AULA 4 – ANÁLISE DA COMPLEXIDADE DE ALGORITMOS

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

1 - Seja uma dada sequência (*array*) de n elementos inteiros e não ordenada. Pretende-se determinar quantos elementos da sequência respeitam a seguinte propriedade:

array [i] = array [i – 1] + array [i + 1], para 0 < i < (n – 1)

- Implemente uma **função eficiente** e **eficaz** que determine quantos elementos (resultado da função) de uma sequência com n elementos (sendo n > 2) respeitam esta propriedade.
  Depois de validar o algoritmo apresente a função no verso da folha.

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

- Considere as seguintes sequências de 10 elementos inteiros, que cobrem algumas situações possíveis de execução do algoritmo.
  Determine, para cada uma delas, o número de elementos que obedecem à condição e o número de comparações efetuadas, envolvendo elementos da sequência.

1	2	3	4	5	6	7	8	9	10
1	2	1	4	5	6	7	8	9	10
1	2	1	3	2	6	7	8	9	10
0	2	2	0	3	3	0	4	4	0
0	0	0	0	0	0	0	0	0	0

Resultado	0
Resultado	1
Resultado	2
Resultado	6
Resultado	8

Nº de operações	8
Nº de operações	8
Nº de operações	8
Nº de operações	8
Nº de operações	8

Depois dos testes experimentais responda às seguintes questões:

- Em termos do número de comparações efetuadas, podemos distinguir alguma variação na execução do algoritmo? Ou seja, existe a situação de melhor caso e de pior caso, ou estamos perante um algoritmo com caso sistemático?

Estamos perante um caso sistemático em termos do número de comparações efetuadas, uma vez que não conseguimos distinguir o melhor do pior caso. O número de operações efetuadas varia de acordo com o número de elementos existentes no array.

- Com base nos resultados experimentais, qual é a ordem de complexidade do algoritmo? Justifique.

A ordem de complexidade do algoritmo é O(n-2), ou seja, é de complexidade linear.

$$\frac{T(2N)}{T(N)} = 2$$

- Determine formalmente a ordem de complexidade do algoritmo. Tenha em atenção que deve obter uma expressão matemática exata e simplificada.
  Faça a análise no verso da folha.

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

Para n=10, o resultado é n-2 = 10-2 = 8, o que está de acordo com os resultados obtidos experimentalmente.

NOME: LARA CATARINA DA SILVA MATOS Nº MEC: 95228

FUNÇÃO

```
int somaArray(int* array, int n)
{
   assert (n > 2);
   int res = 0;

   for (int i = 1; i < (n - 1); i++)
   {
      numComparacoes++;
      if (array[i] == array[i-1] + array[i+1])       // se a propriedade se verificar…
      {
         res++;
      }
   }
   return res;

}
```

ANÁLISE FORMAL DO ALGORITMO

$$\sum_{i=1}^{(n-1)-1} 1 = 1 * [(n-1)-1] = n-2$$

Podemos então comprovar que o algoritmo é de complexidade linear.

2 - Seja uma dada sequência (*array*) de n elementos inteiros e não ordenada. Pretende-se determinar quantos ternos **(i, j, k)** de índices da sequência respeitam a seguinte propriedade:

array [k] = array [i] + array [j], para i < j < k

- Implemente uma **função eficiente** e **eficaz** que determine quantos ternos **(i, j, k)** de índices (resultado da função) de uma sequência com n elementos (sendo n > 2) respeitam esta propriedade. **Depois de validar o algoritmo apresente a função no verso da folha.**

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

- Considere as sequências anteriormente indicadas de 10 elementos inteiros e outras sequências diferentes à sua escolha**; use sequências com 5, 10, 20, 30 e 40 elementos**. Determine, para cada uma delas, quantos ternos **(i, j, k)** de índices respeitam propriedade e o número de comparações efetuadas.

Depois dos testes experimentais responda às seguintes questões:

- Em termos do número de comparações efetuadas, podemos distinguir alguma variação na execução do algoritmo? Ou seja, existe a situação de melhor caso e de pior caso, ou estamos perante um algoritmo com caso sistemático?

Estamos perante um caso sistemático em termos do número de comparações efetuadas, uma vez que não conseguimos distinguir o melhor do pior caso. O número de operações efetuadas varia de acordo com o número de elementos existentes no array.

- Com base nos resultados experimentais, qual é a ordem de complexidade do algoritmo? Justifique.

A ordem de complexidade do algoritmo é O(n^3), ou seja, é de complexidade cúbica.

$\frac{T(2N)}{T(N)}$ é aproximadamente igual a 8.

- Determine formalmente a ordem de complexidade do algoritmo. Tenha em atenção que deve obter uma expressão matemática exata e simplificada. **Faça a análise no verso da folha.**

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

Para n=10, o resultado obtido experimentalmente foi 120.

Usando a expressão matemática obtida no ponto anterior, para n=10: $\frac{10^3}{6} - \frac{10^2}{2} + \frac{10}{3} = 120$

Logo, podemos concluir que o resultado experimental está de acordo com o resultado obtido usando a expressão da análise formal do algoritmo.

FUNÇÃO

```
int sequenciaArray(int *array, int n)
{
    assert (n > 2);
    int res = 0;

    for (int k = 2; k < n; k++)
    {
        for (int j = 1; j < k; j++)
        {
            for (int i = 0; i < j; i++)
            {
                numComparacoes++;
                if (array [k] == array [i] + array [j])
                {
                    res++;
                }
            }
        }
    }

    return res;

}
```

ANÁLISE FORMAL DO ALGORITMO

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

$$\sum_{k=1}^{n-1}\frac{k^2}{2} - \sum_{k=1}^{n-1}\frac{k}{2} - \sum_{k=1}^{1}\frac{k^2-k}{2} =$$

$$= \frac{1}{2}*\frac{1}{6}(n-1)\big((n-1)+1\big)(2(n-1)+1) - \frac{1}{2}*\frac{1}{2}(n-1)(n-1+1) - 0 =$$

$$\frac{1}{12}(n-1)n(2n-1) - \frac{1}{4}(n-1)n = \frac{2n^3-6n^2+4n}{12} =$$

$$= \frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3}$$