CS5800 Algorithms ***B***

Solutions for Exam on October 25, 2019

Ravi Sundaram

- **Logistics:** Read all these instructions. Write your name on the top of every page. Write your name in uppercase in the blank provided on this page. If you need extra space, use the blank facing page.
- **Duration, Problems and Points:** This exam lasts 90 minutes and has 4 problems totaling 100 points. Make sure to read all problems to decide an order to attempt them that would allow quickest progress.
- **Known algorithms:** While describing an algorithm, you may use any algorithm covered in class as a subroutine without elaboration.
- **Proofs and analyses:** You do not need to provide a proof of correctness and/or an analysis of the running time *unless* explicitly asked to.
- **Partial Credit:** Points are determined on the basis of both the correctness and the clarity of description. Show your work, as partial credit may be given. Present the best algorithm you can, even if it doesn't meet all desired properties.
- **Closed-book exam:** This exam is closed-book. No sources of any kind, electronic or physical, can be consulted during the course of this exam. No collaboration of any kind is permitted.
- **Policy on cheating:** Students who violate the above rules on scholastic honesty are subject to disciplinary penalties. Any student caught cheating will *immediately* receive an **F** (failing grade), and the case will be forwarded to the Office of Student Conduct and Conflict Resolution.

  Sign below and confirm that you are strictly abiding by the rules of this test.

Signature: _____

FULL NAME :
(BLOCK letters)

Question	1	2	3	4	Total
Points	10	40	20	30	100

Best of luck!

Problem 1 (True/False) 10

Indicate whether the following statements are True or False by filling in the table with T or F respectively:

a) If an undirected connected graph on n vertices contains cycles, then it must have at least n edges.

b) If a graph contains negative edge weights but it doesn't contain negative cycles, the Bellman-Ford algorithm always returns the shortest path.

c) Given intervals with weights associated with them, finding the set of disjoint intervals with maximum sum of weights can be done using a greedy algorithm.

d) Given jobs with deadlines and durations, executing the jobs in order of decreasing deadlines minimizes maximum lateness. Recall that the lateness of a job is the difference between the deadline and finish time of a job.

e) Trains arrive to and depart from a station at different times, the minimum number of platforms needed by the station equals the maximum number of trains that halt at the same time.

Solution:

Question	a	b	c	d	e
T/F	T	T	F	F	T

■

Problem 2 (Algorithms on a graph) 20 + 20

Consider the weighted undirected graph $G(V, E)$ shown in 3. For each of the following algorithms, fill out the final output when run on G using the given templates.

