CSCI 570 Spring 2015 Discussion 10

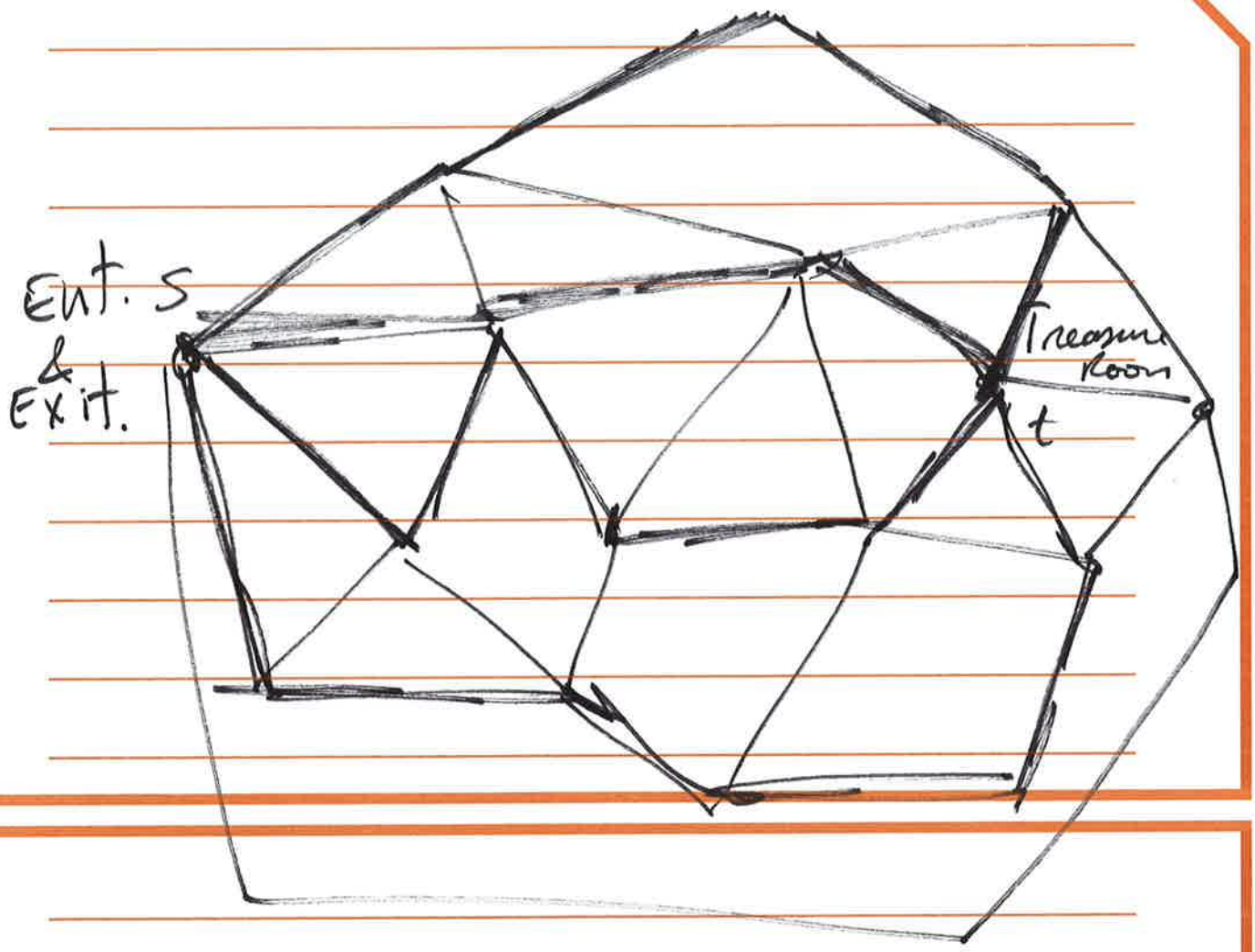
1: Professor Jones has determined that x priceless artifacts (which belong in a museum) are located in a labyrinth. The labyrinth can be thought of as a graph, with each edge representing a path and each node an intersection of paths. All of the artifacts are in the same treasure room, which is located at one of the intersections. However, the artifacts are extremely burdensome, so Jones can only carry one artifact at a time. There is only one entrance to the labyrinth, which is also a node in the graph. The entrance serves as the only exit as well. All the paths are protected by human-eating vines, which will be woken up after someone passes the path, so Jones can only go through each path once.

