Your name
CS 163/265, Winter 2020
Homework 3
Due Jan 31

List of collaborators:

1. (163 only:) Give an example of a directed acyclic graph, and a starting vertex for a depth-first search of the graph, such that using preorder (the same thing as dfs number) instead of postorder does not work for topological ordering. Your graph should include at least one directed edge $p \to q$ for which p is earlier in the preorder than q (that is, p has a smaller dfs number than q), and at least one other directed edge $r \to s$ for which s is earlier in the preorder than r.

(For your graph, the preorder is not a topological ordering because edge $r \to s$ is directed in the wrong direction, and the reverse preorder is not a topological ordering because edge $p \to q$ is directed in the wrong direction. All of the vertices in your graph should be reachable from the chosen starting vertex.)

(265 only:) Recall that the outer loop of depth-first search goes through the vertices in an arbitrary order and starts a new recursive search whenever it reaches an unvisited vertex. We can modify this algorithm by choosing how to order the vertices in the outer loop instead of letting the order be arbitrary.

(a) Describe how to find, for every directed acyclic graph G, an ordering of the vertices of G with the following property: Each recursive search, started from the outer loop, finds only a single unvisited vertex. Equivalently, each tree in the depth-first search forest contains only a single vertex.

(b) If a depth-first search has the property described in part (a), describe how to obtain a topological ordering of G from the preorder of its depth-first search forest.

■

2. Find a directed acyclic graph G with three vertices and exactly three different topological orderings. List the three topological orderings of your graph.

■

3. Suppose we are trying to schedule a project that has six tasks, A, B, C, D, E, and F. Tasks A and B can start immediately, but task C can only start when task A is completed, task D can only start when both A and B are completed, task E can only start when both B and C are completed, and task F can only start when both C and D are completed.

(a) Draw a directed acyclic graph (an activity-on-edge graph) in which:

- Each vertex represents a milestone for the project (the start or end of at least one task)
- One of the milestones (the start of the project) can reach all other vertices, and another of the milestones (the end of the project) can be reached by all other vertices
- Each directed edge is either labeled (representing a task) or unlabeled (representing an ordering constraint between two milestones)
- Each task label is the label of exactly one edge
- The labeled edges along every path from the start milestone to the end milestone represent a sequence of tasks that must be performed one after the other.

(b) Suppose that the number of days required to complete each task is A:4, B:3, C:1, D:5, E:2 F:3. What is the sequence of tasks on the critical (longest) path of your network? What is the length of the path?

■

4. Suppose that instead of finding a single optimal path in a directed acyclic graph, we wish to count the number of different paths from a given starting vertex s to a given ending vertex t. We will do this by assigning to each vertex a value $P[v]$, counting the number of paths from s to v, and then returning $P[t]$, according to the following algorithm outline:

```
for each vertex v in a topological order of the given graph:
    if v == s:
        P[v] = 1
    else:
        P[v] = (formula for the number of paths from s to v)
return P[t]
```

Write the missing formula for the number of paths from s to v, expressing it in terms of the values of $P[u]$ for other vertices u in the graph. Your formula should only use values $P[u]$ for vertices u that are earlier than v in the topological ordering, so that these values will have already been computed when they are needed. You can assume in your formula that $v \neq s$. Make the formula as simple as possible.

■