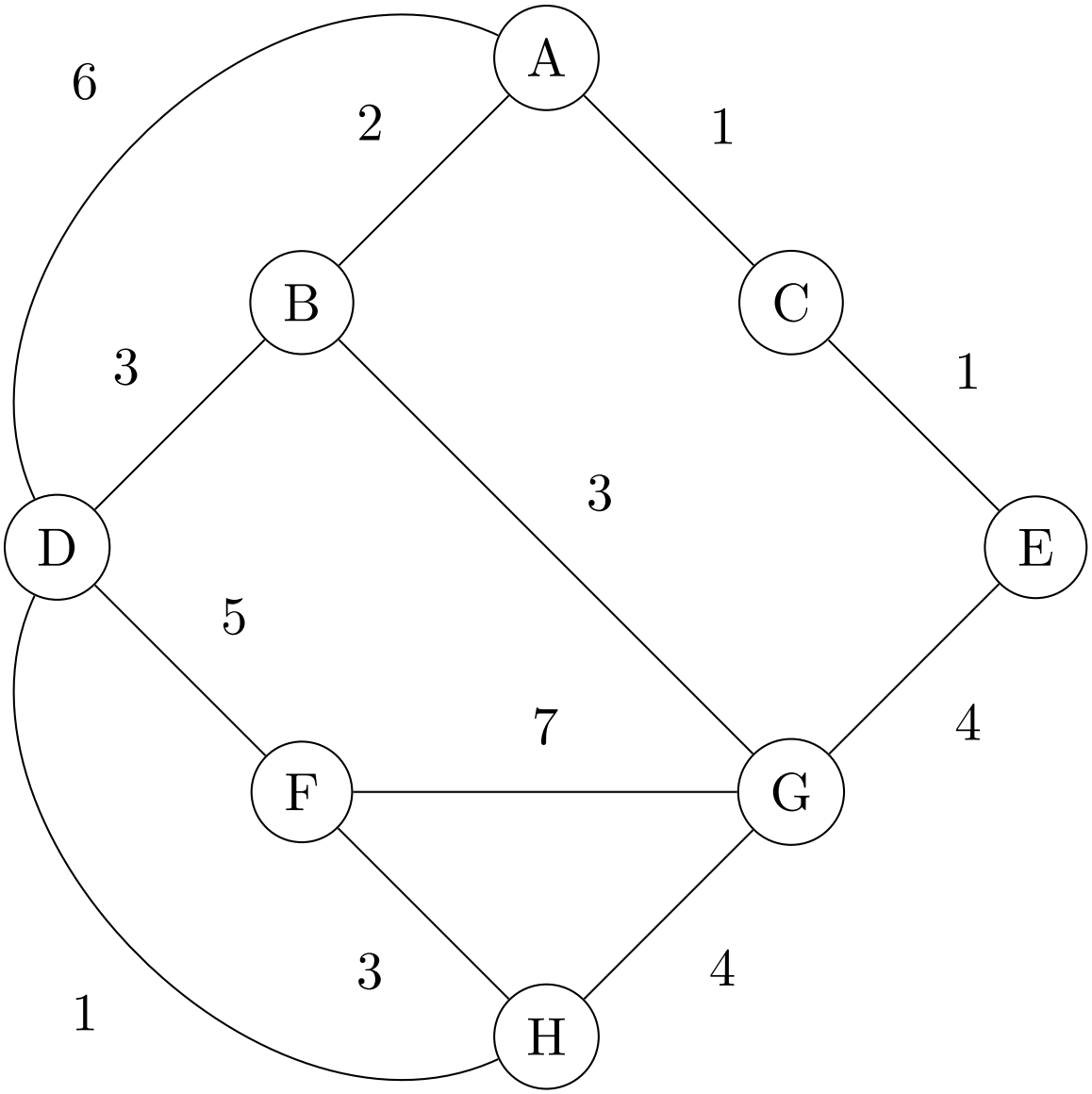

Figure 1: Graph G

a) **Dijkstra's algorithm** starting at node A. Give the shortest path tree: the tree formed by the edges on the shortest paths starting at node A. Note that these edges will be directed away from node A on the shortest paths; make sure to mark these directions.

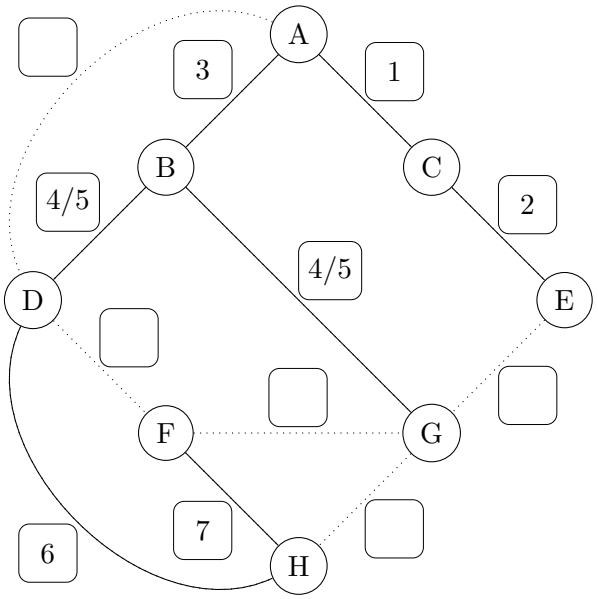

Figure 2: Fill in with the shortest path tree obtained via **Dijkstra's algorithm**.

Note: *For each edge that is **added** to the output answer, use the box adjacent to it to write the order in which it gets added. For example, for the first edge that gets **added** write 1, write 2 for the second edge and so on. Also, fill in the edge along the dotted line if that edge is added. If there are multiple edges that could be added at a step according to the algorithm, you can break such ties arbitrarily.*

b) **Kruskal's algorithm**. Give the minimum spanning tree.

