CS570 Fall 2018:

Analysis of Algorithms

Exam II

Grading Rubric

1) 15 pts. (**Zimo Li + Amrita Dasgupta**)

There is a series of $n > 0$ jobs lined up one after the other. The i-th job has a duration $t_i \in \mathbb{N}$ units of time, and you earn $p_i \geq 0$ amount of money for doing it. Also, you are given the number $s_i \in \mathbb{N}$ of immediately following jobs that you cannot take if you perform that i-th job. Design a dynamic programming algorithm to maximize the amount of money one can make in T units of time.

a) Define (in plain English) subproblems to be solved. (4 pts)

Let OPT(i, t) be the max profit for jobs from {1,2,…, i} and total time t.

A word description also suffices.

b) Write the recurrence relation for subproblems. (7 pts)

OPT(i, t) = max (OPT(i + 1, t), p_i + OPT(i + 1 + s_i, t – t_i))

-1 point for putting only (i + s_i) instead of (i+1+s_i)

Base case: OPT (i , t) = 0 for $i > n$
OPT (i , t) = 0 for $t < 1$

NOTE: max(OPT(i -1, t), p_1 + OPT(i - 1 - s_t, t- t_j)) is also acceptable. Essentially, reversing it so it means the number of jobs before, as the question could be read this way mistakenly, and it amounts to the same situation.

c) State the runtime of the algorithm. Is it polynomial? Explain your answer.(4 pts)

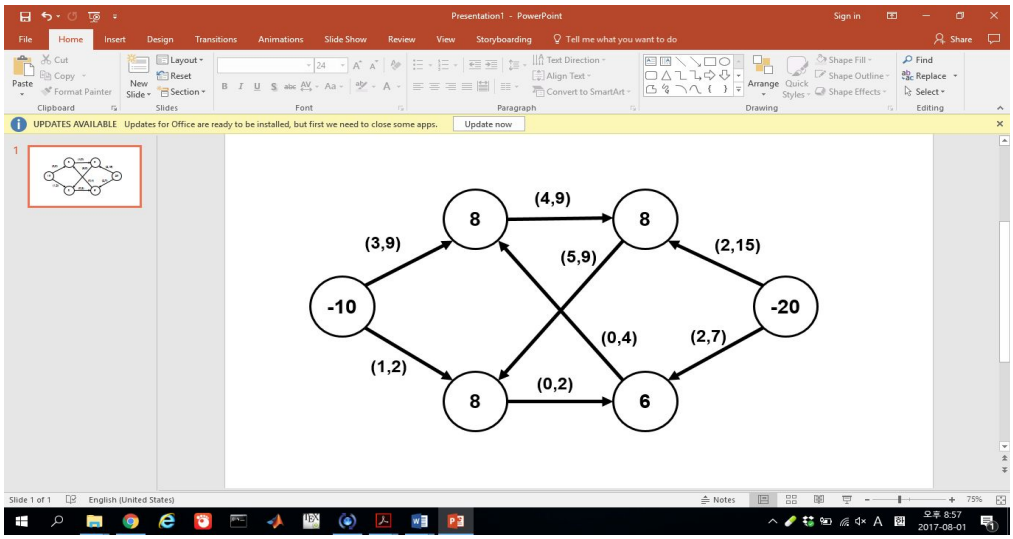
O(n T), pseudopolynomial. They must explain why it's not polynomial

the running time is O(nT). Let M be the number of bits it takes to describe T. Then, M = log(T), so the total running time is O(n2^M), which is why it is pseudopolynomial. Remember, pseudopolynomial means it is polynomial in the size of the input but not in the number of bits needed to describe the input.

O(nT) = 1 point
Pseudopolynomial = 1 pt
describing why it is pseudopolynomial = 2pts

2) 10 pts. (**Youwei + Ang Li**)

