CS570 Summer 2018: Analysis of Algorithms Exam III

	Points
Problem 1	20
Problem 2	16
Problem 3	16
Problem 4	16
Problem 5	16
Problem 6	16
Total	100

Instructions:

1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
5. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
6. If using a pencil to write the answers, make sure you apply enough pressure so your answers are readable in the scanned copy of your exam.
7. Do not write your answers in cursive scripts.

1) 20 pts

Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
An undirected graph is said to be Hamiltonian if it has a cycle containing all the vertices. Based on this definition, is the following true or false?
Any DFS tree on a Hamiltonian graph must have depth V-1.

[TRUE/FALSE]
At the termination of the Bellman-Ford algorithm, even if the graph has a negative length cycle, a correct shortest path is found for a vertex for which shortest path is well-defined.

[TRUE/FALSE]
The main difference between divide and conquer and dynamic programming is that divide and conquer solves problems in a top-down manner whereas dynamic-programming does this bottom-up.

[TRUE/FALSE]
If problem A is polynomial-time reducible to a problem B (i.e., $A \leq_p B$), and B is NP-complete, then A must be NP-complete.

[TRUE/FALSE]
If problem A is polynomial-time reducible to a problem B (i.e., $A \leq_p B$), and A is NP-complete, then B must be NP-complete.

[TRUE/FALSE]
Flow f is maximum flow if and only if there are no augmenting paths.

[TRUE/FALSE]
If P ≠ NP, then the general optimization problem TRAVELING-SALESPERSON has a poly-time approximation algorithm with approximation factor 1.5.

[TRUE/FALSE]
We currently only have a weakly polynomial time solution for the 0/1 knapsack problem, but not a strongly polynomial time solution.

[TRUE/FALSE]
Suppose that all edge capacities c(e) is a multiple of α. Then the value of the maximum flow is also a multiple of α

[TRUE/FALSE]
Suppose that all edge capacities c(e) is a multiple of α. In a maximum flow f, the flow f(e) on edge e is always a multiple of α

2) 16 pts.

Recall that in exam 1 (just a couple of weeks ago), you were asked to find a polynomial time solution to this problem:
Input: Undirected, connected graph $G = (V, E)$ with edge weights w_e. We are also given U which is a subset of vertices of the graph ($U \subset V$).
Output: Find the spanning tree of minimum total weight such that all nodes of U have degree 1.

Now prove that the following problem (MST-D2) is NP-Hard:
Input: Undirected, connected graph $G = (V, E)$ with edge weights w_e. We are also given U which is a subset of vertices of the graph ($U \subset V$).
Output: Find the spanning tree of minimum total weight such that all nodes of U have **degree 2** (if such a tree exists).

Solution:
Show that Hamiltonian Path $\leq_p$ MST-D2

Given G an undirected graph – an instance of the Hamiltonian Path problem, we will construct G' an undirected graph with unit weights on all edges. Then for every pair of nodes in G (say v and u), we will call the black box that solves MST-D2 and set U=V-{v,u}. There will be O(n2) calls to the black box. If any of these calls returns a tree such that all nodes in the tree have degree 2 except for v and u, it means that we have found a Hamiltonian Path in G with v and u as its end points. If none of the calls to the black box manage to return such a tree, then we can conclude that G has no Hamiltonian Paths.

Rubric:

8 pts if the transformation is correct and sound.
8 pts for the proof

3) 16 pts.
You are traveling by renting cars from car rental locations of cities. There are n cities along the way. Also, a company named Trojan Rental dominates the car rental market, and therefore there is only one car rental location in each city. Before starting your journey, you are given for each $1 \le i < j \le n$, the fee $f_{i,j}$ for renting a car from city i to city j, which is arbitrary. For example, it is possible that $f_{2,3} = 8$ and $f_{2,5} = 6$. Also, city j charges t_j% in taxes on the fee ($f_{i,j}$) collected in city j when the car is dropped off.

You begin at city 1 and must end at city n. Your goal is to minimize the rental cost. Give the most efficient algorithm you can for this problem. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

a) Define (in plain English) subproblems to be solved (3 pts)

The sub problem is to calculate the minimum rental cost with taxes from a city i to n where 1<i<=n

b) Write the recurrence relation for subproblems. (5 pts)
Solution:

Let $m[i]$ be the rental cost for the best solution to go from post i to post n for $1 \le i \le n$. The final answer is in $m[1]$. We can recursively, define $m[i]$ as follows:

$$m[i] = \begin{cases} 0 \quad if\ i = n \\ \min\limits_{i<j\le n} f_{i,j} + tax + m[j] \quad otherwise \end{cases}$$

Points given for correct recurrence relations :
base condition: 1
rest: 4

c) Write pseudo-code to solve the above problem using the recurrence relation in part b. (6 pts)

Initialization and returning correct value: 2+1
Pseudocode implemented correctly from correct recurrence relation: 3

d) Compute the runtime of the algorithm in terms of n. (2 pts)
Solution:

For the time complexity there are n subproblems to be solved each of which takes $O(n)$ time. These subproblems can be computed in the order $m[n], m[n-1], \ldots, m[1]$. Hence the overall time complexity is $O(n^2)$.

