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?

|  |
| --- |
| Neste caso estamos perante um algoritmo com caso sistemático, pois o número de comparações é igual para todos os casos. |

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

|  |
| --- |
| Algoritmo com ordem de complexidade 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.

|  |
| --- |
| Uma imagem com texto, relógio  Descrição gerada automaticamenteCom esta expressão obtemos o número de operações para n=10. Experimentalmente obtemos os mesmos valores como mostra na tabela acima. |

Função

**int** func(**int** **\***arr, **int** size){

    assert(size **>** 2);

**int** certo **=** 0;

    NUM\_COMPS **=** 0;

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

**int** val **=** arr[i**-**1] **+** arr[i**+**1];

**if**(arr[i] **==** val){

            certo **+=** 1;

        }

        NUM\_COMPS **+=**1;

    }

**return** certo;

}

Análise Formal do Algoritmo

|  |
| --- |
| Sendo este um algoritmo de caso sistemático temos:  Uma imagem com texto, relógio, manómetro  Descrição gerada automaticamente  PIOR CASO-W(N) =  CASO MEDIO-A(N) =  MELHOR CASO-B(N) = |

**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?

|  |
| --- |
| Vai ter sempre o mesmo número de comparações independentemente da ordem dos valores. O número de operações só muda se aumentarmos/diminuirmos o tamanho do array, logo estamos perante um algoritmo com caso sistemático. |

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

|  |
| --- |
| Com base nos resultados podemos verificar que o ratio tende para 8 quando duplicamos o número de elementos do array.  É um algoritmo exponencial de ordem 3.  Logo: |

* 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;  Uma imagem com texto, relógio  Descrição gerada automaticamente  Este resultado é idêntico ao resultado simulação. |

Função

**int** func(**int** **\***arr, **int** n){

    assert(n **>** 2);

    NUM\_COMPS **=** 0;

**int** certos **=** 0;

**for**(**int** k **=** 2; k **<** n; k**++**){

**for**(**int** j **=** 1; j **<** k; j**++**){

**for**(**int** i **=** 0; i **<** j; i**++**){

**int** val **=** arr[i] **+** arr[j];

**if**(arr[k] **==** val){

                    certos **+=** 1;

                }

                NUM\_COMPS **+=** 1;

            }

        }

    }

**return** certos;

}

Análise Formal do Algoritmo

|  |
| --- |
| Sendo este um algoritmo de caso sistemático temos:  Uma imagem com texto, dispositivo, manómetro  Descrição gerada automaticamente  PIOR CASO-W(N) =  CASO MEDIO-A(N) =  MELHOR CASO-B(N) = |