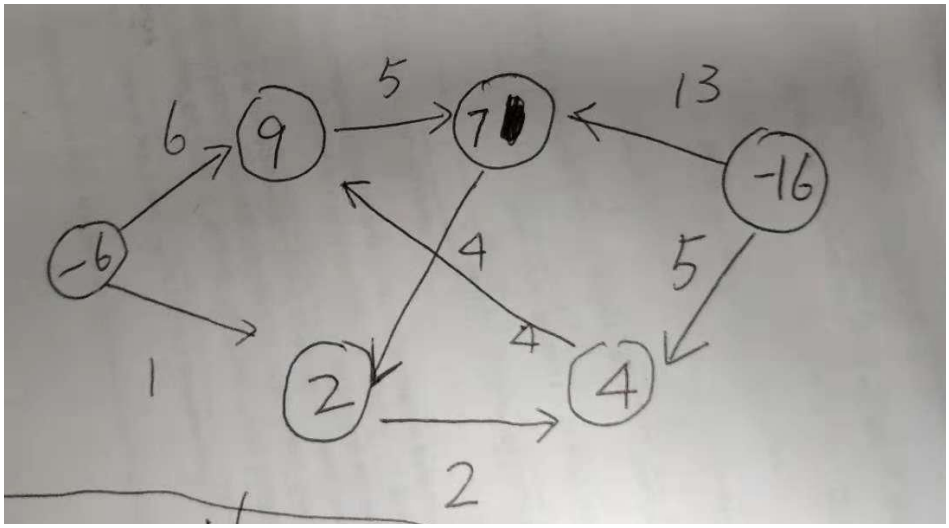
In the network below, the demand values are shown on vertices. Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if a feasible circulation exists or not. If it does, show the feasible circulation. If it does not, show the cut in the network that is the bottleneck. Show all your steps.



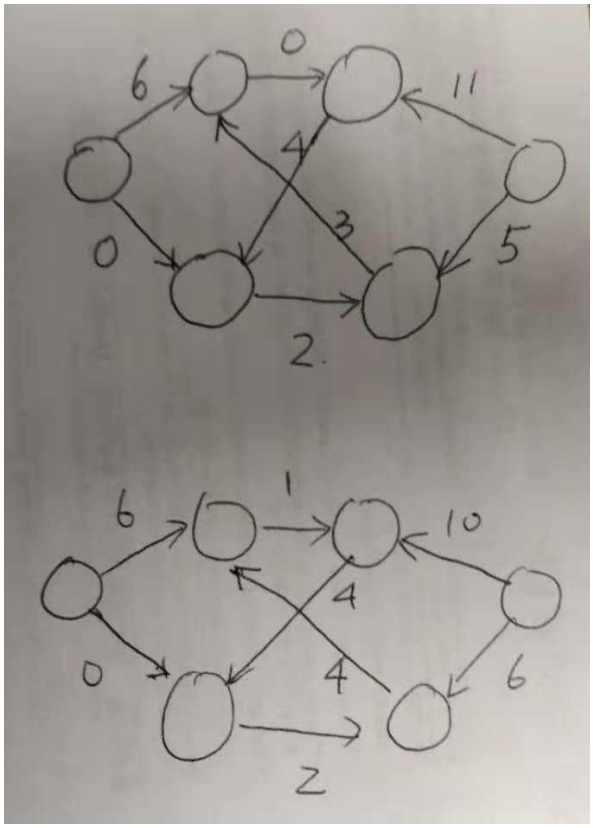
Solution.

First, we eliminate the lower bound from the edges. Then, we attach a super-source s* to each node with negative demand, and a super-sink t* to each node with positive demand. The capacities of the edges attached accordingly correspond to the demand of the nodes.

(5 pts) (A) Turn it into a circulation problem without lower bound.
Describe the node and edge values in the new graph. It is OK without drawing the graph.



(5 pts) (B) Turn it into a max flow problem.
Describe the edge flow values. It is OK without drawing the graph.
Note that the max flow answer is not unique.



(-3 pts) (A) For incomplete node/edge values. (no supersource or supersink etc.)
(-5 pts) (A) For wrong node/edge values.
(-3 pts) (B) For incomplete flow values.
(-5 pts) (B) For wrong flow values (flow exceeds capacity, flow does not follow the demand/supply constraint)

3) 15 pts (**Aida + Raksha Shastry**)

A company has n locations in city X and plans to move some of them to another city Y. The i-th location costs a_i per year if it is in the city X and b_i per year if it is in the city Y. The company also needs to pay an extra cost c_{ij} per year for traveling between locations i and j. We assume that $c_{ij} = 0$ if locations i and j are in the same city. Design an efficient algorithm to decide which company locations should be moved to the city Y in order to minimize the total yearly cost.

