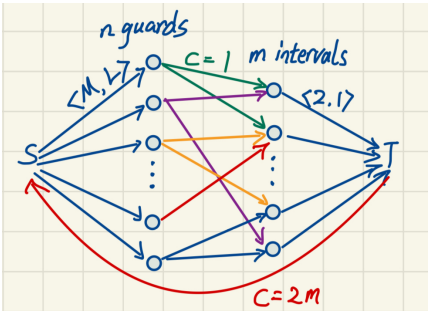
Homework 9

Jiahao Liu March 22, 2024

1. There is a precious diamond that is on display in a museum at m disjoint time intervals. There are n security guards who can be deployed to protect the precious diamond. Each guard has a list of intervals for which he or she is available to be deployed. Each guard can be deployed to at most M time slots and has to be deployed to at least L time slots. Design an algorithm that decides if there is a deployment of guards to intervals such that each interval has either one or two guards deployed. (10 pts)

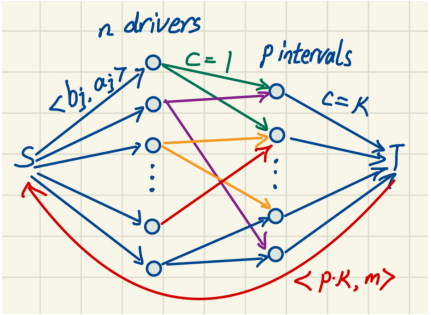
Construct a circulation network G such that G can have a feasible flow iff there is a deployment meets all requirement. The construction will involve the following sets of nodes and edges:
Nodes: n nodes for guards, m nodes for time intervals, and a source node s and a sink node t.
Edges: n directed edges from s to each guard node with capacity M and lower bound L; m directed edges from each time interval node to t with capacity 2 and lower bound 1; Directed edges from each guard node to corresponding time interval node with capacity 1 if this guard is available at this time interval; One directed edge from t to s with capacity 2m;



Then solve the feasible circulation with lower bound problem. If there is a feasible circulation in G, we can find a deployment of guards to intervals. Otherwise, there is no such deployment.

2. Consider LAX, a notoriously busy airport with many arriving passengers who want to get to their destinations as soon as possible. There is an available fleet of n Uber drivers to accommodate all passengers. However, there is a traffic regulation at the airport that limits the total number of Uber drivers at any given hour-long interval to 0 <= k < n simultaneous drivers. Assume that there are p time intervals. Each driver provides a subset of the time intervals he or she can work at the airport, with the minimum requirement of a_j hour(s) per day and the maximum b_j hour(s) per day. Lastly, the total number of Uber drivers available per day must be at least m to maintain a minimum customer satisfaction and loyalty. Design an algorithm to determine if there is a valid way to schedule the Uber drivers with respect to these constraints. (20 pts)

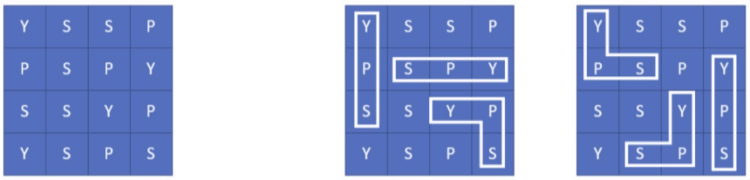
Construct a circulation network G such that G can have a feasible flow iff there is a valid way to schedule the Uber drivers. The construction will involve the following sets of nodes and edges:
Nodes: n nodes for drivers, p nodes for time intervals, and a source node s and a sink node t.
Edges: n directed edges from s to each driver node with capacity b_j and lower bound a_j; p directed edges from each time interval node to t with capacity k; Directed edges from each driver node to corresponding time interval node with capacity 1 if this driver is available at this time interval; One directed edge from t to s with capacity k*p and lower bound m;



