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?

Não

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

Ordem de complexidade é O(n), linear

- 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, ou seja, um array com 10 elementos, o resultado será 8

NOME: **Nº MEC:**

FUNÇÃO

```
int main(void){
	int n = 2;
	int c = 0;
	int ne = 0;

	// Array input
	printf("Size of the array: ");
	scanf("%d", &n);
	int a[n];
	for(int i = 0; i < n; i++){
		printf("n[%d]: ", i);
		scanf("%d", &a[i]);
	}

	// Array operations
	for(int i = 1; i <= n-2; i++){
		c++;
		if(a[i] == (a[i-1] + a[i+1])){
			ne++;
		}
	}

	printf("Nº de elementos: %d\nNº de comparações: %d\n", ne, c);
}
```

ANÁLISE FORMAL DO ALGORITMO

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

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 algoritmo com caso sistemático, qualquer que seja o array, a comparação será sempre realizada

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

A ordem de complexidade é de $O(n^3)$
(n=20) / (n=10) = 9.5
(n=40) / (n=20) = 8.67
(n=80) / (n=40) = 8.3
(n=160) / (n=80) = 8.15

A divisão de 2n/n aproxima-se de 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.

N = 10
Nº ternos: 20
Nº comparações: 120

NOME: **Nº MEC:**

FUNÇÃO

```
int ne = 0; // contador de ternos
int c = 0; // contador de comparações
for(int k = 2; k < n; k++){
        for(int j = 1; j < k; j++){
                for(int i = 0; i < j; i++ ){
                        c++;
                        if(a[k] == a[i] + a[j]){
                                ne++;
                        }
                }
        }
}
```

ANÁLISE FORMAL DO ALGORITMO

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

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

$$= \frac{1}{2}\left(\frac{(n-1)(n)(2n-1)}{6} - \frac{(n-1)(n+1)}{2}\right) = \frac{1}{2}\left(\frac{(n^2-n)(2n-1)}{6} - \frac{(n^2-1)}{2}\right) =$$

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

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

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