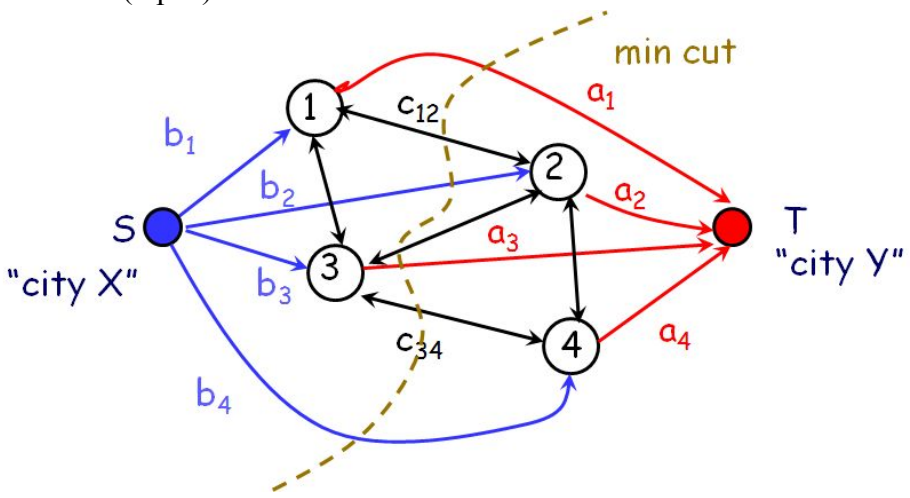
a) Describe how to construct a flow network to solve this problem, including the description of nodes, edges, edge directions and their capacities. (7 pts)

Solution.

Create a complete graph, where each vertex v i is a company location in the city X, and each bidirectional edge has capacity c ij . Connect a source S to all v i with the capacity b i . Connect each v i to a target T with the capacity a i .
Partial credit:
Each item below has (1) point.
1) complete graph
2) each vertex v i is a company location in the city X
3) each edge capacity cij
4) connect S to all vi
5) connect all vi to T
6) capacity s to vij = bi
7) capacity yij to T = ai

How would you decide on which locations should be moved from city X to city Y? (8 pts.)

b) How would you decide on which locations should be moved from city X to city Y? (8 pts.)



Find a min-cut. The total capacity of the ST-cut is
$\sum i \in X \; a\, i + \sum j \in Y \; b\, j + \sum i \in X, j \in Y \; c\, ij$
This total capacity is exactly the yearly cost moving the locations from X to another city Y. Thus, the minimum cut of this network corresponds to the optimal solution of the relocation problem.

Partial credit:
The key ideas to look for:

1) Do they either find a max-flow or min-cut. (1) points
2) Do they specify the cost of relocation? (1) points
3) Do they explain how it relates to the solution of min-cut (or flow)? (1) points
4) Do they explain which cities should move based on the solution of max-flow or min-cut? (1) points
5) Is their answer correct? (4) points

4) 10 pts (**Aida + Bingxue Zhang**)

Given a set S of *n* people, and a set L of pairs of people that are mutually friends. Can these *n* people be seated for dinner around a circular table such that each person will sit next to friends on both sides? Prove that the problem (in short, DINNER) of finding such a sitting arrangement is NP-Complete.

a) Show that DINNER is in NP. (2 pts)

The certifier takes as input a seating arrangement. It verifies in linear time that each person sits next to a friend.

Partial credit:
The key words to look for:
1) seating arrangement/solution to the problem… (1) point
2) checking in linear time/polynomial time. (1) point

a) Describe a polynomial reduction. (4 pts)

Prove HC $\leq_p$ DINNER

Given undirected unweighted G = (V, E) and a HC. Let each vertex corresponds to a person, so S = V. Let each edge on the cycle corresponds to a pair of people who are friends, so L is a set of edges that belong to the HC. The construction takes linear time by performing a traversal on a given HC.

Partial credit:
The key ideas to look for:
1) Do they specify an NP-complete problem (1) point:
- The description of the problem (0.5) point
- is it NP-complete? (0.5) point

2) Do they reduce FROM the NP-complete problem (1) point

3) Do they specify how the an instance of the DINNER problem is constructed? (1) point
4) Is their construction poly time time? (1) point

b) Prove that is a correct reduction. (4 pts)

Claim: G has a HC iff a required sitting arrangement exists.
->) by construction
<-) Given, S and L. Create a graph with S vertices, where two vertices are connected by an edge if those two people are friends. To get a HC on this graph, just follow the sitting arrangement.

Partial credit:
The key ideas to look for:
Each side of the proof (2) points — > Only if they show correctly how the solutions are constructed. For example, if they are reducing from another NP-complete problem, other than HC, they must show that the reduction is correct.

