|  |  |  |  |
| --- | --- | --- | --- |
| NMec: | 88194 | Nome: | Gil Lopes Teixeira |

**Aula 3 – Análise da Complexidade de Algoritmos**

**1 -** Seja uma dada sequência (*array*) de n elementos inteiros. Pretende-se determinar quantos elementos da sequência são diferentes do seu elemento anterior. Ou seja:

array [i] ≠ array [i–1], para i > 0

* Implemente uma **função** **eficiente** e **eficaz** que determine quantos elementos (resultado da função) de uma sequência com n elementos (sendo n > 1) respeitam esta propriedade.

**Depois de validar o algoritmo apresente-o no verso da folha.**

* Determine experimentalmente a **ordem de complexidade do número de comparações** efetuadas pelo algoritmo e envolvendo elementos da sequência. Considere as seguintes 10 sequências de 10 elementos inteiros, todas diferentes, e que cobrem as distintas 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.

|  |  |  |
| --- | --- | --- |
| **Sequência** | **Resultado** | **N.º de operações** |
| {3, 3, 3, 3, 3, 3, 3, 3, 3, 3} | 9 | 9 |
| {4, 3, 3, 3, 3, 3, 3, 3, 3, 3} | 8 | 9 |
| {4, 5, 3, 3, 3, 3, 3, 3, 3, 3} | 7 | 9 |
| {4, 5, 1, 3, 3, 3, 3, 3, 3, 3} | 6 | 9 |
| {4, 5, 1, 2, 3, 3, 3, 3, 3, 3} | 4 | 9 |
| {4, 5, 1, 2, 6, 3, 3, 3, 3, 3} | 3 | 9 |
| {4, 5, 1, 2, 6, 8, 3, 3, 3, 3} | 1 | 9 |
| {4, 5, 1, 2, 6, 8, 7, 3, 3, 3} | 5 | 9 |
| {4, 5, 1, 2, 6, 8, 7, 9, 3, 3} | 2 | 9 |
| {4, 5, 1, 2, 6, 8, 7, 9, 3, 0} | 0 | 9 |

**Depois da execução do algoritmo 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?
* R: Este algoritmo efetua sempre n – 1 comparações para um dado array de n elementos e é portanto um caso sistemático.
* Qual é a ordem de complexidade do algoritmo?
* R: O(n - 1) O(n), Complexidade 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.
* R: para N = 10, são efetuadas N – 1 = 9 comparações. Todos os arrays que testei experimentalmente tinham N = 10 e efetuaram-se sempre 9 comparações.

**Apresentação do Algoritmo**

|  |
| --- |
| int find\_dups(int a[], int aSize) {  int repetitions = 0;  for(int i = 1; i < aSize; i++) {  if(a[i] == a[i-1]) repetitions++;  }  return repetitions;  } |

**Análise Formal do Algoritmo**

**E(n) = =**

**2 -** Seja uma dada sequência (*array*) de n elementos inteiros e não ordenada. Pretende-se determinar qual é o primeiro elemento da sequência que tem mais elementos menores do que ele atrás de si, e indicar a posição (índice do *array*) onde esse elemento se encontra.

Por exemplo, na sequência { 1, 9, 2, 8, 3, 4, 5, 3, 7, 2 } o elemento 7, que está na posição de índice 8 da sequência, é maior do que 6 elementos seus predecessores. Na sequência { 1, 7, 4, 6, 5, 2, 3, 2, 1, 0 } o elemento 6, que está na posição de índice 3 da sequência, é maior do que 2 elementos seus predecessores. Mas, na sequência { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 } nenhum elemento é maior do que qualquer um dos seus predecessores, pelo que deve ser devolvido -1 como resultado.

* Implemente uma **função** **eficiente** e **eficaz** que determine o índice do primeiro elemento (resultado da função) de uma sequência com n elementos (sendo n > 1) que tem o maior número de predecessores menores do que ele.

**Depois de validar o algoritmo apresente-o no verso da folha.**

* Determine experimentalmente a **ordem de complexidade do número de comparações** efetuadas envolvendo elementos da sequência. Considere as sequências anteriormente indicadas de 10 elementos inteiros e outras sequências diferentes à sua escolha. Determine, para cada uma delas, o índice do elemento procurado e o número de comparações efetuadas.
* R: Independentemente do índice do elemento procurado, caso sistemático, o total de comparações por tamanho, n, do array segue:

|  |  |
| --- | --- |
| Tamanho | Comparações |
| 2 | 3 |
| 3 | 6 |
| 4 | 10 |
| 5 | 15 |
| 6 | 21 |
| 7 | 28 |
| 8 | 36 |
| 9 | 45 |
| 10 | 55 |

Expectativa: n + (n-1) \* n/2

**Depois da execução do algoritmo 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?
* R: O algoritmo é um algoritmo sistemático para um dado tamanho do conjunto de elementos.
* Qual é a ordem de complexidade do algoritmo?
* R: O algoritmo é de ordem quadrática. O( ).
* 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.
* R: O valor obtido para N = 10 é de 55 comparações que corresponde aos resultados esperados.

**Apresentação do Algoritmo**

|  |
| --- |
| int firstGreat(int a[], int aSize) {  assert(aSize > 1);  int i, j, currentCount;  int maxCount = 0;  int maxId = -1;  for(i = 0; i < aSize; i++) {  currentCount = 0;  for(j = i-1; j >= 0; j--) {  if(a[i] > a[i-1]) {  currentCount++;  }  }  if(currentCount > maxCount) {  maxCount = currentCount;  maxId = i;  }  }  return maxId;  } |

**Análise Formal do Algoritmo**

**E(n) = = = =**