2 points for time complexity of n^2 .

Since the question asked for the most efficient solution: 2D array implementation which is not optimal has been given partial credits even if they yield a correct result.

4) 16 pts

Suppose you have 4 production plants for making cars. Each works a little differently in terms of labor needed, materials, and pollution produced per car:

	Labor	Materials	Pollution
Plant 1	2	3	15
Plant 2	3	4	10
Plant 3	4	5	9
Plant 4	5	6	7

Suppose we need to produce at least 400 cars at plant 3 according to a labor agreement. We have 3300 hours of labor and 4000 units of material available. We are allowed to produce 12000 units of pollution, and we want to maximize the number of cars produced.

Formulate a linear programming problem, using minimal number of variables, to solve the above task of maximizing the number of cars. You need to provide

a) Clear definition of all variables in the LP formulation (3 pts)

b) Objective function and constraints (13 pts)

Solution:

We need four variables, to formulate the LP problem: x_1, x_2, x_3, x_4, where x_i denotes the number of cars at plant-i.

Maximize $x_1 + x_2 + x_3 + x_4$

s.t.

$x_i \geq 0$ for all i

$x_3 \geq 400$

$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 3300$

$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 4000$

$15x_1 + 10x_2 + 9x_3 + 7x_4 \leq 12000$

3 points for mentioning all the four variables

2 points per constraint specified correctly.

3 points for writing obj function correctly

5) 16 pts

Suppose that we are given a graph G = (V, E) together with one if its minimum spanning trees T. Now we add a new vertex v together with k edges that are incident on v in graph G. Given an algorithm that runs in O(k|V |) time to find the new minimum spanning tree.

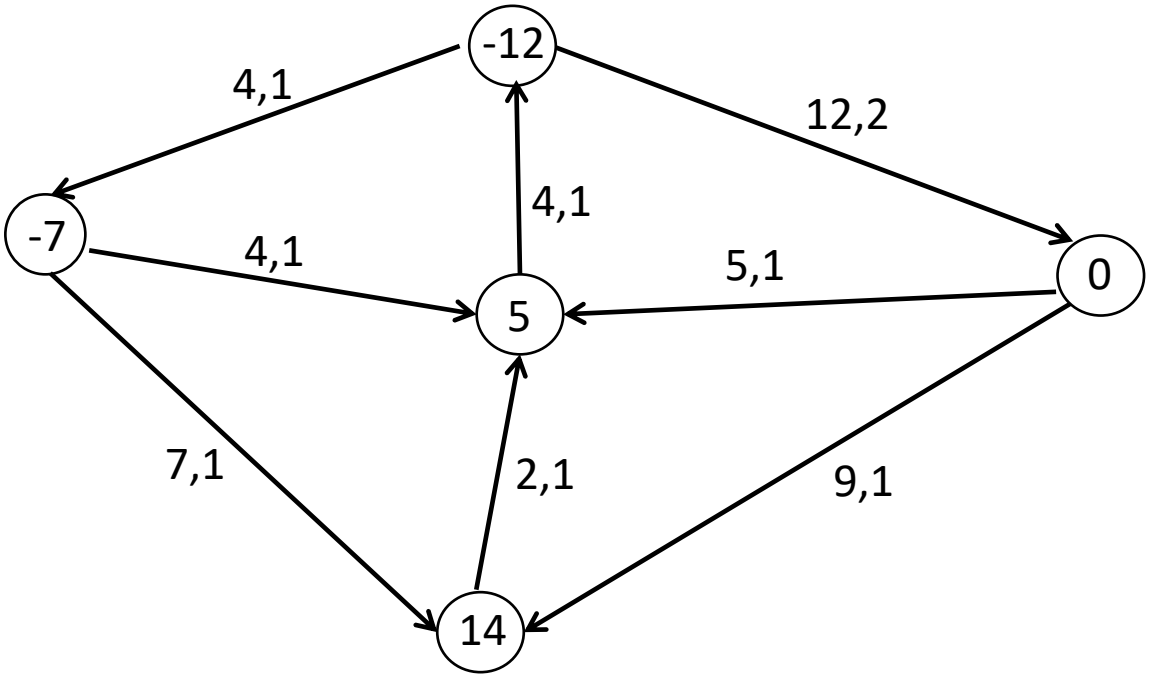
Solution:
Let e be the lightest edge incident to v, where v is the new vertex, then e must be included in the new MST (unless there are multiple lightest edges, but this argument does not lose generality). Therefore, we first add e to T to get $T' = T \cup \{e\}$. Then, for every other new edge $e' \neq e$, we first add the edge e' to the tree T' , and then remove the maximum edge from the resulting cycle in $T' \cup \{e'\}$.
For proof of correctness, show that following lemma is true:
Lemma 1. Let e be a maximum-weight edge on some cycle of a connected graph $G = (V, E)$, then there is a minimum spanning tree of G that does not contain e.

