Homework 9

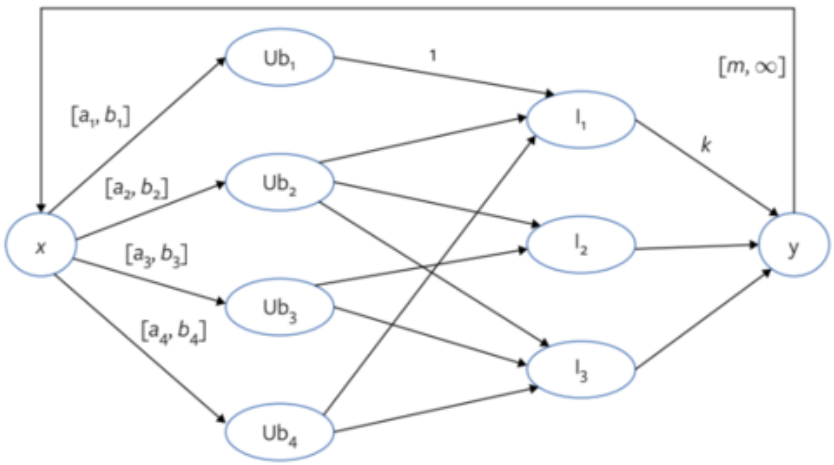
Solutions

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

   Solution:
   We create a circulation network as follows. For the i-th guard, introduce a vertex gi and for the j-th time interval, introduce a vertex tj. If the i-th guard is available for the jth interval, then introduce an edge from gi to tj of capacity 1. Add a source s and a sink t. To every guard vertex add an edge from s of capacity M and lower bound L. From every interval vertex add an edge to t of capacity 2 and lower bound 1. Add an edge from t to s of infinite capacity.
   Next we reduce this to a problem with no lower bounds. We will get the following demands: d(s) = n L, d(gi) = -L, d(tj) = 1, d(t) = -M.
   Then we reduce it to a flow problem, by creating a super source S and super sink T. We connect S to t and gi. We connect s and tj to T. We also reduce capacities.
   Claim: there exists a valid deployment if and only if the above network has a max flow equal to m + n L.

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 aj hour(s) per day and the maximum bj 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)

Solution:



We will reduce the Uber driver's problem to a circulation problem. First, we build a bipartite graph having the drivers Ubi on one side and hour-long time intervals Ij on the other side. We insert the edge between driver Ubi and time interval Ij if the driver prefers to work at that hour. The capacity of this edge is 1. There could be many drivers willing to work at that hour, so having flow 0 on that edge is interpreted as a driver not covering that time interval.
Next, we add two new vertices x and y. Connect x to all Ubi and all Ij to y. The edge (x, Ubi) has lower bound ai and upper bound bi. The edge (Ij, y) has capacity k.

Finally, we add the edge (y, x). The flow on this edge represents the total number of Uber drivers serving the airport.

Claim. There is a valid way to schedule the Uber drivers if and only if there is a feasible circulation in H.

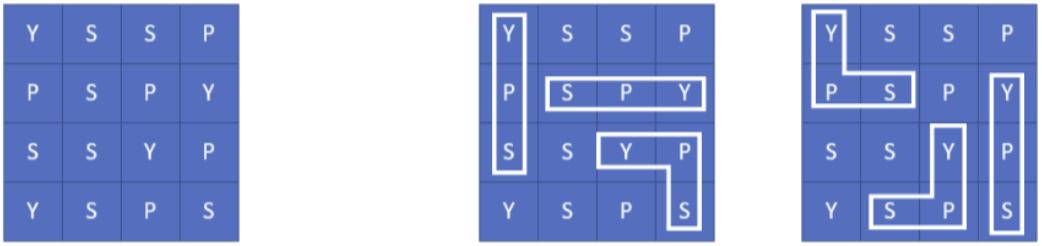
Proof.)
Forward Claim:
Assume that there is a valid way to schedule at least m Uber drivers per day.
We construct a flow in H as follows. If a driver Ubiworks during a time interval Ij, we create a flow of one unit on edge (Ubi, Ij). A particular driver Ubi may work during several time intervals. Therefore, we set the flow along the edge (s, Ubi) to the number of time intervals that driver works. We set the flow along the edge (Ij, t) to the number of drivers who work during that time interval Ij. Finally, we set the flow on edge (t, s) to the total number of Uber drivers serving the airport. Thus, we have constructed a feasible circulation.

Backward Claim:
Consider a feasible circulation in H. For each edge (Ubi, Ij) that carries one unit of

flow, driver Ubiworks at hour lj. Flow on the edge (s, Ubi) represents the total number hours that driver works. By the flow conservation law, that number is between ai and bi. Similarly, the flow along the edge (lj, t) cannot exceed k, implying that only at most k drivers can work at that hour lj.

