PROGRAMMING

Question1 - Converging Maze:

You are given a maze with N cells. Each cell may have multiple entry points but not more than one exit (ie. entry/exit points are unidirectional doors like valves). The cells are named with an integer value from 0 to N-1.

You need to find the following (all mandatory):

1. The length of the largest cycle in the maze. Return -1 if there are no cycles.
2. Maximum number of entry points (incoming edges) for any cell in the maze
3. Nearest meeting cell: Given any two cells - C1,C2, find the closest cell Cm that can be reached from both C1 and C2

-- Aim for O(N) solution for #1 and #2.
-- Aim for O(Log (N)) solution for #3, assuming there is an equal spread of incoming edges across cell.

INPUT FORMAT

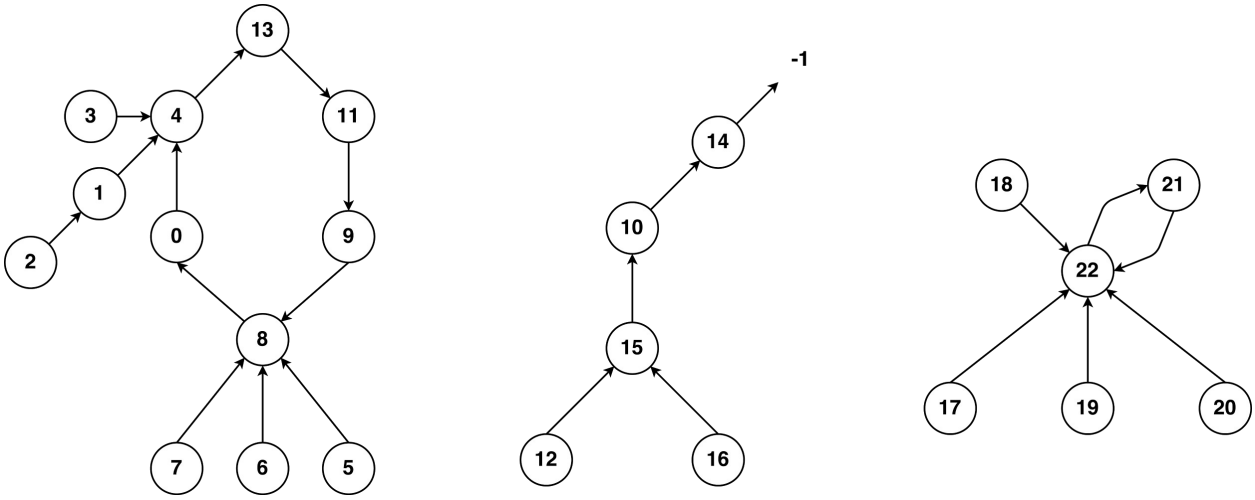
- First line has the number of cells N
- Second line has list of N values of the edge[] array. edge[i] contains the cell number that can be reached from of cell 'i' in one step. edge[i] is *-1* if the 'i'th cell doesn't have an exit.
- Third line contains two cell numbers whose nearest meeting cell needs to be found. (return -1 if there is no meeting cell from the two given cells)

OUTPUT FORMAT

- First line = length of the largest cycle.
- Second line = max entry points in any cell.
- Third line = nearest meeting cell (NMC)

EXAMPLE:
Here is a maze with two loops and three disconnected parts.

INPUT
```
23
4 4 1 4 13 8 8 8 0 8 14 9 15 11 -1 10 15 22 22 22 22 22 21
9 2
```
OUTPUT
```
6
5
4
```



(note: NMC of 2,13 is 13, NMC of 13,7 is 8 or 13)

CODE SUGGESTIONS:

```
// variable names
int N; -- Total number of nodes
int[] edges; -- Link between an exit of cell to entry of next cell

// method signatures
int largestCycle(int[] edges, int N) { }
int maxIncomingEdges(int[] edges, int N) {}
int nearestMeetingCell(int[] edges, int N, int cell1, int cell2) {}
```