|  |  |  |  |
| --- | --- | --- | --- |
| **NMec:** | **93020** | **Nome:** | **Roberto Oliveira Graça** |

**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?
* R: O número de comparações não varia na execução do algoritmo. Se o número de elementos for N, o número de comparações será N-1. Estamos perante um algoritmo com caso sistemático.
* Qual é a ordem de complexidade do algoritmo?
* R: A complexidade do algoritmo é linear, pois o algoritmo para N elementos, efetua N-1 comparações. O tempo decorrido é diretamente proporcional ao número de elementos. Logo, a ordem de complexidade é O(N).
* 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 somatório de i=2 até N de 1, é o mesmo que 1 \* N-1, logo para N = 10, o valor da expressão será 9. Comparando, com os valores obtidos das sequências propostas, verifica-se que estes possuem 10 elementos e apresentam 9 comparações realizadas.

**Apresentação do Algoritmo**

|  |
| --- |
| int checkDifferent(int a[],int n){  unsigned int i=1,cont=0;  for(;i<n;i++){  if(a[i]!=a[i-1]){  cont++;  }  nops++;  }  return cont;  } |

**Análise Formal do Algoritmo**

**E(n) = Somatório de I=2 até N de 1**

**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.

**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 número de comparações não varia na execução do algoritmo. Se o número de elementos for N, o número de comparações será (N(N+1)/2)-N. Estamos perante um algoritmo com caso sistemático.
* Qual é a ordem de complexidade do algoritmo?
* R: A complexidade do algoritmo é quadrática, pois o algoritmo para N elementos, efetua (N(N+1)/2)-N comparações. Logo, a ordem de complexidade é O(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.
* O somatório de k=1 até N-1 de k, é o mesmo que 1+2+...+N-2+N-1, logo para N = 10, o valor da expressão será 1+2+3+4+5+6+7+8+9 = 45. Comparando, com os valores obtidos das sequências propostas, verifica-se que estes possuem 10 elementos e apresentam 45 comparações realizadas.

**Apresentação do Algoritmo**

|  |
| --- |
| int getIndex(int a[],int n){  unsigned int i=1,j=0,max=0,cont=0,index=0;  for(;i<n;i++){  cont=0;  for(j=0;j<i;j++){  if(a[j]<a[i]){  cont++;  }  nops++;  }  if(cont>max){  max=cont;  index=i;  }  }  if(index<=0){  return -1;  }  else{  return index;  }  } |

**Análise Formal do Algoritmo**

**E(n) = somatório de i=1 até N-1 de I**