Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

8th December 2016

Algorithmen & Datenstrukturen Exercise Sheet P12 AS 16

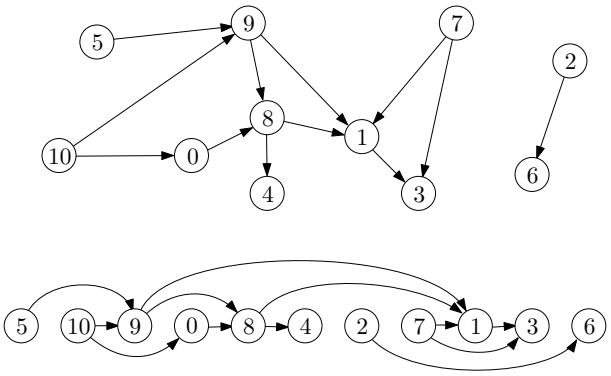
Hand-in: Before Thursday, 15th December 2016 10:00 via the online judge (source code only).

Exercise P12.1 *Longest path in a directed acyclic graph.*

Given a directed acyclic graph (also called a *DAG*), your task is to compute the longest directed path in the graph. The n vertices of the graph are numbers $0, 1, \ldots, n-1$ and there are m directed edges (also called *arcs*), each from vertex s_i to vertex t_i for $i = 1, \ldots, m$. A *directed path* of length l is a sequence of l distinct[1] vertices $p_1, \ldots, p_l$ such that there is a directed edge from p_i to p_{i+1} for every $i = 1, \ldots, l-1$. The graph being *acyclic* means that there is no directed cycle in the graph, or formally there is no directed path from p_1 to p_l such that there is also an edge from p_l to p_1 (closing the directed cycle).

For finding the longest path in a graph, no efficient algorithms are known that would work for *every* graph[2]. But in the case of DAGs, simple and efficient solutions exist and we suggest the following:

First, find a *topological ordering* on the vertices (an ordering $v_1, \ldots, v_n$ of the vertices such that edges from v_i only go to v_j with $j > i$) as seen in the lecture. Then use a dynamic programming or a similar approach on that ordering to find the longest directed path (we leave the details to you).



Example Consider a graph with $n = 11$, $m = 12$ and the following edges (from-to): 7→3, 1→3, 7→1, 2→6, 5→9, 10→9, 10→0, 0→8, 9→8, 9→1, 8→4, 8→1. The graph and one of its

[1] In directed acyclic graph, every directed path is a simple path, as repeating a vertex would imply a directed cycle.

[2] Look up "Hamiltonian path problem" and "NP-completeness" on Wikipedia if you want to know more.

topological orderings are drawn above. The longest directed path has length 5 and there are 3 longest paths: 5-9-8-1-3, 10-9-8-1-3 and 10-0-8-1-3 (there may be many more, and we only care about the length).

Input The input consists of several cases. The first line of the file contains the number of cases to follow.

Each case consists of two lines: The first line contains the integers $1 \leq n \leq 10\,000$ and $0 \leq m \leq 10\,000$, separated by a space. The second line contains $2m$ integers $s_1, t_1, s_2, t_2, \ldots, s_m, t_m$, the start and end points of the directed edges, all $s_i, t_i \in \{1, \ldots, m\}$, separated by spaces.

The graphs contain no loops (e.g. edges from v to the same vertex v) or multiple parallel edges from u to v. Also note that if there is an edge from u to v, there may be no edge from v to u, as that would break acyclicity. The input graph may or may not be connected, as you can see in an example below. Also note that there may be *undirected* cycles (cycles when you ignore the direction of the edges), also illustrated in the example. The edges may be listed in any order.

Output For every test case, write the length of a longest directed path on a separate line.

Example

Input (for the example above and an empty graph)

```
2
11 12
7 3 1 3 7 1 2 6 5 9 10 9 10 0 0 8 9 8 9 1 8 4 8 1
5 0
```

Output

```
5
1
```

Grading You will get 1 bonus point for every 100 judge points, rounded down, with maximum of 200 judge points. The program should run in time $\mathcal{O}(m + n)$ to get full points. Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P12.1`, enroll password is "`quicksort`".

Notes For this exercise, we provide a program template as an Eclipse project archive on the lecture website, which will load the input for you. After loading, the graph is represented in three ways for your use: A list of start- and end-vertices s_i and t_i for the m edges. For every vertex v a list of out-neighbors (vertices with direct edge *from* v). For every vertex v a list of in-neighbors (vertices with a direct edge *to* v). Note that adjacency matrix is not well-suited for this task.

The archive also contains more test data for your local testing. You can use the provided `Judge.java` program to run your `Main.java` on your computer on all the provided tests – just open and run `Judge.java` in the project as you would run `Main.java`. This does not change the way that your `Main.java` works and is just an extra tool. Also, the local test data are of course different and generally smaller than the data that are used in the online judge.

Extra If you can solve the problem above and find the length l of a longest path, how would you count all the directed longest paths in time $\mathcal{O}(m + n)$? (You may assume the final number fits into an `int`.) This is an extra question for no points, but see if you can solve it.