**Estruturas de Dados**
Ficha Laboratorial Nº 2
Engenharia Informática – Instituto Superior de Engenharia de Coimbra

## *Pesquisa Binária e Variações*

O método `criaArrayCom(int valor, int dimensao, boolean diferentes)` aqui apresentado devolve um *array* de inteiros ordenado (todos diferentes se o parâmetro `diferentes` for verdadeiro), com a dimensão especificada, que inclui o valor indicado.

```
static int[] criaArrayCom(
            int valor,
            int dimensao,
            boolean diferentes){
      int m[]=new int[dimensao];
      if(diferentes){
            for(int i=0;i<dimensao;i++)
                  m[i]=i*10;
            if((valor%10!=0)||(0>valor)||(valor>(dimensao-1)*10))
                  m[0]=valor;
      }
      else{
            Random r=new Random();
            int gama=(Math.abs(valor)<10)?10:Math.abs(valor);
            for(int i=0;i<dimensao;i++)
                  m[i]=r.nextInt(gama*4)-gama*2;
            m[0]=valor;
      }
      Arrays.sort(m);
      return m;
      }
```

***Use este método para testar a resolução das alíneas seguintes.***

1 – Construa um método que efetua uma pesquisa binária de forma **recursiva**, devolvendo `true` ou `false` conforme o valor seja encontrado ou não. A função deve receber como parâmetros a chave e a tabela. Calcule a complexidade temporal e espacial do algoritmo.

2 – Construa um método que efetua uma pesquisa binária de forma **iterativa**, devolvendo `true` ou `false` conforme o valor seja encontrado ou não.

3 – Construa um método que efetua uma pesquisa binária. Este método deve devolver a **posição** em que o valor procurado se encontra, ou então -1 caso este não esteja no *array* indicado. Teste o método, conforme o indicado nos exercícios anteriores.

4 – Construa um método que efetua uma pesquisa binária. Este método deve devolver a posição em que o valor procurado se encontra, ou então um valor negativo (-X) caso este não esteja no array indicado. O valor de `abs(X + 1)` deve indicar uma posição em que o valor procurado poderia ser inserido para preservar a ordem. Teste o método, conforme o indicado nos exercícios anteriores.
*Exemplo:*

**Array:** {3,7,12,15}
Valor procurado=15 resultado= 3
Valor procurado=3 resultado= 0
Valor procurado=1 resultado= -1 (o valor deveria ser inserido na posição 0)
Valor procurado=4 resultado= -2 (o valor deveria ser inserido na posição 1)
Valor procurado=10 resultado= -3 (o valor deveria ser inserido na posição 2)
Valor procurado=13 resultado= -4 (o valor deveria ser inserido na posição 3)
Valor procurado=16 resultado= -5 (o valor deveria ser inserido na posição 4)

***Para resolver os exercícios seguintes, tenha em consideração o método de pesquisa binária e as variantes apresentadas nas alíneas anteriores. Todos os exercícios devem ser resolvidos com algoritmos com complexidade O(log N).***

5 – Construa um método que recebe um *array* ordenado de inteiros, todos diferentes, e um valor, e devolve a percentagem de valores do *array* que são menores do que o valor indicado. O algoritmo deve ser de ordem de complexidade logarítmica.
*Exemplo:*
**Array:** {3,7,12,15}
Valor =15 resultado= 0.75
Valor =14 resultado= 0.75
Valor =3 resultado= 0.0
Valor =1 resultado= 0.0
Valor =100 resultado= 1.0

6 – Construa um método que recebe um *array* ordenado de inteiros, todos diferentes, e dois valores que definem uma intervalo. O método deve indicar quantos valores do *array* se encontram dentro do intervalo especificado.
*Exemplo:*
**Array:** {3,7,12,15}
Valores =(0,15) resultado= 4
Valores=(3,7) resultado= 2
Valores =(4,14) resultado= 2
Valores =(4,5) resultado= 0
Valores =(0,100) resultado= 4

7 – Construa um método que recebe um *array* ordenado de inteiros, e um valor. O método deve indicar se esse valor se encontra repetido no *array*.
*Exemplo:*
**Array:** {3,3,7,12,12,15}
Valor =15 resultado= false
Valor =14 resultado= false
Valor =12 resultado= true
Valor =3 resultado= true

8 - Construa um método que recebe por parâmetro um *array* ordenado de inteiros, não repetidos, bem como um inteiro Z, e devolve o **maior** elemento do *array* **menor** do que Z (ou Z se esse elemento não existir).
*Exemplo:*
**Array:** {3,7,12,15}
Valor =15 resultado= 12

Valor =14  resultado= 12
Valor =3  resultado= 3
Valor =1  resultado= 1
Valor =100  resultado= 15

9 – Considere um método que recebe um *array* de inteiros no qual os números estão dispostos da seguinte forma: todos os números negativos se encontram em posições maiores do que os números positivos, e todos os número positivos e negativos se encontram ordenados entre si.
*Exemplo:* {3,6,8, -10,-3,-2,-1}
Construa um método que procura um número no *array*.

10 – Construa um método que recebe um *array* ordenado de inteiros, não repetidos, e que devolve o indice da primeira posição na qual o valor guardado é superior ao índice (ou -1 se esta posição não existir).
*Exemplos:*
**Array:** {3,7,12,15} Resultado=0
**Array:** {-3,1,7,12,15} Resultado=2
**Array:** {-15,-14,1,2,3,4} Resultado= -1