CPSC 4370 Lab 5

Due Date: November 17, 2025

SUBMISSION NOTES: Your submission will include the requested output files to Gradescope.

Prerequisites

This lab assumes you have a working installation of Python.

1 Topological Sorting of Serializable Schedules

We have mentioned transactions repeatedly throughout the semester, and will talk about it in more detail in the coming weeks. Database systems typically allow concurrent execution - that is, multiple transactions can execute in an interleaved manner. Ensuring that the properties of an individual transaction are preserved in this concurrent execution environment requires some extra work. For this lab, please read sections 17.1 to 17.6 of the book. This reading will introduce you to the key concepts and challenges of transaction processing systems, and it will also prepare you for the discussions in class on Tuesday Nov. 11, 2025.

In task 1, we will build a precedence graph from a schedule and determine if the schedule is conflict serializable. When a schedule is conflict serializable, we will find all the topolological orderings of the schedule.

Example schedule

Consider the following schedule:

Transaction	T1	T2	T3	T4	T5
	read(A)	–	read(B)	read(E)	read(F)
	write(A)	read(H)	write(B)	write(E)	write(F)
	–	write(H)	read(A)	read(G)	read(F)
	read(C)	–	read(A)	read(H)	write(F)
	write(C)	read(D)	read(A)	read(H)	–
	–	write(D)	read(E)	write(H)	read(C)
	–	read(D)	write(E)	–	write(C)
	read(G)	write(D)	–	read(F)	read(C)

To clarify notation, consider that for two operations i, j of transactions T_m, T_n, if the row of i in the column of T_m is above the row of j in the column of T_n, then operation i must run before operation j. In case i and j share the same row, there is no limitation on which operation may run first.

For the above schedule, the resulting precedence graph is as follows:

- Source $T1 \rightarrow [T5, T3]$
- Source $T3$ has no outgoing edges

- Source $T4 \to [T3]$
- Source $T5 \to [T4]$
- Source $T2 \to [T4]$

The precedence graph is free of cycles, so the schedule is conflict serializable. The topological orderings of the schedule are:

- [`T1, T5, T2, T4, T3`]
- [`T1, T2, T5, T4, T3`]
- [`T2, T1, T5, T4, T3`]

1.1 Building a Precedence Graph

The file `precedence_graph.py` contains a class `PrecedenceGraph` that implements a directed graph—represented as an adjacency list. Complete the private functions `_has_conflict` and `_build_graph` to build the precedence graph given a schedule by following the TODO items.

1.2 Serializable Schedule Check

The file `precedence_utils.py` contains a function `is_conflict_serializable` that determines if a schedule is conflict serializable. The function takes a precedence graph as input and returns a boolean value. The function `has_cycles` is a helper function that determines if a graph has cycles. You will implement both functions by following the TODO items.

1.3 Topological Orderings

The file `precedence_utils.py` contains a function `find_all_topological_sorts` that finds all topological orderings of a schedule. The function takes a precedence graph as input and returns a list of all topological orderings. You will implement this function by following the TODO items.

Submission

Submit the files `precedence_graph.py` and `precedence_utils.py` as part of your submission to Gradescope.

Submission

Collaboration Policy

Assignments in this course are designed to assess your individual understanding and skills. While you are encouraged to engage in discussions with the instructor, teaching fellows, undergraduate learning assistants, or classmates to deepen your understanding of course-related material, it is imperative that the work you submit is entirely your own creation. This means that any code, solutions, or written responses must be developed and written by you without copying from others.

Collaboration on assignments, unless explicitly permitted, is not allowed. This includes not only direct copying but also closely mimicking the logic or structure of another person's work. If you find yourself benefiting substantially from a discussion or an external resource, it is your responsibility to clearly acknowledge this in your submission, detailing the nature and extent of the assistance received. However, remember that your final work must still be predominantly a product of your own efforts.

As a reminder, plagiarism in any form is a serious violation of academic integrity. This includes the unauthorized use of AI tools like ChatGPT, online forums, code repositories, or any other sources that provide direct solutions. Seeking general advice and debugging help is permissible, but the line between assistance and infringement of academic integrity should not be crossed. If in doubt, consult the instructor or teaching staff for guidance on acceptable use of external resources.

Ultimately, the goal is for you to develop and demonstrate your own skills and knowledge. Adhering to these guidelines ensures a fair and meaningful educational experience for everyone involved.

Assignment Submission

Submit the following files to Gradescope:

1. `precedence_graph.py`
2. `precedence_utils.py`

Grading

The assignment is worth 30 points. The grade will be broken down as:

Code Quality and Organization (8 Points)

Provide clear, well-organized, and well-commented code so graders can follow your logic and methodology.

Code and Output Files (22 Points)

1. `precedence_graph.py` (9 points)
2. `precedence_utils.py` (13 points)