Estruturas de Dados

Teste Laboratorial 1 – 10 de Novembro de 2015

Nome:_____

Número:_____

1 – Considere o seguinte código e indique a complexidade em função de N. Apresente uma breve justificação:

```
for(int i=0;i<N;i++)
      for(int k=0;k<j;k++)
            soma ++;
for(int i=0;i<N;i++)  soma ++;
```

R:

2- Assuma que dispõe de um método `int pesquisa (int m[],int valor)` que efectua uma pesquisa binária. Este método devolve 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)` indica uma posição em que o valor procurado poderia ser inserido para preservar a ordem. **Este método já existe e não precisa de o fazer**. Construa um método `int proximo(int m[],int valor)` que indica qual o menor inteiro do array que é maior do que `valor`. Caso não exista nenhum elemento nestas condições, deve devolver `valor`. O array `m` encontra-se ordenado, e existem, no máximo, duas cópias de cada valor. O método deve ter desempenho logarítmico.

R:

Estruturas de Dados

Teste Laboratorial 1 – 10 de Novembro de 2015

Nome:_____

Número:_____

3 – Construa o protótipo de um método `f` que recebe três parâmetros:

- um parâmetro `lista`, que é um `Arraylist` de um tipo genérico `T`

- um parâmetro `valor1` que pode ser comparado com qualquer valor contido em `lista`, através de uma instrução como `valor1.compareTo(lista.get(0))`.

- um parâmetro `valor2` pode ser comparado com `valor1` através de uma instrução como `valor2.compareTo(valor1)`.

R:

4 – Considere a classe iterável `Pessoa`, que armazena um nome e uma morada, que são inicializados sempre através do seu construtor. Não e possível alterar estes valores após a sua inicialização. Construa um iterador adequado, incluindo suporte para todas as excepções que podem ser lançadas (deve **ignorar** aquelas que não fazem sentido para esta estutura de dados).

R:

Estruturas de Dados

Teste Laboratorial 1 – 10 de Novembro de 2015

Nome:_____

Número:_____

5- Considere o método seguinte.

```
static void duplicaZeros(List<Integer> lista){
ListIterator<Integer> it=lista.listIterator();
while(it.hasNext()){
      Integer i=it.next();
      if(i==0) {
            // insere 0 antes do próximo elemento
            // o proximo next() devolve o valor que se encontra
            // imediatamente a seguir ao valor inserido
            it.add(0);
      }
}
```

Qual é a complexidade do método, no pior caso? (considere que o objecto `lista` tanto pode ser um `ArrayList` como uma `LinkedList`). Justifique a sua resposta.