To obtain a single artifact, Jones would have to find a tour which starts at the entrance, reaches the artifact and leads back again, without getting Jones captured by the vines. If he tries to get another artifact, he must find another tour which does not use the edges he's already traversed. Given the map of the labyrinth and the location of the treasure, give a polynomial-time algorithm that determines how many artifacts Jones can obtain (and how to do so). Justify your algorithm.

2: We're asked to help the captain of the USC tennis team to arrange a series of matches against UCLA's team. Both teams have n players; the tennis rating (a positive number, where a higher number can be interpreted to mean a better player) of the ith member of USC's team is t_i and the tennis rating for the kth member of UCLA's team is b_k. We would like to set up a competition in which each person plays one match against a player from the opposite school. Because we get to select who plays against whom, our goal is to make sure that in *as many matches as possible*, the USC player has a higher tennis rating than his or her opponent. Use network flow to give a polynomial-time algorithm to decide which matches to arrange to achieve this objective.

3: Suppose I am teaching a large class and have many TAs. I want a total of k office hours to be held each week, and I have a set of k hour-long time intervals $I_1, I_2, \ldots I_k$ in which the TA office hours room is available. I have collected, from each TA, a list of which subset of the time intervals he or she can hold office hours. I also know, for each TA j, the maximum m_j hours he or she can hold office hours, and I have for each a minimum l_j hours he or she should be required to hold. Lastly, I have some value H for the total number of hours I'd like held during the week.

Design a polynomial-time algorithm that takes as input an instance of this problem (the time slots, the TA schedules, the sets of l_j and m_j values, and the value H) and determines if there is a valid way to schedule the TA's office hours to respect these constraints.



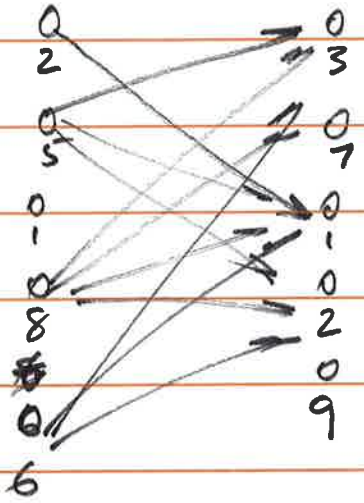
No of artifacts = $\min\left(\left\lfloor \frac{P}{2} \right\rfloor, X\right)$

