JNOME: JOAQUIM PEDRO GONÇALVES ANDRADE Nº MEC:93432

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?

  O melhor caso são as sequências em que a diferença entre os 2 primeiros números é diferente de 1.

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

  O pior caso são as sequências em que a sequência é uma progressão aritmética de razão 1, ou quando apenas os 2 últimos elementos da sequência não têm uma diferença de 1.

- Determine o número de adições efetuadas no caso médio do algoritmo (**para n = 10**).
  6

- Qual é a ordem de complexidade do algoritmo?

É 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 isValid( int* a, int size){

   assert(size>1);
   int a=1, count=0;
   for (int i = 0; i < size-1; i++)
   {
      count++;
      if ((argc[i] + 1) != argc[i+1])
      {
         a = 0;
         break;
      }
   }
   return a;
}
```

ANÁLISE FORMAL DO ALGORITMO

MELHOR CASO - B(N) = 1

PIOR CASO - W(N) = n − 1

CASO MÉDIO - A(N) =1/N(N-1+N-1)
=1/N(2N-2)

B(10)=1, O MELHOR CASO TEM 1 COMPARAÇÃO.
B(10)=9, O PIOR CASO É 9
B(10)=

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	2	2	2	2	2	2	2	2	2
Final	2								

Nº de comparações	8
Nº de cópias	28

Justifique a sua resposta: Quando o inicial tem os números todos iguais, o código elimina um por um, fazendo apenas a comparação do primeiro termo com n-1 termos seguintes.

- 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
Final	1	2	3	4	5	6	7	8	9

Nº de comparações	36
Nº de cópias	0

Justifique a sua resposta: Embora não tenha deslocações, quando os números iniciais são todos diferentes, o algoritmo percorre os dois ciclos for na totalidade, comparando todos os termos.

- 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 remove_duplicates(int *givenarray, int size)
{
   int countdesl = 0, countcomparacoes = 0;

   for (int i = 0; i < (size - 1); i++)
   {
      for (int j = (i + 1); j < size; j++)
      {
         countcomparacoes++;
         if (givenarray[i] == givenarray[j])
         {
            for (int h = j; h < (size - 1); h++)
            {
               givenarray[h] = givenarray[h + 1];
               countdesl++;
            }
            size--;
            j--;
         }
      }
   }
}
```

ANÁLISE FORMAL DO ALGORITMO

Nº DE COMPARAÇÕES
MELHOR CASO - B(N) = N-1

PIOR CASO - W(N) = (N-2)^2 = O(N^2)

Nº DE DESLOCAMENTOS DE ELEMENTOS
MELHOR CASO - B(N) = 0

PIOR CASO - W(N) = N^3