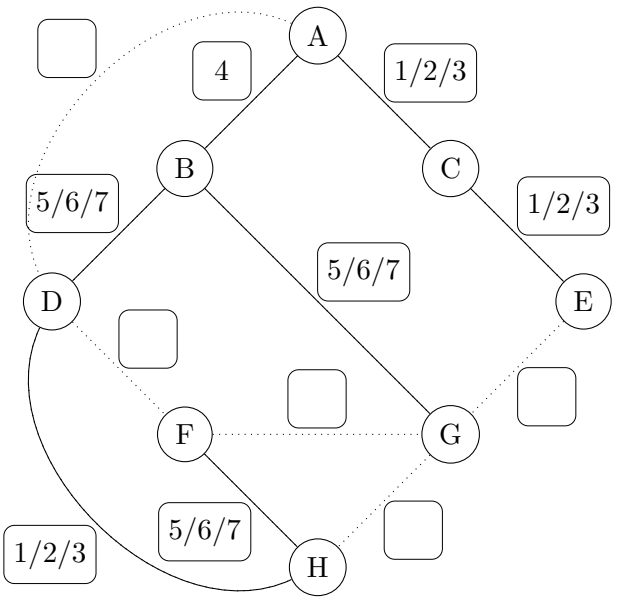

Figure 3: Fill in with the Minimum Spanning Tree obtained via **Kruskal's algorithm**.

Note: *For each edge that is **added** to the output answer, use the box adjacent to it to write the order in which it gets added. For example, for the first edge that gets **added** write 1, write 2 for the second edge and so on. Also, fill in the edge along the dotted line if that edge is added. If there are multiple edges that could be added at a step according to the algorithm, you can break such ties arbitrarily.*

Problem 3 (Bellman Equations) 20

Consider a robot that can jump either of $\{a_1, a_2, \ldots, a_m\}$ steps in a single jump. Given the total number of stairs n that the robot reached by jumping, find number of ways the robot could have jumped to reach that total. As an example if the robot can jump either of $\{1, 2\}$ and $n = 3$ then there are three ways $\{(2, 1), (1, 2), (1, 1, 1)\}$ it can get to a total of n.

Write down the Optimal Bellman Equation you get for this problem. Make sure to write down an explanation for your logic in English. You do **not** need to write the full algorithm or explain any time complexity.

Solution:

To reach n, you must be at $n - a$ previously, where $n \in \{a_1, a_2, \ldots a_m\}$. Hence, let $N(k)$ be the number of ways to reach n, then,

$$N(k) = \begin{cases} 0 \ \textit{if } n < 0 \\ 1 \ \textit{if } n = 0 \\ \textit{otherwise } \Sigma_{i=1}^{n} S(k - a_i) \end{cases}$$

and $N(n)$ gives the answer. ■

Problem 4 (Managing stock traders) 30

The price of a stock on each day is given to you in an array. Assume you have enough money to buy a stock on all of the days. However, you cannot buy if you already have a stock in hand (every buy must be followed by a sell before another buy). To be more precise, on each day you have one of three options. You can either buy exactly one share of the stock, sell exactly one share of the stock, or do nothing. However, you can't buy any additional shares if you already possess one share, and you cannot sell any shares if you don't possess one share. In other words, you can only have exactly 0 or 1 share at any given moment. A transaction is 1 buy followed by 1 sell. You can perform at most K transactions.

As the manager of some traders in your company you want to come up with an algorithm to find out the *maximum loss your traders might possibly incur* by buying/selling the stock on these days.

For example, if the given array is [100, 200, 250, 330, 40, 30, 700, 400], and $K = 2$ the maximum loss will be incurred when buying on day 4, selling on day 6, again buy on day 7 and sell on day 8. As another example, if the given prices in the array keep increasing, then no loss will be incurred.

Note: *If you use a greedy algorithm, write the pseudocode and prove the algorithm is optimal. If you use a dynamic programming algorithm then instead of pseudocode, use the DP template: English description, Bellman Equation (including the base case), iterative approach, number of subproblems, time for each subproblem, final running time, final value to return.*

Solution:

The given problem requires a DP solution, because, consider an array of prices, where k = 1 :
$[4, 1, 5, 2]$
A greedy solution will choose : either Buy at 4 , Sell at 1 for a loss of 3, or Buy at 5, Sell at 2 for a loss of 3. However, we could have just selected Buy at 5 , Sell at 2 for a loss of 3.

Let $loss[t, i]$ be the maximum loss that can be made by making at most t transactions upto the i^{th} day. Then on the i^{th} day, a choice can be made:

- Either don't do any transaction (meaning no buy/sell)
- Make the maximum loss by completing a transaction on the i^{th} day. To complete a transaction, we need to sell the stock on the i^{th} day an buy it on j^{th} day such that $j < i$. We add this loss to the earlier max. loss we had made with 1 less transaction upto j^{th} day.

Hence, our Bellman equation looks like this:

$loss[t][i] = max(loss[t][i-1], max(\forall j \in [1, i-1], price[j]–price[i] + loss[t-1][j]))$

If we create a table $loss[t][i]$, then filling each cell requires time $O(n)$, and as the table is of size kn, total complexity is $O(kn^2)$.

■