**Nome: Roberto Oliveira Graça**

**NºMec: 93020**

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.



1

0

0

0

0

0

0

0

0

1

0

2

3

4

5

6

7

8

9

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?

R: A primeira sequência.

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

R: A nona e a décima sequência.

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

R: A(n) = (n+2)(n-1)/2n, logo, para n=10, A(10)=5.4.

* Qual é a ordem de complexidade do algoritmo?
* R: O algoritmo tem ordem de complexidade linear.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 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.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Apresentação do Algoritmo

int isArithmetic1(int a[],int n){

unsigned int i=0;

for(;i<n-1;i++){

nops++;

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

return 0;

}

}

return 1;

}

Análise Formal do Algoritmo

Melhor Caso - B(n) = 1, para n=10, o melhor caso será 1.

Pior Caso - W(n) = N-1, para n=10, o pior caso será 9.

Caso Médio - A(n) = (n+2)(n-1)/2n, para n=10 o caso médio será 5.4.

**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?



1 1 1 1 1 1 1 1 1 1

1

9

36

Justifique a sua resposta: O melhor caso é onde houver menor número de comparações. Logo, se na primeira vez que se percorrer o array os restantes elementos forem iguais ao primeiro, são eliminados. Só se conta as comparações entre o primeiro elemento e os restantes.

* 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?



0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

45

0

Justifique a sua resposta: O pior caso é onde houver o maior número de comparações. Logo, se a sequência dada não possuir elementos a serem removidos, ou seja, se os elementos forem todos diferentes, têm de se realizar as comparações todas entre o primeiro e os seguintes, entre o segundo e os seguintes, etc.

* 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

unsigned static int nComps=0,nShifts=0;

void repRemove(int \*x, int \*n){

assert(\*n>1);

int \*a;

int \*b;

int \*c;

for(a=x;a<x+\*n;a++){

for(b=a+1;b<x+\*n;b++){

nComps++;

if(\*a==\*b){

for(c=b;c<x+\*n-1;c++){

nShifts++;

\*c = \*(c+1);

}

(\*n)--;

b--;

}

}

}

}

Análise Formal do Algoritmo

Nº de Comparações

Melhor Caso - B(n) = N-1, para N=10 será 9.

Pior Caso - W(n) = somatório de I=1 até N-1 de I, para N=10 será 45.

Nº de Deslocamentos de Elementos

Melhor Caso - B(n) = 0, para N=10, será 0.

Pior Caso - W(n) = somatório de I=2 até N de I-2, para N=10 será 36.