AULA 5 - ANÁLISE DA COMPLEXIDADE DE ALGORITMOS

1 – Considere uma sequência (*array*) de **n valores reais**. Pretende-se determinar se os elementos da sequência são sucessivos termos de uma **progressão geométrica**:

$$\mathbf{r = a[\,1\,] \,/\, a[\,0\,]} \quad \text{e} \quad \mathbf{a[\,i\,] = r \times a[\,i-1\,],\ i > 1.}$$

- Implemente uma função **eficiente** (utilize um algoritmo em lógica negativa) e **eficaz** que verifique se os n elementos (n > 2) de uma sequência de valores reais são sucessivos termos de uma progressão geométrica. A função deverá devolver 1 ou 0, consoante a sequência verificar ou não essa propriedade. **Depois de validar o algoritmo apresente a função no verso da folha.**

- Pretende-se determinar experimentalmente a **ordem de complexidade do número de multiplicações e divisões** efetuadas pelo algoritmo e envolvendo elementos da sequência.

- Considere as seguintes sequências de 10 elementos, que cobrem as distintas situações possíveis de execução do algoritmo. Determine, para cada uma delas, se satisfazem a propriedade e qual o número de operações de multiplicação e de divisão efetuadas pelo algoritmo.

1	2	3	4	5	6	7	8	9	10
1	2	4	4	5	6	7	8	9	10
1	2	4	8	5	6	7	8	9	10
1	2	4	8	16	6	7	8	9	10
1	2	4	8	16	32	7	8	9	10
1	2	4	8	16	32	64	8	9	10
1	2	4	8	16	32	64	128	9	10
1	2	4	8	16	32	64	128	256	10
1	2	4	8	16	32	64	128	256	512

Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	0
Resultado	1

Nº de operações	2
Nº de operações	3
Nº de operações	4
Nº de operações	5
Nº de operações	6
Nº de operações	7
Nº de operações	8
Nº de operações	9
Nº de operações	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?

Das sequências apresentadas a que corresponde ao melhor caso do algoritmo é a primeira, {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. Sendo o número de operações realizadas 2, B(10) = 2.

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

Das sequências apresentadas a que corresponde ao pior caso do algoritmo é a primeira, {1, 2, 4, 8, 16, 32, 64, 128, 256, 10}. Sendo o número de operações realizadas 9, W(10) = 9 = N-1. Este é pior caso porque realiza o maior número de comparações e o último número não pertence à sucessão.

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

Com todas as sequências acima equiprováveis:

$$A_C = \frac{2+3+4+5+6+7+8+9+9}{10} = \frac{53}{10} = 5{,}3$$

- Qual é a ordem de complexidade do algoritmo?

A Ordem de Complexidade do algoritmo é linear, O(N).

$$\frac{T(2N)}{T(N)} = 2$$

- **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.** Deve obter expressões matemáticas exatas e simplificadas. **Faça essas análises no verso da folha.**

FUNÇÃO

```
int condEval(int *array, int n)
{
   assert (n > 2);

   nOp++;
   int r = array[1]/array[0]; //r = a[ 1 ] / a[ 0 ]

   for (int i = 2; i < n; i++)
   {
      nOp++;
      if (array[i] != array[i-1] * r)
      {
         return 0;
      }
   }
   return 1;

}
```

ANÁLISE FORMAL DO ALGORITMO

- Melhor Caso:
  $B(N) = 2$
  Com o mínimo de operações verifica-se o melhor caso, mesmo que o resultado seja 0.

- Pior Caso:
  $W(N) = N - 1$
  Com o máximo de operações e o resultado a 0, verifica-se o pior caso.

- Caso Médio:
  $$A(N) = \sum_{i=0}^{N-1} \frac{1}{2} = \frac{N}{2}$$

- Calcule o valor das expressões para n = 10 e **compare-os com os resultados obtidos experimentalmente**.

$B(10) = 2$

$W(10) = 10 - 1 = 9$

$A(10) = \frac{10}{2} = 5$

2 – Considere uma sequência (array), possivelmente não ordenada, de n elementos inteiros e positivos. Pretende-se **eliminar os elementos da sequência que sejam iguais ou múltiplos ou submúltiplos de algum dos seus predecessores**, sem fazer a sua ordenação e sem alterar a posição relativa dos elementos.

Por exemplo, a sequência { 2, 2, 2, 3, 3, 4, 5, 8, 8, 9 } com 10 elementos será transformada na sequência { 2, 3, 5 } com 3 elementos; e a sequência { 7, 8, 2, 2, 3, 3, 3, 8, 8, 9 } com 10 elementos será transformada na sequência { 7, 8, 3, } com 3 elementos.

- Implemente uma função **eficiente** e **eficaz** que elimina os elementos iguais ou múltiplos ou submúltiplos de algum dos seus predecessores 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 a função no verso da folha.

- Pretende-se determinar experimentalmente a **ordem de complexidade do número de comparações** e **do número de deslocamentos** (i.e., cópias) efetuados pelo algoritmo e 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? Justifique.

Inicial:	1	1	1	1	1	1	1	1	1	1
Final:	1									

Nº de comparações	9
Nº de cópias	9

Com os números todos relacionados, repetições ou múltiplos, verificamos o melhor caso do número de comparações, este passa a ser N-1, ocorrendo o seu valor mínimo.

- 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? Justifique.

Inicial:	2	3	5	7	11	13	17	19	23	29
Final:	2	3	5	7	11	13	17	19	23	29

Nº de comparações	45
Nº de cópias	0

Com os números todos não relacionados, sem repetições nem múltiplos, verificamos o pior caso do número de comparações, este passa para $\frac{N(N-1)}{2}$, ocorrendo o seu valor máximo.

- Determine formalmente a ordem de complexidade do algoritmo nas situações do **melhor caso** e do **pior caso**, considerando uma sequência de tamanho n. Deve obter expressões matemáticas exatas e simplificadas. **Faça essas análises no verso da folha.**

FUNÇÃO

```
void dupeElem(int *array, int *n)
{
    int s = *n;
    assert(s > 1);
    // Array Processing
    for (int i = 0; i < s; i++)
    {
        for ( int j = i+1 ; j < s;)
        {
            nComp++;
            if((array[j] % array[i]) == 0){
                nShift++;

                for(int k = j; k < s; k++){
                    array[k] = array[k+1];
                }

                s--; // Retira o Valor
            }else{
                if(array[i] % array[j] == 0){
                    nShift++;

                    for(int k = j; k < s; k++){
                        array[k] = array[k+1];
                    }
                    s--;
                }else{
                    j++;
                }
            }
        }
    }

    *n = s;
}
```

ANÁLISE FORMAL DO ALGORITMO – COMPARAÇÕES – MELHOR CASO – PIOR CASO

- Melhor Caso:
  $B(N) = N - 1$, acontece quando os elementos do array são repetições e múltiplos de um número.

- Pior Caso:
  $W(N) = \sum_{i=0}^{N-1} i \ = \ \frac{N(N-1)}{2}$, acontece quando os elementos do array não tem múltiplos nem repetições entre si,

ANÁLISE FORMAL DO ALGORITMO – DESLOCAMENTOS – MELHOR CASO – PIOR CASO

- Melhor Caso:
  $B(N) = 0$, os elementos do array são todos diferentes e não há múltiplos nem submúltiplos. Não sendo assim necessário realizar cópias.

- Pior Caso:
  $W(N) = N - 1$, os elementos são todos iguais ou múltiplos e submúltiplos entre si. Sendo necessário realizar o máximo de cópias, N-1.