P = no. of edge disjoint paths

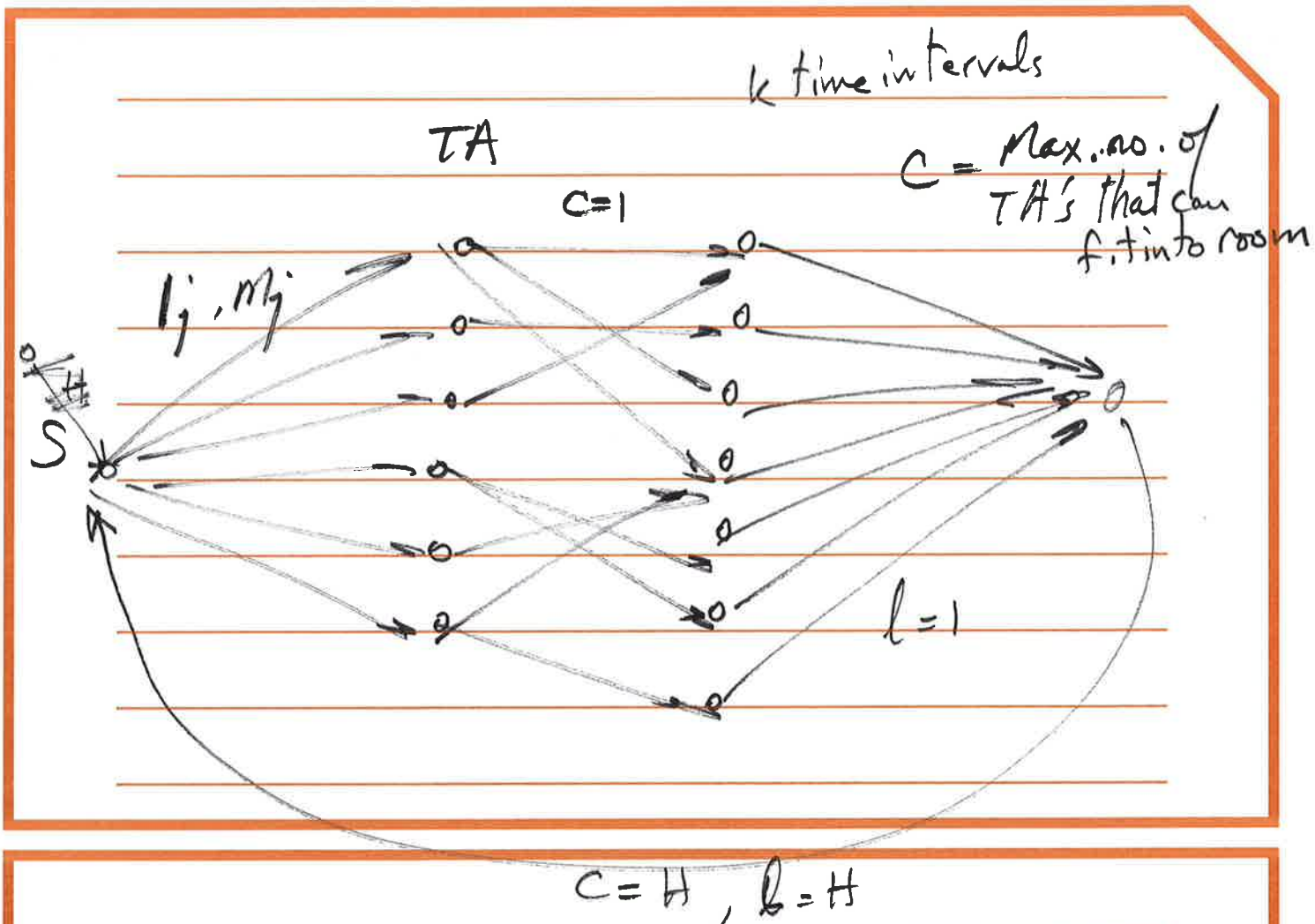
X = no. of artifacts

this takes $O(mn)$

USC UCLA



complexity will be $O(mn)$


k time intervals
TA
C = Max. no. of TA's that can fit into room
C=1
l_j, m_j
S
$l=1$
$C=H, l=H$

Design a polynomial-time algorithm that takes as input an instance of this problem (the time slots, the TA schedules, the sets of l_j and m_j values, and the value H) and determines if there is a valid way to schedule the TA's office hours to respect these constraints.

1: This is a take on the edge-disjoint paths problem. If there are p edge-disjoint paths, then the floor of (p/2) is the number of trips Indy can make. If this is more than x, then some trips are useless, so the answer is min (floor(p/2), x). The goal of giving this question is mostly so students recall the application of flow. There's a great follow-up problem in the textbook for them to look at (although saying it right after this gives away part of the solution) where in O(k logn) time, they have to determine which edges were cut.

2: create a bipartite graph: n vertices for Trojan players, and another n for Bruins players. Add an edge from each Trojan to each vertex that represents a Bruins player that he or she has a higher rating than. Create a source and a sink; connect the source to each Trojan and the sink to each Bruin. Give each edge a capacity of one, and run any polynomial-time version of network flow (actually, even the basic Ford-Fulkerson would be poly time here, because C is $O(n)$ for this graph); the edges with flow are the matches that will be set up. Due to conservation of flow, each player is involved in at most one match (as each player node has either one total incoming, or one total outgoing, capacity).
Unmatched players can be paired in any way desired (or randomly).

~~(I've given a more complicated version of this in the past, adding that one could use min-cut to get the set of vertices that would cause a non-CS person that we can't win every match, by giving a subset of the Trojans players and a smaller subset of the Bruins players, such that the Bruins selected are the only ones that the Trojans selected can beat... I use this problem to discuss bipartite matching and Hall's Th[illegible] when I teach 270)~~

3: Another bipartite matching, with an edge from a vertex like a source to each interval (capacity 1), and an edge from each interval to any TA that can cover it (capacity 1). Then, each TA has an edge to the sink with minimum and maximum based on their hours. Connect the sink to the source-like vertex with an edge of capacity of H and lower bound of H. Technically, the source and sink are a pseudo-source and a pseudo-sink here, and we put an edge sink to source, and call circulation.