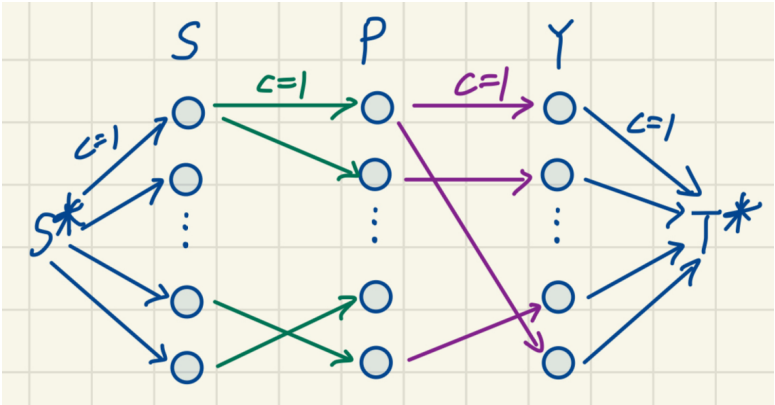
Then solve the feasible circulation with lower bound problem. We can find a valid way to schedule the Uber drivers if and only if there is a feasible circulation in G.

Proof. **Forward Claim:** Assume that there is a valid way to schedule at least m Uber drivers per day. We construct a flow in G as follows. If a driver Ub_i works during a time interval I_j, we create a flow of one unit on edge $(Ub_i, Ij_)$. A particular driver Ub_i may work during several time intervals. Therefore, we set the flow along the edge (s, Ub_i) to the number of time intervals that driver works. We set the flow along the edge (I_j, t) to the number of drivers who work during that time interval I_j. Finally, we set the flow on edge (t, s) to the total number of Intervals Uber drivers serving the airport. Thus, we have constructed a feasible circulation.
Backward Claim: Consider a feasible circulation in G. For each edge (Ub_i, I_j) that carries one unit of flow, driver Ub_i works at hour I_j. Flow on the edge (s, Ub_i) represents the total number hours that driver works. By the flow conservation law, that number is between a_i and b_i. Similarly, the flow along the edge (I_j, t) cannot exceed k, implying that only at most k drivers can work at that hour I_j. □

3. Counter Espionage Academy instructors have designed the following problem to see how well trainees can detect SPY's in an n*n grid of letters S, P, and Y. Trainees are instructed to detect as many disjoint copies of the word SPY as possible in the given grid. To form the word SPY in the grid they can start at any S, move to a neighboring P, then move to a neighboring Y. (They can move north, east, south or west to get to a neighbor.) The following figure shows one such problem on the left, along with two possible optimal solutions with three SPY's each on the right. Give an efficient network flow-based algorithm to find the largest number of SPY's. (20 pts)

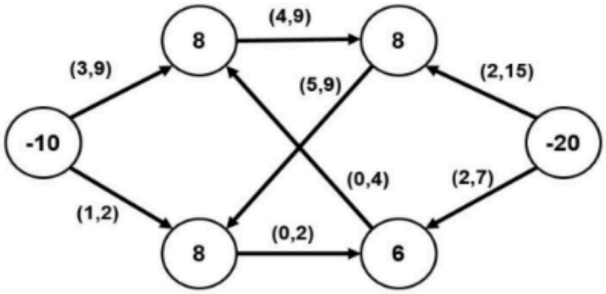


Note: We are only looking for the largest number of SPY not the actual location of the words. No proof is necessary.
We can reduce this to a **Node-Disjoint Path Problem**. Construct a directed graph G with n*n nodes and two special nodes s and t. Find the maximum number of node-disjoint s-t paths in G, which corresponds to the largest number of SPY's.
Nodes: n*n nodes for each letter in the grid (Draw the nodes with same letter in the same vertical line) and two special nodes s and t.
Edges: Directed edges from s to each S node with capacity 1; Directed edges from each S node to its neighboring P nodes (if there is) with capacity 1; Directed edges from each P node to its neighboring Y nodes (if there is) with capacity 1; Directed edges from each Y node to t with capacity 1.

