Algorithmics

Laboratory Exercise – Breadth-first Search

This is a 'warm-up' for the forthcoming assessed exercise, which provides some practice with graph algorithms and should be used as the basis for the assessed exercise.

The exercise itself involves implementing a program to carry out a breadth-first search in an undirected graph, represented using *adjacency lists*.

Requirements: the setup files are available under Moodle. You will obtain:

- the Java classes `AdjListNode`, `Vertex` and `Graph` discussed in lectures (the latter also includes a pseudocode version of a method implementing breadth-first search);
- a Java class `Main` containing a skeleton main method that includes some file handling statements;
- a test input file (`input.txt`).

The objective is to implement a program with the following specification:

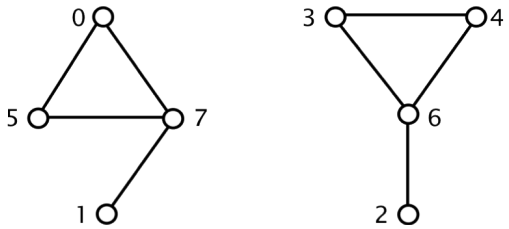
Input: a text file where

- line 1 contains the number (n) of vertices in the graph;
- line j for $j = 2, \ldots, n+1$ contains row j of the adjacency matrix (space-separated 0's and 1's).

Output: a text file containing, for each i ($0 \leq i \leq n-1$), the parent of vertex i in the breadth-first spanning forest found (if vertex i is the root of a tree in this forest, then the parent of vertex i should be vertex i itself).

The names of the input and output files are to be provided as program parameters.

Example: consider the following undirected graph



The input and output generated by you program are of the following form:

Input:

```
8
0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 1 0 1 0
0 0 0 1 0 0 1 0
1 0 0 0 0 0 0 1
0 0 1 1 1 0 0 0
1 1 0 0 0 1 0 0
```

Output:

```
0 7 2 6 6 0 2 0
```

Note: other valid outputs are possible, depending on which vertex in each component is used as the starting point of the breadth-first search. (The output above corresponds to starting with vertices 0 and 2 in each component, since in the output the parent vertices of these vertices are themselves.)