Nome: Lúcia Maria Bessa de Sousa Nº mec: 93086 Turma: P1

Aula 4 - Análise da Complexidade de Algoritmos

**1 –** Considere uma sequência (*array*) de n elementos inteiros, ordenada por **ordem não decrescente**. Pretende-se determinar se a sequência é uma **progressão aritmética de razão 1**, i.e., a[i+1] – a[i] = 1.

* Implemente uma função **eficiente** (utilize um algoritmo em lógica negativa) e **eficaz** que verifique se uma sequência com n elementos (n > 1) define uma sequência contínua de números. A função deverá devolver 1 ou 0, consoante a sequência verificar ou não essa propriedade.

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

* Determine experimentalmente a **ordem de** **complexidade do número de adições/subtrações** 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, se satisfaz a propriedade e qual o número de operações de adição/subtração efetuadas pelo algoritmo.

|  |  |  |
| --- | --- | --- |
| **Sequência** | **Resultado** | **N.º de operações** |
| {1, 3, 4, 5, 5, 6, 7, 7, 8, 9} | 0 | 1 |
| {1, 2, 4, 5, 5, 6, 7, 8, 8, 9} | 0 | 2 |
| {1, 2, 3, 6, 8, 8, 8, 9, 9, 9} | 0 | 3 |
| {1, 2, 3, 4, 6, 7, 7, 8, 8, 9} | 0 | 4 |
| {1, 2, 3, 4, 5, 7, 7, 8, 8, 9} | 0 | 5 |
| {1, 2, 3, 4, 5, 6, 8, 8, 9, 9} | 0 | 6 |
| {1, 2, 3, 4, 5, 6, 7, 9, 9, 9} | 0 | 7 |
| {1, 2, 3, 4, 5, 6, 7, 8, 8, 9} | 0 | 8 |
| {1, 2, 3, 4, 5, 6, 7, 8, 9, 9} | 0 | 9 |
| {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} | 1 | 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?

{1, 3, 4, 5, 5, 6, 7, 7, 8, 9}

Sequências cuja diferença entre o segundo e o primeiro elemento seja diferente de um, ou seja, (a[1] - a[0] != 1).

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

{1, 2, 3, 4, 5, 6, 7, 8, 9, 9} e {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Sequências cuja diferença de dois elementos seja sempre 1, ou que apenas a diferença dos dois últimos elementos seja zero, ou seja, (a[n-1] - a[n-2] != 1).

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

Caso Médio [A(10)] = = = 5

* Qual é a ordem de complexidade do algoritmo?

A ordem de complexidade deste algoritmo é liner, O(n).

* 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. Tenha em atenção que deve obter expressões matemáticas exatas e simplificadas. **Faça as análises no verso da folha.**
* Calcule o valor das expressões para n = 10 e compare-os com os resultados obtidos experimentalmente.

Melhor caso: B(10) = 1

Pior caso: W(10) = n-1 = 10-1 = 9

Caso médio: A(10) = = = 5

Experimentalmente, as sequências que correspondiam ao pior caso eram {1, 2, 3, 4, 5, 6, 7, 8, 9, 9} e {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, o número de operações era 9, tal como o valor calculado.

A sequência que correspondia ao melhor caso era {1, 3, 4, 5, 5, 6, 7, 7, 8, 9}, B(10) = 1, e experimentalmente foi esse o resultado obtido.

O caso médio será aproximadamente 5, tanto experimentalmente como obtendo pela expressão.

Apresentação do Algoritmo

***int* isValid(*int* \* a*rr*,*int* *n*){**

**assert(n > 1);**

***int* i;**

***int* count = 0;**

***int* ncomp = 0;**

***int* res = 0;**

**for(i = 0; i < n-1; i++){**

**ncomp++;**

**if(arr[i+1]-arr[i] != 1){**

**return 0;**

**}**

**}**

**return 1;**

**}**

Análise Formal do Algoritmo

Melhor Caso - B(n) = 1

Pior Caso - W(n) = N-1

Caso Médio - A(n) =

**2 –** Considere uma sequência (array) não ordenada de n elementos inteiros. Pretende-se eliminar os elementos repetidos existentes na sequência, sem fazer uma pré-ordenação e sem alterar a posição relativa dos elementos. Por exemplo, a sequência { 1, 2, 2, 2, 3, 3, 4, 5, 8, 8 } com 10 elementos será transformada na sequência { 1, 2, 3, 4, 5, 8 } com apenas 6 elementos. Por exemplo, a sequência { 1, 2, 2, 2, 3, 3, 3, 3, 8, 8 } com 10 elementos será transformada na sequência { 1, 2, 3, 8 } com apenas 4 elementos. Por exemplo, a sequência { 1, 2, 3, 2, 1, 3, 4 } com 7 elementos será transformada na sequência { 1, 2, 3, 4 } com apenas 4 elementos. Mas, a sequência { 1, 2, 5, 4, 7, 0, 3, 9, 6, 8 } permanece inalterada.

* Implemente uma função **eficiente** e **eficaz** que elimina os elementos repetidos 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-o no verso da folha.**

* Determine experimentalmente a **ordem de** **complexidade do número de comparações** e **do número de deslocamentos** 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?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inicial | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Final | 1 |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Nº de comparações | 9 |
| Nº de cópias | 36 |

Justifique a sua resposta: O array inicial tem todos os elementos iguais, o algoritmo compara dois elementos e remove um deles, faz deslocamento, o array diminui o tamanho, tornando o número de comparações mais pequeno.

* 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?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inicial | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Final | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

|  |  |
| --- | --- |
| Nº de comparações | 45 |
| Nº de cópias | 0 |

Justifique a sua resposta: A sequência inicial tem todos os elementos diferentes, o algoritmo compara dois elementos e não remove, não faz deslocamento, não tornando o array menor, sendo assim o número de comparações será maior.

Determine formalmente a ordem de complexidade do algoritmo nas situações do **melhor caso** e do **pior caso**, considerando uma sequência de tamanho n. Tenha em atenção que deve obter expressões matemáticas exatas e simplificadas. **Faça as análises no verso da folha**Apresentação do Algoritmo

***void* removeDuplicates(int\* input,int\* n){**

**int i,j,k;**

***int* ncomp = 0;**

***int* ndesl = 0;**

**for(i = 0; i < \*n; i++){**

**for(j = i + 1; j < \*n; j++){**

**ncomp++;**

**if(input[j] == input[i]){**

**for(k=j;k<(\*n)-1;k++){**

**ndesl++;**

**input[k]=input[k+1];**

**}**

**j--;**

**(\*n)--;**

**}**

**}**

**}**

**}**

Análise Formal do Algoritmo

Nº de Comparações

Melhor Caso - B(n) = N-1

Pior Caso - W(n) =

Nº de Deslocamentos de Elementos

Melhor Caso - B(n) = 0

Pior Caso - W(n) =