Project 3: Dynamic programming

COT 4400, Fall 2015

Due Dec. 2, 2015

1 Overview

This project requires you to first theoretically solve a dynamic programming problem[1] and write a program that implements your solution.

You are only allowed to consult the class slides, the textbook, the TAs, and the professor. **In particular, you are not allowed to use the Internet.** This is a group project. The only people you can work with on this project are your group members. This policy is strictly enforced.

Specifically, you will solve the *longest ordered path* problem. You will need to find the longest path in a given *ordered graph*, where an ordered graph is a directed graph in which the following are true:

1. The vertices of the graph are numbered 1 to n.
2. The edges of the graph are numbered 1 to m.
3. Every edge goes from a vertex with a lower index to a node with a higher index. That is, every directed edge has the form (i, j) with $i < j$.

Note, your solution to this problem *must use dynamic programming*. Solutions based on graph traversal will not receive credit.

2 Modeling the problem

Before you can write a program to solve this problem, you must first write a report describing how you will solve this problem. This report should address the following:

1. Describe in English how you can break down the larger problem into one or more smaller problem(s). This description should include how the solution to the larger problem is constructed from the subproblems.
2. What recurrence can you use to model the problem using dynamic programming?
3. Prove that your recurrence is correct.
4. Describe a pseudocode algorithm that uses memoization to solve the problem.

[1]Adapted from Algorithm Design, by Kleinberg and Tardos.

5. Analyze the complexity of your memoized algorithm.
6. Describe a pseudocode algorithm that solves the problem iteratively (using dynamic programming). Your algorithm should be optimal in terms of its time and space complexity.

For full credit, your pseudocode must be clear enough that any competent programmer will understand how youf algorithm works and could implement your algorithm in their preferred programming language.

3 Coding your solutions

In addition to the report, you should implement the *iterative* version of your algorithm so that it can solve the longest ordered path problem. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines.

Your code may be split into any number of files. In addition, you are allowed to make use of any built-in library, and C++ users may use the Boost library in their implementations. You may read more about Boost at `www.boost.org`.

3.1 Input format

Your program should read its input from the file `input.txt`, in the following format. This file may contains multiple instances. It begins with a single positive integer on a line by itself indicating the number of problem instances in the file. This line is followed by a blank line, and there is also a blank line between consecutive instances.

The first line of each instance contains two integers n and m, representing the number of vertices and edges for this instance, respectively. The next m line contains each contain two integers separated by a space, and each line represents an edge in the ordered graph. The first integer in this line represents source vertex and the second integer represents the target vertex. The edges are numbered $1, 2, \ldots, m$, according to the order they appear in the input file.

3.2 Output format

Your program should write its output to the file `output.txt`, in the following format. The output file should contain two lines for each input, with a blank line between consecutive outputs. The first output line should indicate the length of the longest path, while the second output line should indicate the edges traversed along the longest path, in the order they are traversed. This line should have k integers, where k is the length of the longest path, with one space between each integer. The i^{th} integer in this line represents the label of the i^{th} edge that is traversed along the longest path. If the input graph has more than one longest path, then any one will do.

3.3 Example

The ordered graph in Figure 1 has a longest path of length 3: (v_1,v_2), (v_2,v_4), (v_4,v_5).

An input file containing this instance appears below. Comments are included so there is no ambiguity, but they will not appear in the actual input file.

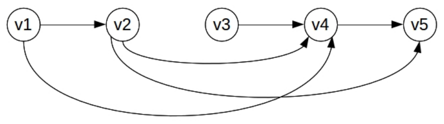

Figure 1: An Ordered Graph

```
1

5 6 // 5 vertices, 6 edges
1 2 // edge #1, from vertex 1 to vertex 2
1 4 // edge #2, from vertex 1 to vertex 4
2 4 // edge #3, from vertex 2 to vertex 4
2 5 // edge #4, from vertex 2 to vertex 5
3 4 // edge #5, from vertex 3 to vertex 4
4 5 // edge #6, from vertex 4 to vertex 5
```

The expected output for this instance would be:

```
3
1 3 6
```

An example input/output file will be posted on Canvas.

4 Submission

Your submission for this project will be in two parts.

The first part of your submission is a zip archive containing 1) your report (described in Section 2) as a PDF document, 2) your code (described in Section 3), and 3) a README file describing how to compile and run your code to Canvas. If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. A simple command might be something like:

```
g++ *.cpp -o longpath
```

If you are using Boost in your solution, you must provide a Makefile and/or shell script that uses the environment variable `$BOOST_HOME` (pointing to the Boost installation directory) to compile your code.

Be aware that your project report and code will be checked for plagiarism.

The second part of your submission is a text file that includes 1) the names of all of your teammates (including yourself), 2) the team member responsibilities, 3) whether or not your teammates were cooperative, 4) a numeric rating indicating the proportional amount of effort each of you put into the project, and 5) other issues we should be aware of when evaluating your and your teammates' relative contribution. The numeric ratings must be integers that sum to 30.

5 Grading

Report	**40 points**
Dynamic programming model	10
Proof of correctness	10
Memoized pseudocode	5
Memoized complexity	10
Iterative pseudocode	5
Code	**30 points**
README file	5
Compiles and is correct	20
Good coding style	5
Teamwork	**30 points**
Follows the given format	5
Participation	25

Note, if your algorithm is inefficient in terms of space or time, you may lose points both for your pseudocode and your implementation.