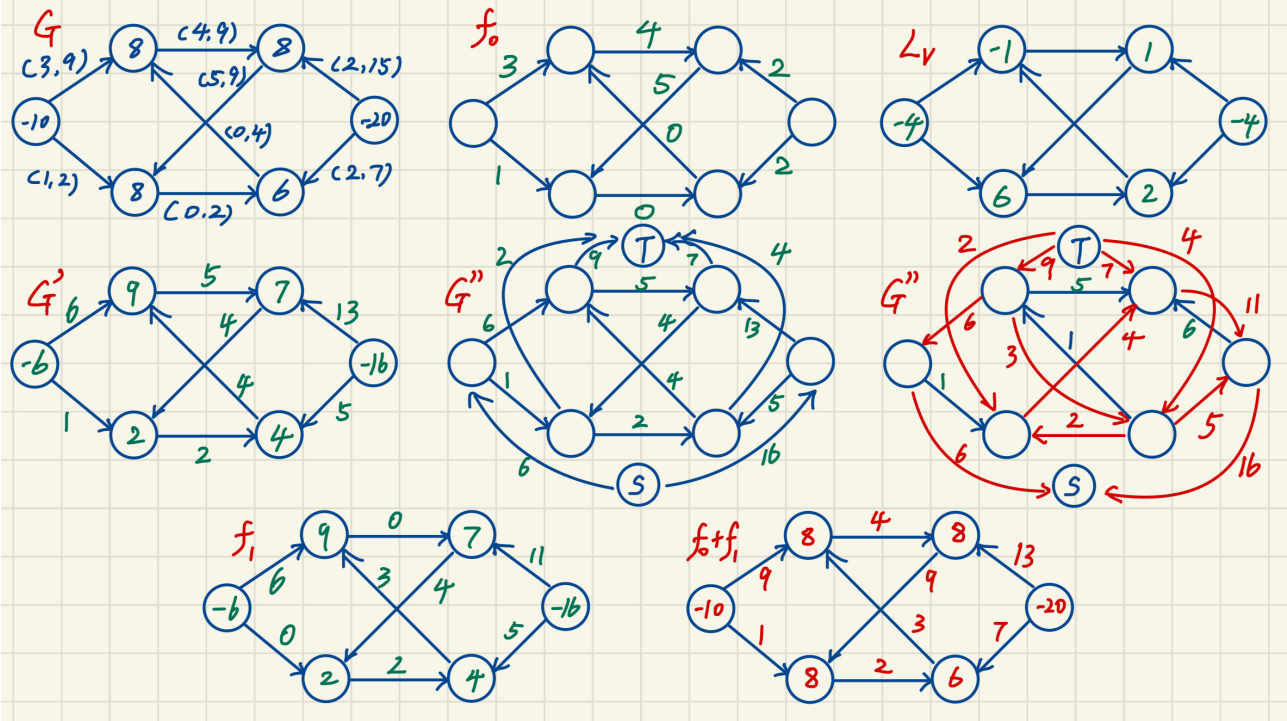


Then solve this Node-Disjoint Paths Problem by splitting each letter nodes into two nodes and adding an edge between them with capacity 1. Here we only need to split all Ps into two nodes and add one edge of capacity 1 between them. Because the S and Y nodes already have constraints of capacity 1 edge to make sure they are used only once. Find the maximum flow in the modified graph. The maximum flow value is the largest number of SPY's.

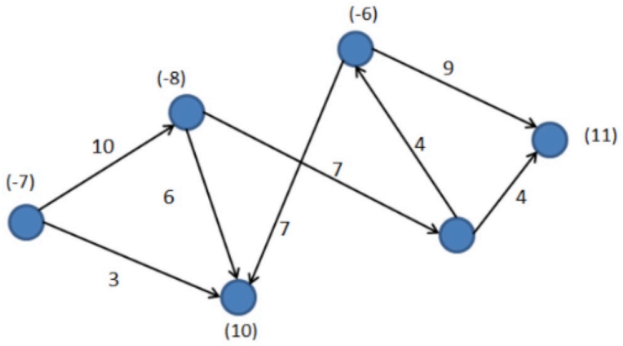
4. In the network below, the demand values are shown on vertices. Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if a feasible circulation exists or not. If it does, show the feasible circulation. If it doesnot, show the cut in the network that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



