Please enter your name and uID below.

Name:

uID:

Collaborators, if any, and how you collaborated:

Submission notes

- Due at 11:59 pm Thursday, Nov 3rd.
- Solutions must be typeset using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) *without* space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for this problem set, you are allowed to collaborate in detail with your peers, as long as you cite them. However, you must write up your own solution, alone, from memory. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Artisanal Cooking, 30pts)

   Describe (and analyze) an *efficient* algorithm to solve the PS6 programming problem.

   Your algorithm description should contain enough detail, written in full sentences, for one of the course instructors to be able to turn it into code. This should include (but isn't necessarily limited to) how you will map your input to a graph, what data structure(s) you will use, what algorithm(s) you will run on your graph, how you will produce the desired output, and why your algorithm works. In describing your algorithm you may directly use algorithms from the book as subroutines without rewriting them, but if you use a variation of an algorithm from the book you should be *very clear* about how your variation differs, or rewrite the algorithm with your modification included.

   You should also give the running time of your entire algorithm, in terms of the input parameters N and M, and how you determined that running time. There are many possible algorithm implementations with different running times. Our only criteria for being *efficient* is that an unoptimized implementation of your algorithm would solve the programming problem without timing out.

   Hint: use a topological sort as part of your algorithm.

2. (Exact Paths, 45pts)

   Let G be a directed acyclic graph with positive integer edge lengths given by $\ell(v{\to}w)$. Each part below considers some start vertex s and some target path length[1] L.

   a. Describe a dynamic programming algorithm to determine whether G has any path starting at s with length exactly L. Your algorithm should return true if there is a length L path starting at s, or false otherwise.
   b. Describe a dynamic programming algorithm to count the *number* of paths in G with length L. You may describe your algorithm relative to part (a), e.g. "everything would be the same as in part (a), but the recurrence would instead be ...".
   c. How could you use your algorithm from part (b) to count the *total* number of length L paths in G starting from *any* vertex?

[1] Recall that the length of a path is the sum of its edge lengths.