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**:

**r = a[ 1 ] / a[ 0 ]** e **a[ i ] = r × 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 |  | Resultado | **0** |  | Nº de operações | **2** |
| 1 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  | Resultado | **0** |  | Nº de operações | **3** |
| 1 | 2 | 4 | 8 | 5 | 6 | 7 | 8 | 9 | 10 |  | Resultado | **0** |  | Nº de operações | **4** |
| 1 | 2 | 4 | 8 | 16 | 6 | 7 | 8 | 9 | 10 |  | Resultado | **0** |  | Nº de operações | **5** |
| 1 | 2 | 4 | 8 | 16 | 32 | 7 | 8 | 9 | 10 |  | Resultado | **0** |  | Nº de operações | **6** |
| 1 | 2 | 4 | 8 | 16 | 32 | 64 | 8 | 9 | 10 |  | Resultado | **0** |  | Nº de operações | **7** |
| 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 9 | 10 |  | Resultado | **0** |  | Nº de operações | **8** |
| 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 10 |  | Resultado | **0** |  | Nº de operações | **9** |
| 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |  | Resultado | **1** |  | 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 que corresponde ao melhor caso é: {1, 2, 3, 4, 5, 6, 7, 8, 9,10}, sendo Bop(n) = 2** |

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

|  |
| --- |
| **A sequência que correspondem ao pior caso são:**   * **{1, 2, 4, 8, 16, 32, 64, 128, 256, 10}** * **{1, 2, 4, 8, 16, 32, 64, 128, 256, 512}** * **Wop(n) = 9** |

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

|  |
| --- |
| **Aop** |

* Qual é a ordem de complexidade do algoritmo?

|  |
| --- |
| **A ordem de complexidade do algoritmo desenvolvido é O(n) linear, porque o nº de operações no pior caso é 9, i.e, n-1.** |

* **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 & Análise Formal do Algoritmo

|  |
| --- |
| ***Melhor caso*: a ordem complexidade é constante**  **Isto acontece sempre que o 3º elemento não é igual ao 2º elemento vezes a razão definida pelos dois primeiros**  ***Pior caso:* a ordem de complexidade é linear , isto é, Wop(n) = n-1,**  **Isto acontece quando todos elementos são percorridos verificando-se no final do array se é ou não uma PG**  ***Caso médio:***   * + 1. **Os casos possíveis vão de I1 até In-1 em que I1 corresponde a situação do melhor caso e In-1, In-2 correspondem as situações do pior caso, que fazem maior nº de operações**     2. **Para I1 são feitas duas operações, para I2 são feitas três operações, etc**     3. **Para In-1, In-2 são feitas (n-1)**     4. **A probabilidade é idêntica para todos os casos p = 1/(n-1)**   **Assim,**  **Aop(n) =**  **Aop(n) =** |

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

|  |
| --- |
| **Aop(9) =**  **O resultado obtido experimentalmente é 5,88 o que comprova o valor obtido formalmente com pequena diferença.** |

**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: | **2** | **2** | **2** | **2** | **2** | **2** | **2** | **2** | **2** | **2** |  | Nº de comparações | **9** |
| Final: | **2** |  |  |  |  |  |  |  |  |  |  | Nº de cópias | **45** |

|  |
| --- |
| **Se todos os elementos começando no segundo forem iguais ou múltiplos ou divisores do primeiro,**  **só é feita uma comparação para cada um elemento que seguem ao primeiro.** |

* 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 | **90** |
| Final: | **2** | **3** | **5** | **7** | **11** | **13** | **17** | **19** | **23** | **29** |  | Nº de cópias | **0** |

|  |
| --- |
| **Se os números forem primos (não múltiplos e nem divisores) não há cópias efetuadas e é efetuado o nº máximo de comparações.** |

* 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

Análise Formal do Algoritmo – Comparações – Melhor Caso – Pior Caso

|  |
| --- |
| ***Melhor caso*: a ordem de complexidade é O(n) linear, i.e, Bcomp(n) = n-1, ou seja, o melhor caso acontece quando todos elementos de um array são todos iguais.**  **Podemos provar o resultado obtido calculando Bcomp(10) = 10-1 = 9**  ***Pior caso*: a ordem de complexidade é O(n2) quadrático.**  **Wcomp(n) =**  **Wcomp(n) =**  **O pior caso acontece quando todos elementos de um array são todos números primos.**  **Podemos provar o resultado obtido calculando Wcomp(10) = 10(10-1)=10x9 = 90** |

Análise Formal do Algoritmo – Deslocamentos – Melhor Caso – Pior Caso

|  |
| --- |
| ***Melhor caso*: a ordem de complexidade é O(n2) quadrático .**  **Bcopies(n) =**  **O melhor caso acontece quando todos elementos de um array são todos iguais.**  **Podemos provar o resultado obtido calculando Bcopies(10) =**  ***Pior caso:* a ordem de complexidade é constante O(1)**  **O pior caso acontece quando não há cópias a serem feitas e é efetuado o nº máximo de comparações, i.e, quando todos números de um array são primos.** |