isep Instituto Superior de Engenharia do Porto

Estruturas de Informação

Exame Época Recurso

4 de Fevereiro 2020

Departamento de Engenharia Informática

Duração 2 horas

Resolva cada exercício em folhas separadas

3.5 ptos

1. Escreva um método que recebe como parâmetros duas listas de strings, uma das quais é uma lista de nomes e a outra é a correspondente lista dos apelidos, e retorna o primeiro nome associado ao maior número de apelidos distintos. Considere as duas seguintes listas:

```
   nomes: [Bob, Mary, Steve, Derek, Mary, Derek, Joe, Derek, Nicole, Mary]
apelidos: [Jones, Ford, Akers, Smith, Giles, Smith, Caiu, Jones, Jones, Stepp]
```

Ao primeiro nome (índice 0, lista nomes) corresponde o primeiro apelido (índice 0, lista apelidos); ao nome no índice 1 corresponde o apelido no índice 1; e assim sucessivamente, ou seja, ambas as listas têm o mesmo comprimento.
Para as listas acima o método deve devolver "Mary" pois Mary tem associado três apelidos (Ford, Giles e Stepp), e todos os outros nomes estão associados apenas a um ou dois **apelidos distintos**.
Considere a seguinte assinatura

```
public static String commonFirstName(List<String> names, List<String> nicknames)
```

Nota: Será valorizada uma resolução o mais eficiente possível e com o menor número de estruturas de dados

3 ptos

2. Considere a seguinte função recebe um vetor ordenado de inteiros

```
public static int mistery(Integer[] a, int n, Integer x) {

    if (a[n-1] < x)
        return n;

    if (a[0] >= x)
        return 0;

    int l=0, u=n-1;
    while (l<u) {
        int m = (l+u)/2;
        if (a[m] < x)
          l = m+1;
        else
          u = m;
    }
    return l;
}
```

a) Explique sucintamente o que o método *mistery* faz e diga qual o resultado quando invocado

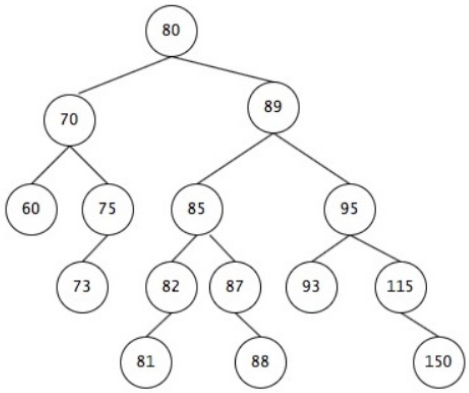
```
Integer[] C = {1,2,4,6,8,10,12,14};
mistery (C, C.length, 10);
```

b) Analise o algoritmo quanto à sua complexidade temporal, utilizando a notação Big-Oh. Justifique adequadamente.

Resolva cada exercício em folhas separadas

5 ptos

3. O **menor ancestral comum** entre um **conjunto de nós** S = {N_1, N_2, …, N_n} é definido como o nó de nível mais baixo na árvore binária de pesquisa T que possui todos os nós do conjunto S como descendentes. Permite-se que um nó seja descendente de si mesmo. Por exemplo na árvore apresentada o menor ancestral do conjunto de elementos {81, 88, 150} é **89**, para o conjunto de elementos {81, 82, 85} é **85**, {60, 81, 115} é **80**.



Implemente o método *findLCA* na classe genérica TREE<E> que devolva o menor ancestral comum de um conjunto de elementos dos nós S = {N_1, N_2, …, N_n}

```
public class Tree<E extends Comparable<E>> extends BST<E> {
      public E findLCA(Set<E> elements){
      }
}
```

5 ptos

4. Considere um grafo que representa a rede metropolitana de uma cidade em que as estações são identificadas por um nome único e as várias linhas são associadas a cores. Note que duas estações podem ser ligadas por mais do que uma linha. Seja a classe Metro uma implementação de uma rede metropolitana usando a classe Map de adjacências.

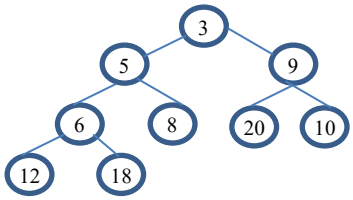
```
public class Metro {
    private Graph<String,String> g = new Graph<>(false);

    public LinkedList<String> pathSameLine(Graph<String,String> g,
        String stOrig, String stDest, String line)
}
```

Escreva uma função que devolva um **caminho entre duas estações** que usa apenas **ligações da mesma cor**. Note que tal caminho entre duas estações pode não existir.

3.5 ptos

5. Implemente um método genérico para ser adicionado na classe HeapPriorityQueue<K,V> que devolve uma lista com os elementos que ficam num dado nível da heap. Para a heap abaixo representada observe os exemplos:



Exemplo: [3,5,9,6,8,20,10,12,18]
Level = 3
List→ {6,8,20,10}
Level = 4
List→ {12,18}

Considere a seguinte assinatura:

```
public List<V> getElemsLevel(int level)
```