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 |  | Resultado | 0 |  | Nº de operações | 8 |
| 1 | 2 | 1 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  | Resultado | 1 |  | Nº de operações | 8 |
| 1 | 2 | 1 | 3 | 2 | 6 | 7 | 8 | 9 | 10 |  | Resultado | 2 |  | Nº de operações | 8 |
| 0 | 2 | 2 | 0 | 3 | 3 | 0 | 4 | 4 | 0 |  | Resultado | 6 |  | Nº de operações | 8 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | Resultado | 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 existe a situação de melhor ou pior caso porque o número de operações apenas depende exclusivamente do número de elementos do array. Para um array do mesmo tamanho, o número de operações é o mesmo independentemente do conteúdo deste. |

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

|  |
| --- |
| A ordem de complexidade é O(n), porque o número de elementos do array tem a mesma ordem do número de operações. O valor do número de operações vai aumentando linearmente com o aumento do tamanho do array. |

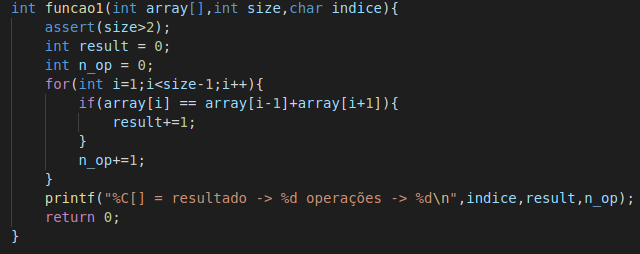
* Determine formalmente a ordem de complexidade do algoritmo. Tenha em atenção que deve obter uma expressão matemática exata e simplificada.

**Faça a análise no verso da folha.**

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

|  |
| --- |
| O(n) para n = 10 :  num\_op = 10 - 2  num\_op = 8  Podemos verificar assim que o resultado coincide com o resultado obtido experimentalmente. |

Função



Análise Formal do Algoritmo

|  |
| --- |
| Como estamos perante um algoritmo sistemático e com 1 ciclo ‘for’, podemos assim chegar a este algoritmo para chegar ao número de operações :  O(n) = , ‘n’ representa o número de elementos do array. |

**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?

|  |
| --- |
| Não existe a situação de melhor ou pior caso porque o número de operações depende exclusivamente do número de elementos do array. Para um array do mesmo tamanho, o número de operações é o mesmo independentemente do conteúdo deste. |

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

|  |
| --- |
| A ordem de complexidade é O(n³).  Como o algoritmo é sistemático podemos verificar a ordem de complexidade dividindo T(2N) por T(N) :  - T(2N) array com 40\*2 elementos tem 82160 operações.  - T(N) array com 40 elementos tem 9880 operações.    Podemos assim verificar que estamos perante ordem de complexidade de O(n³). |

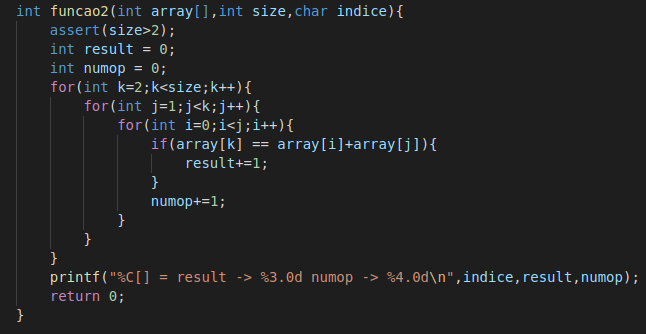
* Determine formalmente a ordem de complexidade do algoritmo. Tenha em atenção que deve obter uma expressão matemática exata e simplificada.

**Faça a análise no verso da folha.**

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

|  |
| --- |
| O(n) para n = 10 :  Podemos verificar assim que o resultado coincide com o resultado obtido experimentalmente. |

Função



Análise Formal do Algoritmo

|  |
| --- |
| Como estamos perante um algoritmo sistemático e com 3 ciclos ‘for’, podemos assim chegar a este algoritmo para chegar ao número de operações :    Cada somatório é equivalente a cada ciclo ‘for’ do algoritmo.  Podemos assim simplificar a expressão da seguinte forma:      O(n³) = , onde ‘n’ representa o número de elementos do array. |