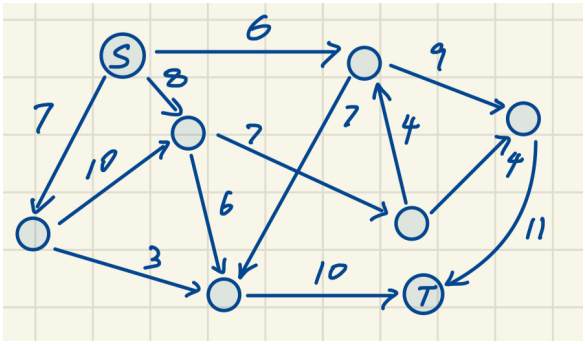
As shown in the following steps, feasible circulation exists.



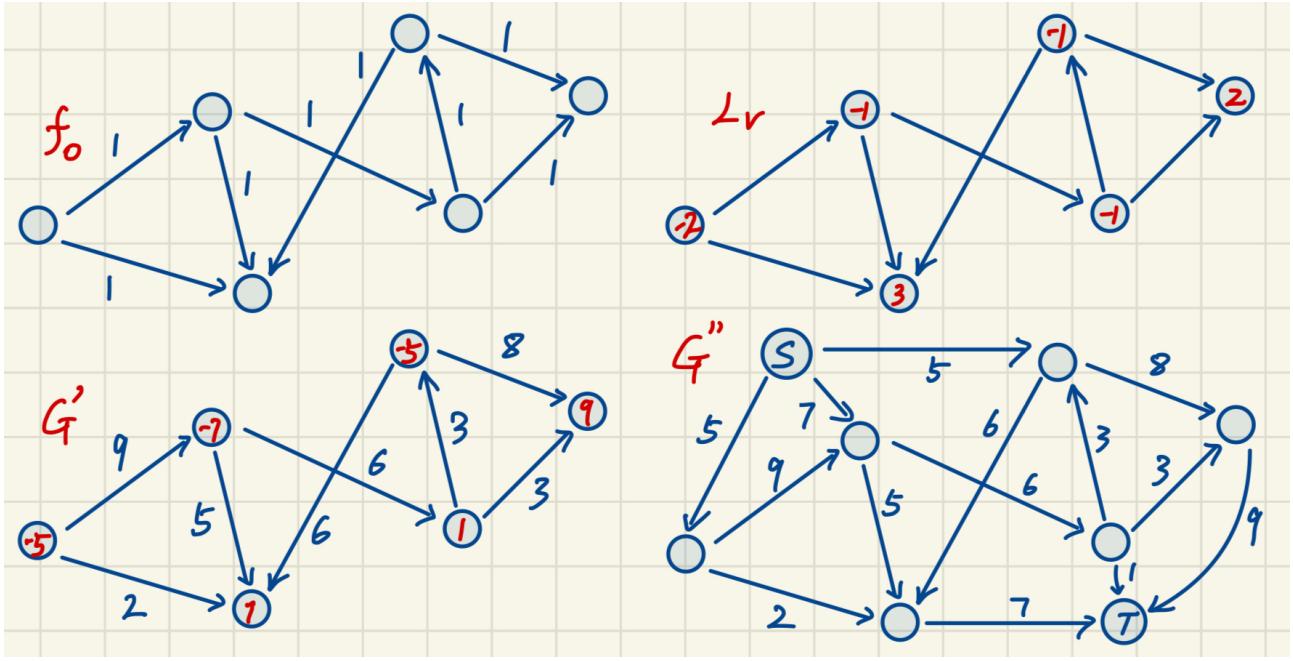
5. The following graph G is an instance of a circulation problem with demands. The edge weights represent capacities and the node weights (in parantheses) represent demands. A negative demand implies source.



