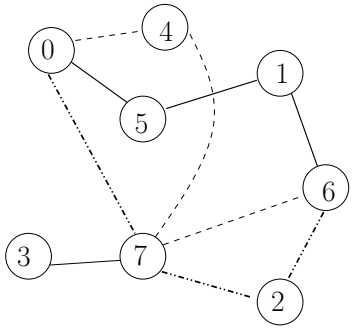
Computer Science 320SC – (2017)

Assignment 6

Due: Friday October 27th (9pm)

Conjested Networks

Given a connected computer network (bidirectional communication) we want to find two different nodes u and v such that we can maximize the congestion between u and v with a continuously sent virus being sent between the pair. We define the *congestion level* as the maximum number of edge-disjoint paths between nodes u and v. For example, the network shown in the following figure has three different paths between nodes 0 and 6 such that each edge is only part of one of the paths. Note that two paths are allowed to go through the same node, such as node 7. No other pair of nodes has a higher congestion level.



Input Specification

We will be given a sequence of connected computer networks each with n nodes, $n \leq 40$, labeled $\{0, 1, \ldots, n-1\}$. The last input case will be followed by a network of $n = 0$ nodes, which should not be processed. The specification for a computer network will be as follows: the first line contains a single non-negative value n, denoting the number of nodes. This is then followed by n lines of integers, separated by spaces, denoting the neighbors (zero indexed) of each node. Expect up to 2000 test cases.

Output Specification

For each input case output one integer on a line by itself denoting the maximum congestion level possible for some pair of its network nodes.

Sample Input

```
8
4 5 7
5 6
6 7
7
0 7
0 1
1 2 7
0 2 3 4 6
4
1 2
0 2
3 0 1
2
0
```

Output for Sample Input

```
3
2
```

We plan to set up two test cases for automated marking. For this assignment name your source code `edgeNetworkE.ext` or `edgeNetworkH.exe`, where `ext` denotes one of { `java`, `cpp`, `py` } that indicates java/c++/python language. Please use just one source file per problem. Here the suffix `E` of the basename denotes 'E'asy (test data) and `H` denotes 'H' arder (test data). Two marks are allocated for `edgeNetworkE.ext` and one mark is for `edgeNetworkH.exe`; or, three marks obtainable for this part of the assignment.

Conjested Networks (part 2)

We extend the problem of the previous section to the case were we want to count only *vertex-disjoint* paths as a measure of conjestion. For this case, the two paths through node 7 of the previous example are not allowed. However, there does exist another pair of nodes (e.g. 6 and 7) that do have three vertex-disjoint paths between them. The input and output specifications are the same as before. For this problem submit a program named `vertexNetworkH.ext`. This problem is worth two of the five total marks allocated for this assignment.