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 sequência {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} corresponde ao melhor caso do algoritmo em relação ao número de operações, uma vez que o algoritmo acaba quando encontra uma condição que não satisfaz a progressão geométrica, logo teremos o menor número de operações efetuadas.

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

A sequência {1, 2, 4, 8, 16, 32, 64, 128, 256, 10} é a sequência que corresponde ao pior caso do algoritmo, uma vez que efetua todas as comparações, mas o resultado é 0, visto que o último elemento da sequência não faz parte da progressão geométrica.

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

$$A_c(10) = \frac{2+3+4+5+6+7+8+9+9}{10} = \frac{53}{10} = 5{,}3$$

O caso médio é calculado com a médias das operações realizadas através da análise experimental do algoritmo e admitindo que todos os números das operações são equiprováveis.

- Qual é a ordem de complexidade do algoritmo?

A ordem de complexidade do algoritmo será linear, ou seja, O(N).
$\frac{T(2N)}{T(N)} = 2$

- **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 func1(int* a, int n)
{
   assert (n > 2);
   int res = 1;              // elemento neutro da conjuncao
   int r = a[1] / a[0];      // razao da progressão geométrica

   numComparacoes++;

   for (int i = 2; i < n; i++)
   {
      numComparacoes++;

      if (a[i] != r*a[i-1])
      {
         res = 0;
         break;
      }
   }

   return res;

}
```

ANÁLISE FORMAL DO ALGORITMO

- Melhor Caso:
  $B(N) = 2$, uma vez que o melhor caso corresponde ao número mínimo de operações efetuadas apesar de o resultado dar 0.

- Pior Caso:
  $W(N) = N - 1$, uma vez que o pior caso corresponde ao número máximo de operações efetuadas apesar de o resultado dar 0.

- Caso Médio:
  $A(N) = \sum_{i=0}^{N-1} \frac{1}{2} = \frac{N}{2}$

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

Para n=10, o caso médio será aproximadamente 5, logo podemos dizer que $A_C(N) = \frac{N}{2}$.
Para o melhor caso do número de comparações será $B_C(N) = 2$ e para o pior caso será $W_C(N) = N - 1$.
Comparando os resultados obtidos na análise formal com os resultados da análise experimental do algoritmo conclui-se, então, para n=10, $A(10) = \frac{10}{2} = 5$.
Relativamente ao pior caso, $W(10) = 10 - 1 = 9$, o que está de acordo com os resultados obtidos experimentalmente. Isso acontece quando, por exemplo, o array está com todos os seus elementos de uma progressão geométrica, à exceção do último que não pertence a essa progressão.
Podemos também concluir que para o melhor caso do número de comparações que $B(10) = 2$, o que significa que apenas os primeiros 2 elementos do array apresentado satisfazem a condição de pertencerem à progressão geométrica.

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

Uma sequência que conduz ao melhor caso do número de comparações é quando a sequência é composta apenas por elementos que são todos iguais. Podemos concluir que B(N) = N-1.

- 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:	2	3	5	7	11	13	17	19	23	29	Nº de comparações	45
Final:	2	3	5	7	11	13	17	19	23	29	Nº de cópias	0

O pior caso no número de comparações corresponde a uma sequência onde os elementos não têm qualquer relação entre eles e que não existe qualquer número repetido.

- 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 func2(int* a, int* n)
{
   int s = *n;
   assert (s > 1);

   for (int i = 0; i < s; i++)
   {
      for (int j = i+1; j < s; )
      {
         numOperacoes++;
         if ((a[j] % a[i]) == 0 )
         {
            numCopias++;
            for (int k = j; k < s; k++)
            {
               a[k] = a[k+1];
            }
            s--;        //retira valor
         } else {

            if(a[i]%a[j]== 0) {
               numCopias++;
               for (int k = j; k < s; k++)
               {
                  a[k] = a[k+1];
               }
               s--;        //diminui o tamanho do array
            }
            else
            {
               j++;        //acrescenta valor
            }
         }
      }
   }

   *n = s;

   //Para imprimir o array alterado:
   printf("Array Final: ");
   printf("{ ");
   for (int x = 0; x < *n; x++)
   {
      printf("%d ", a[x]);
   }
   printf("}\n");

   printf("Total de copias: %d \n", numCopias);
   printf("Numero de Operacoes: %d \n\n", numOperacoes);

}
```

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

- Melhor Caso: $B_C(N) = N - 1$, ou seja, o melhor caso acontece quando os elementos de um determinado array são todos iguais.

- Pior Caso: $W_C(N) = \sum_{i=0}^{N-1} i = \frac{N(N-1)}{2}$, ou seja, para o pior caso, acontece quando todos os elementos do array são diferentes, por exemplo, uma sequência de apenas números primos, irão fazer $\frac{N(N-1)}{2}$ comparações entre esses elementos.

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

- Melhor Caso: $B(N) = 0$, ou seja, o melhor caso para o número de deslocamentos efetuados acontece quando não existe nenhum número repetido, daí o número de cópias ser 0.

- Pior Caso: $W(N) = N - 1$, ou seja, o pior caso para o número de deslocamentos efetuados acontece quando todos os elementos de um determinado array estão todos relacionados uns com outros. Logo, serão N-1 cópias do mesmo valor.