Practical Class 4

Graphs: Adjacency Map Structure

Estruturas de Informação

1. Represent the network of the European airports and the flight routes between them using a directed graph. For simplicity, consider that each airport is represented by its name, the distance between airports is in miles and the flight routes only include the traffic or the number of passengers carried.

a) Make the declaration of the class AirportNet

Add the following methods to the class AirportNet:

b) Return the numeric key of a given airport

c) Return the airport with a specific numeric key

d) Return the traffic between two directly linked airports

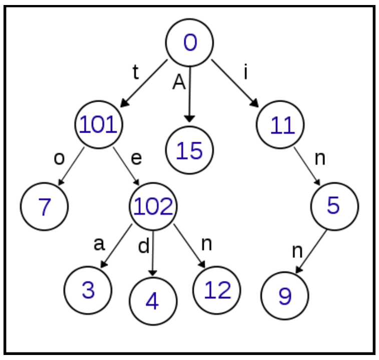
e) Return the miles between two directly linked airports

f) Return the number of routes origin and destination for all airports

g) Return the airports with the greatest connection (more miles)

h) Check whether two airports are reachable

2. A trie is a data structure widely used in Information Retrieval for character string recognition. Generally a trie can be seen as an n-ary tree with symbols in the branches and numerical values in the nodes, where in this case the **terminal nodes** have numerical values **smaller than 100**, the **non terminal nodes** have numerical values **greater than 100** and the initial node has the value zero.



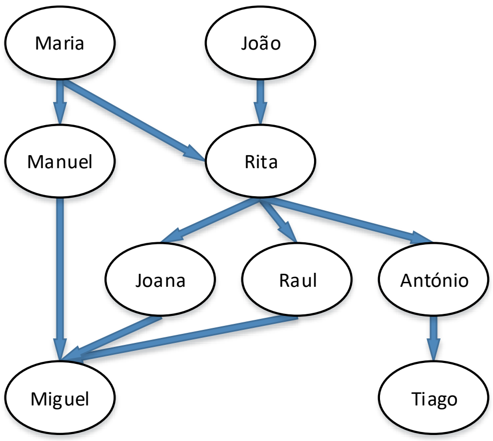
https://pt.wikipedia.org/wiki/Trie

To recognize a sequence of characters, the trie begins at the start node and uses each of the sequence characters to traverse through the branch with that character, if the branch exists. If, at the end of the sequence is at a terminal node the sequence is recognized. The above trie recognizes the sequences: {to, tea, ted, ten, A, i, in, inn}, but the sequences {t, te, x, tent, Al} are not recognized.

Write a method that validates whether a sequence of characters is recognized by the trie. It must return the number corresponding to the terminal node if the sequence is recognized or -1 if it is not.

```
public Integer checkSequence(Graph<Integer,Character> trie, Character[] sequence)
```

3. Consider an oriented and acyclic graph where each vertex represents an employee and each branch x -> y represents semantically that the employee y can only be promoted if the employee x is also promoted. For example, in the graph of the figure if we want to find 2 promotions, there are 2 possible lists: {Maria, João} or {Maria, Manuel}. In case you want 3 promotions, the options are {Maria, João, Manuel} or {Maria, João, Rita}.



Implement a method that, given a graph and a n number of promotions, return a list with the employees to promote. The goal is only to find one of the solutions. The method to be developed must obey the interface:

```
List<String> getPromotions(Graph<String,Integer> g, Integer n)
```

Complementary Exercise

1. The centrality measures are used to calculate the importance of the vertices in a graph. In a social network for example they are used to calculate the degree of influence of their members. Given a graph G and a vertex V of G the **centrality of vertex V** is given by the length of the longest path among all the shorter paths that start from the vertex V. The **radius of a graph G** is defined by the lowest value of centrality of the vertices of the graph. The vertices that have this lower centrality are the centers of the graph. Write an efficient method that determines the centers of a graph and returns its radius

```
Double centersGraph (Graph<V, Double> g, ArrayList<V> lstcenters)
```