Software Engineering 284 (2021)

Assignment 4

Due date October 21, 2021, 10pm

There are **three** problems listed below. To get full credit for this assignment you need to complete all of them! The due date is firm in that there are no penalty options available for late submissions (except zero marks).

The first two problems do not need coding, but you must show your working to get full marks. The answers to these should be submitted via Canvas in a single PDF file. A scanned handwritten submission is acceptable if and only if it is clearly written. Hard to read work may not be marked.

The last problem requires you to submit programs that you have written yourself to the automarker, `https://www.automarker.cs.auckland.ac.nz/`. The automaker is already open for submissions.

1. **A graph-theoretic problem.** (10 marks) The mathematics department plans to schedule the classes Graph Theory (GT), Statistics (S), Linear Algebra (LA), Advanced Calculus (AC), Geometry (G) and Modern Algebra (MA) in the following semester. Ten students (see below) have indicated the courses that they plan to take. What is the **minimum** number of time periods per week that are needed to offer these courses so that every two classes having a student in common are taught at different times during a day. Of course, two classes having no student in common can be taught at the same time. For simplicity, you may assume that each course consists of a single 50 min lecture per week.

Peter: LA, S	Robert: MA, LA, G	Andrew: MA, G, LA
Paul: AC, LA, S	Julia: G, AC	Deborah: GT, MA, LA
Bob: AC, S, LA	Anna: GT, S	Angela: G, LA, AC
Linda: LA, GT, S		

   To get full marks, your answer to this question should be clear and detailed. In particular, you are asked to explain which graph-theoretic concept can be used to model the above situation, apply this concept to the situation, and explain how the resulting graph can be exploited to answer the question.

2. **Weighted digraph algorithms**. (10 marks) Suppose that a weighted directed graph $(G = (V, E), c)$ has a negative-weight cycle. In your own words and using complete sentences, describe a polynomial-time algorithm that will list the vertices of one such cycle. (Pseudo-code is not necessary and will not be marked.)

   Explain why your algorithm works correctly. In other words, will your algorithm
   i) provide the correct solution (will not find one, i.e. list no vertices) when the input graph has no negative-weight cycle, and
   ii) list the vertices of a negative-weight cycle, if the input graph has one.

   Note: We do not except full-length formal proof of correctness, but we want to see clear description of the main idea behind the proof and that the reasoning is sound. In this respect, include details if they are crucial for the explanation.

3. **Graph algorithm implementation.** (30 marks) Write two algorithms to solve the following two algorithmic tasks for DAGs.

   (a) Determine the number of connected components in the underlying graph. (15 marks, 5 marks for each of the three test cases)

   (b) Find the length of a longest (directed) path starting from node 0. (15 marks, 5 marks for each of the three test cases)

   Please provide a separate program to solve each task.

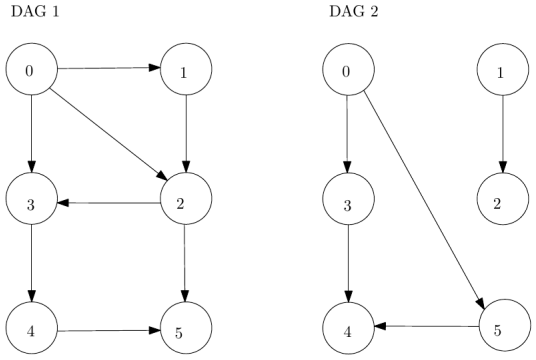
   Input format. The input consists of a sequence of one or more DAGs. Each DAG D is represented by an adjacency list. The first line is an integer n that indicates the order of D. This is followed by n white space separated lists of adjacencies for nodes labeled 0 to $n-1$. The input is terminated by a line consisting of a single zero. This line should not be processed. Each input DAG may contain up to 5000 nodes and up to 10000 arcs. An example input that contains two DAGs is shown next.

```
6
1 2 3
2
3 5
4
5

6
3 5
2

4

4
0
```



   Output format. The output is a sequence of lines, one for each input DAG. Each line contains a single integer indicating the length of the longest path starting from vertex 0.

   Submit your source code to the automated marker `https://www.automarker.cs.auckland.ac.nz`. A set of test cases is provided on Canvas but you are strongly encouraged to write your own (boundary) test cases and test your code before submitting to the automated marker. As in previous assignments, note that the automated marker reads input from stdin. Your program should be efficient, i.e. it must complete with the correct answer for any feasible input in no longer than 4 seconds for the hardest test case.(Note different test cases will have different time limit.)

   - Your implementation must be from first principles and cannot use an existing library methods that might solve the problem (e.g. performs graph operations etc.).
   - The automarker runs on a Linux box. Read the automarker help and FAQ for more details.
   - Supported languages are Java, C,C++, Python, Go, C#, Rust, F#, Javascript.
   - A sample input and output file for each question is available from Canvas.
   - Your maximum number of submissions is fixed at 12 for each subtask (more precisely, for each test case).