5) 15 pts. (**Youwei + Heena Arora**)
Given an infinite supply of bins, each of which can hold the maximum weight of 1. There are also n objects, each of which has a weight $w_i \leq 1$. Your goal is to place all the objects into bins in such a way, that the total number of used bins is minimized. Formulate the problem as an integer linear programming problem.

(a) Describe your variables for the linear program. (4 pts)

Variables y_i to represent i-th bin. $y_i = 1$ if it's used, and $y_i = 0$ o.w. (2 pts)
Variables x_{ij} to represent j-th item placed into i-th bin (2 pts)

(b) Write out the objective function. (3 pts)

min $(y_1+y_2+\ldots+y_n)$ (2 pts)
in the worst case we need at most n bins. (1 pt. also get the point if "there are n yi variables" is mentioned in (a)).

(c) Describe the set of constraints for ILP. You need to specify the number of constraints needed and describe what each constraint represents. (8 pts)

1. The total weights of objects in each bin should not exceed 1.

$w_1 x_{i1} + w_2 x_{i2} + \ldots + w_n x_{in} \leq \mathbf{y_i}$ for i = 1, …, n (3 pts)

2. each item must be placed in some bin

$x_{1j} + x_{2j} + \ldots + x_{nj} = 1$ for j = 1, …, n (3 pts)

3. $y_i = \{0, 1\}$, $x_{ij} = \{0, 1\}$ (2 pts)

6) 15 pts. (**Yixian Di + Tianlei Wang**)
Write the dual for the following linear program

$\max (3 x_1 + 8 x_2)$

subject to

$x_1 + 4 x_2 \leq 20$

$x_1 + x_2 \geq 7$

$x_1 \geq -1$

$x_2 \leq 5$

Solution.
Convert to the standard form

$\max (3 z_1 - 8 z_2)$

$z_1 - 4 z_2 \leq 1$

$-z_1 + z_2 \leq -3$

$z_1, z_2 \geq 0$

Dual

$\min (y_1 - 3 y_2)$

$y_1 - y_2 \geq 3$

$-4y_1 + y_2 \geq -8$

$y_1, y_2 \geq 0$

7) 10 pts (**Heena Arora + Wentao Zhang**)
Given a Boolean CNF formula with *n* variables and *m* clauses of size 3. Design a randomized algorithm that finds an assignment that satisfy the most clauses. What is the expected number of satisfied clauses?

Solution.
Make a random assignment of each variable to 0 or 1 Let X be the random variable
counting the number of clauses being satisfied. Let Xi

be an indicator random

variable that is 1 when clause i is satisfied, and 0 otherwise.
Since Pr(Xi

= 0) = 1/8, thus Pr(Xi

= 1) = 7/8.

E[X] = E[X1+X2+...+Xn
] = E[X1
]+E[X2
]+...+E[Xm
] = 7/8 m.

10 marks - if correctly explained the probability of any clause being satisfied or not, and if the final expected number of satisfied clauses is correct.
0 marks - if both probability of a single clause being satisfied or not and final expected number of satisfied clauses are incorrect.
9 marks - There were some cases where students answered everything correctly but used variable n to specify number of clauses instead of m (as given in the question).
We deducted 1 point for that. So, 9 marks for them.
8 marks - In few cases students explained the algorithm and the probability of a

single clause correctly, but the final expected number was incorrect. So, -2 for that.

8) 10 pts (**Zimo + Yixian Di**)
Prove or disprove the following statement:
For any set of unique key-priority pairs, there is exactly one treap containing the key-priority pairs which satisfies the treap properties.

Solution. Prove by induction on the number of nodes in the tree. The base case is a tree with zero nodes, which is trivially unique. Assume for induction that treaps with $k - 1$ or fewer nodes are unique. We prove that a treap with k nodes is unique. In this treap, the item x with highest priority must be at the root. The left subtree has items with keys < key[x] and the right subtree has items with keys > key[x]. This uniquely defines the root and both subtrees of the root. Each subtree is a treap of size $\leq k - 1$, so they are unique by induction.

3 pts: attempt induction on the size of the tree
3 pts: point out highest priority element has to be at root
4 pts: left subtree has smaller key than root, right subtree has greater key.

NOTE:
There were also many alternate methods attempted for this problem, whic were graded on a per-individual basis.