12: A working algorithm maintaining the complexity
4: Correctness proof

6) 16 pts

In the circulation network (G) below, the demand and supply values at each node are shown in circles. The (capacity, lower bound)'s are also marked up on all edges.

a) Reduce this circulation-with-lower-bounds problem to max flow.

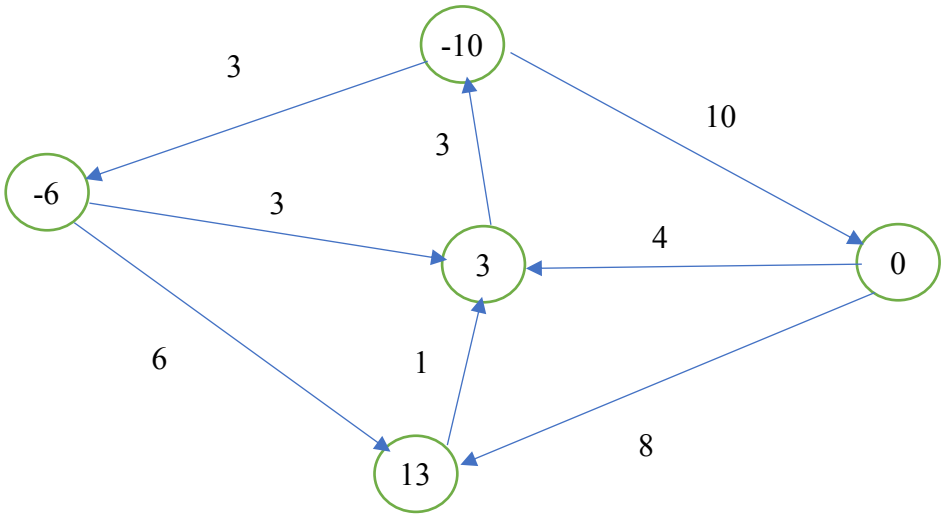


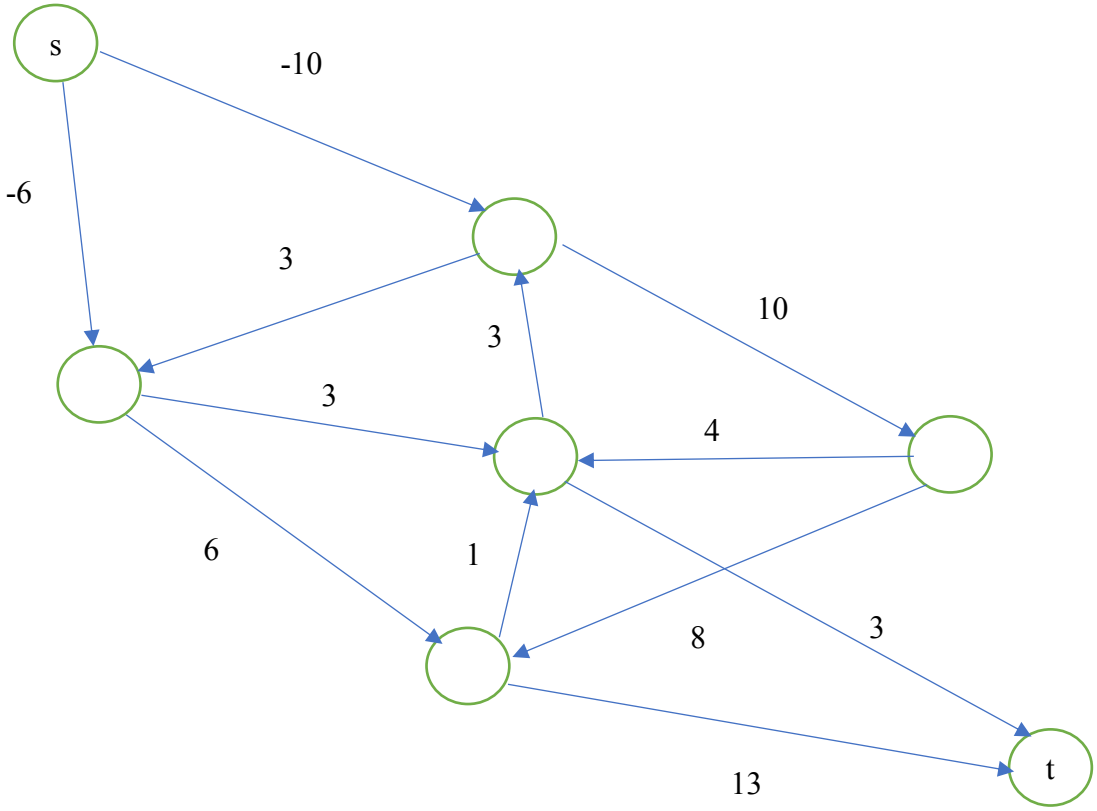
Solution:

$$dv' = dv + \sum_{e,out} l_e - \sum_{e,in} l_e$$

$$c'_e = c_e - l_e$$

12 pts





b) What should the value of max flow in your resulting flow network be in order to have a feasible circulation in G?

Solution: 16
4 pts

Additional Space

Additional Space