|  |  |  |  |
| --- | --- | --- | --- |
| **NMec:** | **92948** | **Nome:** | **Raquel Resende Milheiro Pinto** |

**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} | 0 | 9 |
| {4, 3, 3, 3, 3, 3, 3, 3, 3, 3} | 1 | 9 |
| {4, 5, 3, 3, 3, 3, 3, 3, 3, 3} | 2 | 9 |
| {4, 5, 1, 3, 3, 3, 3, 3, 3, 3} | 3 | 9 |
| {4, 5, 1, 2, 3, 3, 3, 3, 3, 3} | 4 | 9 |
| {4, 5, 1, 2, 6, 3, 3, 3, 3, 3} | 5 | 9 |
| {4, 5, 1, 2, 6, 8, 3, 3, 3, 3} | 6 | 9 |
| {4, 5, 1, 2, 6, 8, 7, 3, 3, 3} | 7 | 9 |
| {4, 5, 1, 2, 6, 8, 7, 9, 3, 3} | 8 | 9 |
| {4, 5, 1, 2, 6, 8, 7, 9, 3, 0} | 9 | 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?

Este logaritmo é de um caso sistemático, pois independentemente do array que dermos à função, o número de comparações vai ser sempre igual ao número de elementos do array menos um (n-1). Ou seja, só depende do número de elementos (n) do array.

* Qual é a ordem de complexidade do algoritmo?

A ordem de complexidade deste algoritmo é O(n), ou seja, é um algoritmo com ordem de 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.

= 10 – 1 = 9 comparações

O valor da expressão para N = 10 foi 9 comparações, assim como nos resultado obtidos experimentalmente.

**Apresentação do Algoritmo**

|  |
| --- |
| static int ncomparacoes = 0; // numero de comparacoes  int diferencaArray(int a[],int n){ // array, tamanho do array  assert (n>1);  int count =0; // Resultado (numero de elementos diferentes no array)  ncomparacoes=0; // Numero de comparacoes  for(int i = 1;i<n;i++){  ncomparacoes++;  if(a[i]!= a[i-1]){  count++;  }  }  return count;  } |

**Análise Formal do Algoritmo**

**E(n) =** = N - 1

Efetuando a analise formar deste algoritmo, podemos ver que tem ordem de complexidade linear - O(N). Atendendo ao número de comparações realizadas podemos ver que é um algoritmo sistemático.

**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.

|  |  |  |
| --- | --- | --- |
| **Sequência** | **Resultado(posição do array)** | **N.º de operações** |
| {1, 9, 2, 8, 3, 4, 5, 3, 7, 2} | 8 | 45 |
| {1, 7, 4, 6, 5, 2, 3, 2, 1, 0} | 3 | 45 |
| {2, 2, 2, 2, 2, 2, 2, 2, 2, 2} | -1 | 45 |
| {4, 5, 1, 2, 6, 8, 7, 9, 3, 0} | 7 | 45 |
| {9, 5, 8, 4, 7, 1, 2, 5, 8, 4} | 8 | 45 |
| {1, 2, 2, 2, 2, 2, 2, 2, 5, 6} | 9 | 45 |
| {9, 8, 7, 6, 5, 4, 3, 2, 1, 0} | -1 | 45 |
| {5, 8, 4, 2, 1, 3, 0, 8, 5, 6} | 9 | 45 |
| {8, 0, 9, 7, 5, 4, 4, 5, 4, 1} | 7 | 45 |
| {0, 9, 8, 7, 7, 5, 4, 3, 2, 1} | 1 | 45 |

**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?

Este logaritmo é de um caso sistemático, pois independentemente do array que dermos à função, o número de comparações vai ser sempre igual a. Ou seja, só depende do numero de elementos (n) do array.

* Qual é a ordem de complexidade do algoritmo?

A ordem de complexidade deste algoritmo é O() ou seja, este algoritmo tem ordem de complexidade quadrática.

* 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.

**E(10)=====**

O valor da expressão para N = 10 foi 45 comparações, assim como nos resultado obtidos experimentalmente.

**Apresentação do Algoritmo**

|  |
| --- |
| static int ncomparacoes = 0; // numero de comparacoes  int countelemtmaiores(int a[], int n){ // array e tamanho do array  assert (n>1);  int max = 0;  int posicao = -1; // posicao do elemento com mais elementos anteriores inferiores a ele. comeca em -1 porque assim quando nenhum elemento é maior do que qualquer um dos seus predecessores devolve -1  ncomparacoes=0; // Numero de comparacoes  for(int i = 1; i<n;i++){  int count= 0; //numero maximo de numeros inferiores a um determinado elemento do array  for(int j = 0;j<i;j++){  ncomparacoes++;  if(a[j]<a[i]){  count++; // conta o numeros menores que determinado elemento do array  }  }  if(count>max){  max = count; // atualiza o valor maximo de numeros menores  posicao =i;  }  }  return posicao;  } |

**Análise Formal do Algoritmo**

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

Efetuando a analise formar deste algoritmo, podemos ver que é de ordem de complexidade quadrática - O(). Atendendo ao número de comparações realizadas podemos ver que é um algoritmo sistemático.