COM S 311 SPRING 2020
HOMEWORK 5
Due: May 3 11:59 p.m.
Early submission: May 2, 11:59 p.m., (5% bonus)
Late Submission Due: May 4, 11:59 p.m. (20% penalty)

(1) Consider the interval scheduling problem studied in class. Suppose that instead of always selecting the earliest job to finish, we instead select the last job to start that is compatible with all previously selected jobs. Prove that this algorithm yields an optimal solution. What is the running time of this algorithm? Justify your answer.

(2) Not just any greedy approach to the interval scheduling problem produces a maximum-size set of mutually compatible jobs.
   (a) Give an example to show that the approach of selecting the job of least duration from among those that are compatible with previously selected activities does not work.
   (b) Give an example to show that the approach of always selecting the compatible job that overlaps the fewest other remaining activities does not work
   (c) Give an example to show that the approach of always selecting the compatible remaining job with the earliest start time does not work.

(3) Let $G = (V, E)$ be a directed graph where every edge $e \in E$ has a positive weight $w(e)$ and let $s \in V$ be a specified source node of G. The ***bottleneck shortest path problem*** is to find, for every node $u \in V$, a path P from s to u such that the weight of the minimum-weight edge in P is maximized.
   (a) Give a $O(m + n)$-time algorithm that solves the bottleneck shortest path problem in a DAG. Justify your algorithm and analyze its running time.
   (b) Give a $O(m \log n)$-time algorithm that solves the bottleneck shortest path problem in an arbitrary directed graph. Justify your algorithm and analyze its running time.

(4) Professor Borden proposes a new divide-and-conquer algorithm for computing minimum spanning trees, which goes as follows. Given a graph $G = (V, E)$, partition the set V of vertices into two sets V_1 and V_2 such that $|V_1|$ and $|V_2|$ differ by at most 1. Let E_1 be the set of edges that are incident only on vertices in V_1, and let E_2 be the set of edges that are incident only on vertices in V_2. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in E that crosses the cut (V_1, V_2), and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

   Either argue that the algorithm correctly computes a minimum spanning tree of G, or provide an example for which the algorithm fails.

Guidelines

- For each problem, if you write the statement "I do not know how to solve this problem" (and nothing else), you will receive 20% credit for that problem. If you do write a solution, then your grade could be anywhere between 0% to 100%. To receive this 20% credit, you must explicitly state that you do not know how to solve the problem.
- You are allowed to discuss with classmates, but you must work on the homework problems on your own. You should write the final solutions alone, without consulting anyone. Your writing should demonstrate that you understand the proofs completely.

- When proofs are required, you should make them both clear and rigorous. Do not hand-waive.
- Please submit your assignment via Canvas.
  - If you type your solutions, then please submit a PDF version.
  - If you hand-write your solutions, then please scan or take a picture of your solutions and submit a PDF version. Please make sure that the quality of the scan/picture is good, and that your handwriting is legible.
  - Please make sure that the file you submit is not corrupted and that its size is reasonable (e.g., roughly at most 10-11 MB).

    If we cannot open your file, your homework will not be graded.
- Any concerns about grading should be expressed within one week of returning the homework.

Note. We reserve the right to grade only a subset of the problems assigned. Which problems will be graded will be decided after the submission deadline.