

CS 211 Data Structures and Algorithms Lab
Autumn 2020

Assignment no.	7
Objective	To implement Depth First Search (DFS) and to use it to do topological sorting of a directed acyclic graph (DAG)
Total marks	10
Due date (without penalty)	12th November (Thursday) 11:59 pm
Cut-off date (with penalty - 5%)	19th November (Thursday) 11:59 pm
Penalty for violating naming convention(s)	5%

Input

Your program should accept an input file as a command-line argument. A typical execution of your program will be `./a.out sample.graph`. The input file represents a directed acyclic graph (DAG). The first line of the file contains two numbers - first being the number of vertices n and the second being the number of edges m . The vertices are numbered from 0 to $n-1$. Every other line is of the form `x y`, which represents a directed edge from vertex x to vertex y . No edge is repeated in the input file. Your program may create an adjacency list of the input graph which can be used for the task.

Task

Implement DFS and use it to do a topological sorting of the input DAG. The output file should be named as `'ts.txt'`. The output file must contain the vertices - one vertex per line - which represents a topological sorting of the input DAG. Please note that there can be many topological sortings of the same DAG depending on the order in which the DFS is done. You can do DFS in any order that you wish and a valid topological sorting is enough to receive full marks.

Submission

- The program you submit should output `'ts.txt'` when run.
- The main file of your program should be named as `<roll no>.<extension>`, where roll no. specifies your roll no. and the extension depends on the language you choose (Usage of **C/C++/Python 3/Java** is mandatory for this assignment). Ex: `180040001.c`. For java programs, please name the program as `Java_<rollno>.java`
- We will be using gcc/g++ version 6.3, Java version 1.8, Python 3 version 3.6.5 for evaluating your program. If you are using some other version of gcc or java, mostly your program will run fine while doing the evaluation. Please do not use Python 2.

- Test well before submission. You may use the attached sample input file(s) for testing. The corresponding output file(s) is also attached. We have some hidden inputs with us to test your program. The mark you obtain is purely based on whether your program correctly gives outputs for the hidden inputs.
- If your program has only a single source file, please submit the file as it is. If your program has multiple source files, please submit your code as a zip file where the name of the zip file should be your roll number. It is important that you follow the input/output conventions exactly (including the naming scheme) as we may be doing an automated evaluation. There will be a penalty of 5% (on the mark you deserve otherwise) if you do not follow the naming conventions exactly.
- Follow some coding style uniformly. Provide proper comments in your code.
- Submit only through moodle. Submit well in advance. Any hiccups in the moodle/internet at the last minute is never acceptable as an excuse for late submission. Submissions through email or any other means will be ignored.
- Acknowledge the people (other than the instructor and TA) who helped you to solve this assignment. The details of the help you received and the names of the people who helped you (including internet sources, if applicable) should come in the beginning of the main file as a comment. Copying others' programs and allowing others to copy your program are serious offences and a deserving penalty will be imposed if found.

Evaluation

- To consider for first evaluation without penalty, you have to submit your program by the due date. If you submit after the due date but on or before the cut-off date, there will be a penalty of 5% on the marks you deserve otherwise.
- If you do not submit by the cut-off date, your program will not be considered for the first evaluation.
- We will do the first evaluation after the cut-off date. Please note that there may be vertices in the DAG not part of any edges. **There are two parts for the evaluation. For part 1, you either receive 0 marks or 2 marks. To receive 2 marks you have to make sure that the topological sorting produced by your algorithms contains exactly the same set of vertices in the DAG. For the part2 (max: 8 marks), the marks you obtain will be proportional to the number of edges that is being respected by the sequence of vertices that your program gives as a topological sorting. A sequence of vertices respects a directed edge xy if x comes before y in the sequence. We will not be using 'diff' for the evaluation of this assignment.**
- After the first evaluation, you will get a chance to improve your program. For this, after modification, you can submit your code for second evaluation. It comes with a 20% penalty. The due date for the second evaluation will be announced after the first evaluation. Those who submit their code after the cut-off date and before the due date for second evaluation will also be considered for the second evaluation. Submissions done after the due date of the second evaluation will be ignored.