE FORMAL DO ALGORITMO – DESLOCAMENTOS – MELHOR CASO – PIOR CASO

Melhor caso B(N) = 0

Pior caso W(N) = $\sum_{i=1}^{n-2} i = \frac{(n-2)\,(n-1)}{2}$