isep Instituto Superior de Engenharia do Porto

Estruturas de Informação

Exame Época Normal

22 de Janeiro 2019

Departamento de Engenharia Informática

Duração 2 horas

Resolva cada exercício em folhas separadas

3.5 p^{tos}

1. Pretende-se implementar um validador simples de XHTML que recebe um conjunto de *tags* extraídas de um documento XHTML pela ordem de ocorrência no mesmo. O documento só é válido se todas as *tags* abertas forem fechadas e uma *tag* só pode ser fechada se for a mais recente *tag* aberta. Nota, a *tag* </xpto> fecha a *tag* <xpto>.

Exemplo válido:

["<body>", "<h1>", "</h1>", " ", "<a>", "", " ", "</body>"]

Exemplos inválidos:

["<body>","<h1>","</h1>"," ","<a>"," ",""," ","<body>"]

["<body>","<h1>","</h1>"," ","<a>",""," "]

Elabore um método que recebe um *array* de *strings* com o conjunto de *tags* de um documento e verifica se o documento é ou não válido. Considere a seguinte assinatura:

```
public boolean validaTags(String[] tags)
```

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

3.5 p^{tos}

2. Considere o seguinte código:

```
public static int mistery(int[] a){

    int max = a[1]-a[0];

    for (int j = 2; j < a.length; j++)
        for (int i = 0; i < j; i++)
            if (a[j] - a[i] > max)
               max = a[j]-a[i];
    return max;
}
```

a) Explique o que faz o método `mistery` acima apresentado e indique o valor devolvido com o seguinte *array* de entrada: {3, 1, 4, 1, 5, 9, 2}

b) Analise a função quanto à sua complexidade temporal. Justifique.

5 p^{tos}

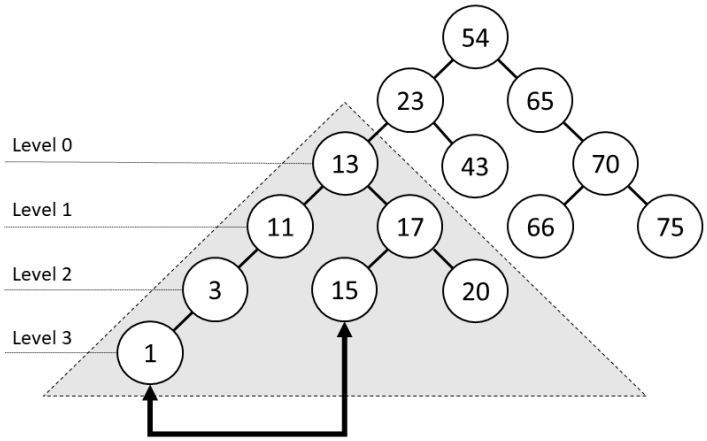
3. A árvore de cobertura de custo mínimo de um grafo G não direcionado, conexo, com pesos positivos nos ramos é uma árvore que liga todos os vértices de G ao menor custo (somatório dos pesos dos ramos). O **algoritmo de *Kruskall*** devolve a árvore de cobertura de custo mínimo para um grafo original. Este algoritmo começa por criar um grafo com todos os vértices isolados e adiciona na árvore os ramos do grafo original por ordem crescente do seu peso, se esse ramo não criar um ciclo. Implemente o algoritmo *Kruskall* de forma genérica, para ser adicionado na classe `GraphAlgorithms`, para tal considere a seguinte assinatura:

```
public static<V,E> Graph<V,E> kruskall(Graph<V,E> g)
```

Resolva cada exercício em folhas separadas

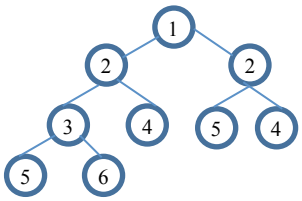
5 p^tos

4. Adicione à classe `TREE<E>` um método que devolva a estrutura da **sub-árvore completa mínima** que contém dois nodos passados por parâmetro. A estrutura devolvida dever ser do tipo `Map<Integer, List<E>>`. Por exemplo na árvore indicada abaixo, dados os nodos 1 e 15, a estrutura devolvida deve conter {0: {13}, 1: {11, 17}, 2: {3, 15, 20}, 3: {1}}.



3 p^tos

5. Implemente um método genérico para ser adicionado na classe `HeapPriorityQueue<K,V>` que devolve uma lista com os elementos que ficam no caminho entre um elemento com o índice idx e o topo da *heap*. Para a *heap* abaixo representada observe os dois exemplos:




```
Exemplos:
          [1,2,2,3,4,5,4,5,6]
idx=8 → {6,3,2,1}
idx=5 → {5,2,1}
```

Considere a seguinte assinatura:

```
public List<V> getElemsPath(int idx)
```