(i) Transform this graph into an instance of max-flow problem.



(ii) Now, assume that each edge of G has a constraint of lower bound of 1 unit, i.e., one unit must flow along all edges. Find the new instance of max-flow problem that includes the lower bound constraint.



6. Determine if it is possible for flow to circulate within the given network depicted as graph G. The demand values, indicated on vertices, represent either supply (if positive) or demand (if negative). Each edge also has specified lower bounds on flow and capacity. Demonstrate all necessary steps to ascertain if a feasible circulation exists within this graph. (20 pts)
(a) Reduce the Feasible Circulation with Lower Bounds problem to a Feasible Circulation problem without lower bounds. (b) Reduce the Feasible Circulation problem obtained in part (a) to a Max Flow problem in a Flow Network. (c) Solve the resulting Max Flow problem and explain whether there is a Feasible Circulation in the original G.

(a) First, Push flow f_0 through G, where $f_0(e) = l_e$ to satisfy all lower bounds l_e.
Second, Construct G' with remaining capacity $c'_e = c_e - l_e$ and updated demands $d'_v = d_v - L_v = d_v - (f_0^{in}(v) - f_0^{out}(v))$, where L_v is called flow imbalance at node v.
Find a feasible circulation f_1 in G'. If it exists, then there is a feasible circulation in G. Otherwise, there is no such circulation.
(b) Construct a Flow Network G" with a Super Source node S^* connecting to all nodes v in demand $d_v < 0$ with capacity $|d_v|$ and a Super Sink node T^* connecting from all node v in demand $d_v > 0$ with capacity d_v.
Find the Max Flow f in G", if $v(f) < D = \sum_{v:\ d_v>0} d_v = \sum_{v:\ d_v<0}(-d_v)$, then there is no feasible circulation in G'. If $v(f) = D$, then there is a feasible circulation in G'.
(c) Solve the Max Flow problem in G" by Edmonds-Karp algorithm.

7. USC Admissions Center needs your help in planning paths for Campus tours given to prospective students or interested groups. Let USC campus be modeled as a weighted, directed graph G containing locations V connected by one-way roads E. On a busy day, let k be the number of campus tours that have to be done at the same time. It is required that the paths of campus tours do not use the same roads. Let the tour have k starting locations $A = \{a_1, a_2, \ldots, a_k\} \subset V$. From the starting locations, the groups are taken by a guide on a path through G to some ending location in $B = \{b_1, b_2, \ldots, b_k\} \subset V$. Your goal is to find a path for each group i from the starting location a_i, to any ending location b_j such that no two paths share any edges, and no two groups end in the same location b_j.
(a) Design an algorithm to find k paths $a_i \to b_j$ that start and end at different vertices, such that they do not share any edges.

Given the directed graph G with k starting locations $A = \{a_1, a_2, \ldots, a_k\}$ and k ending locations $B = \{b_1, b_2, \ldots, b_k\}$, we can reduce this problem to a **Edge-Disjoint Paths Problem**. Construct a directed graph G' from G by adding the following nodes and edges:
Nodes: a Super Source node s and a Super Sink node t.
Edges: Directed edges from s to each starting location a_i with capacity 1; Directed edges from each ending location b_j to t with capacity 1; Add capacity 1 to all the directed edge in G.
Then solve the Edge-Disjoint Paths Problem by finding the maximum number of Edge-Disjoint s-t paths in G'. If the maximum number of s-t paths is k, then there is a solution. Otherwise, there is no such solution.
(b) Modify your algorithm to find k paths $a_i \to b_j$ that start and end in different locations, such that they do not share any edges or vertices.
Just **split each node in G' (except s and t) into two nodes** v_{in}, v_{out} and add an edge between them with capacity 1. Then solve the **Node-Disjoint Paths Problem** by finding the maximum number of node-disjoint s-t paths in the modified graph. If the maximum number of node-disjoint s-t paths is k, then there is a solution. Otherwise, there is no such solution.