3) Counter Espionage Academy instructors have designed the following problem to see how well trainees can detect SPY's in an nXn 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 SPYs not the actual location of the words. No proof is necessary.



Solution:

1. Create one layer of nodes for all the Ss in the grid. Connect this layer to a source vertex s, directing edges from s to S.

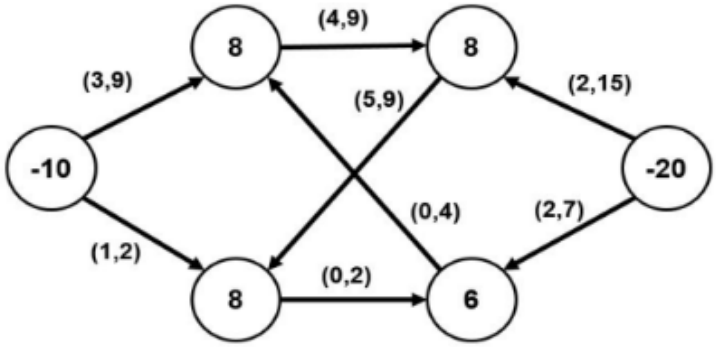
2. Create two layers of nodes for all the Ps in the grid. Connect the first layer to the Ss based on whether or not they are adjacent in the grid, directing edges from S to P. Also, connect the first layer to the second layer by connecting the nodes that correspond to the same location. In other words, we are representing P's as an edge with capacity 1. This is similar to how we solved the node disjoint paths problem.

3. Create a layer of nodes for all the Y s in the grid. Connect these to a sink vertex t, directing edges from Y to t. Also, connect them to the second layer of Ps based if the P and Y are adjacent in the grid, directing edges from P to Y .

Note that we don't need to represent nodes S or P with edges of capacity 1 since there are edges of capacity 1 going into S's and edges of capacity 1 leaving Y's which will force these letters to be picked only once.

Claim: Value of Max Flow in this Flow Network will give us the maximum number of disjoint SPYs.

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

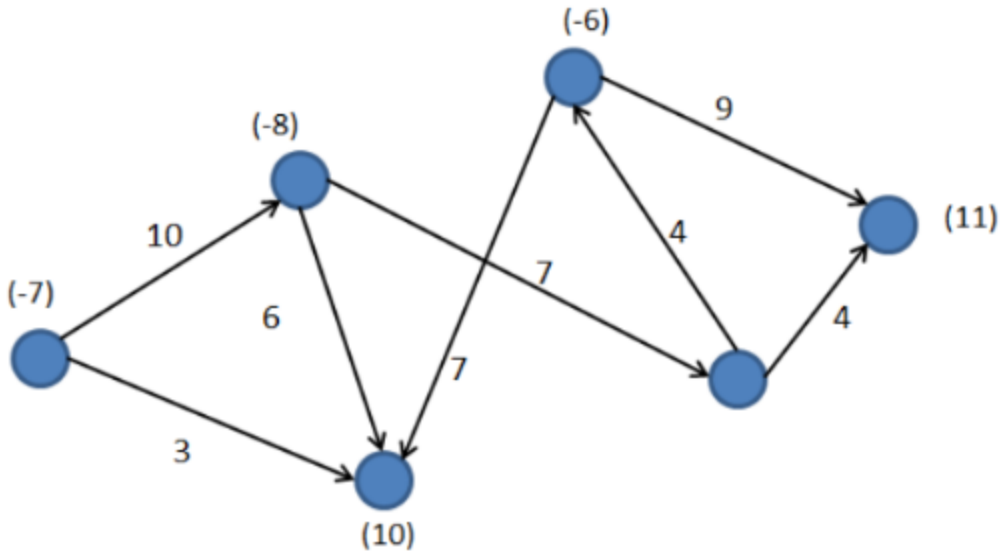


Solution:

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.

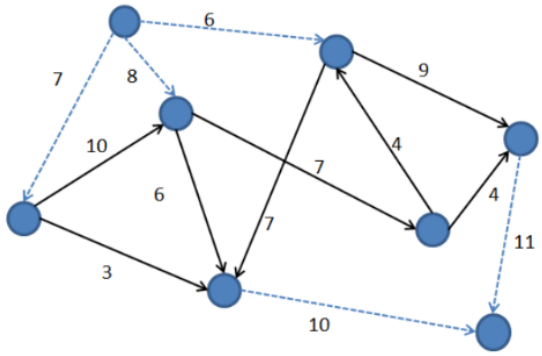
(A) Turn it into a circulation problem without lower bound.
Describe the node and edge values in the new graph.
(B) Turn it into a max flow problem.
Describe the edge flow values.

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



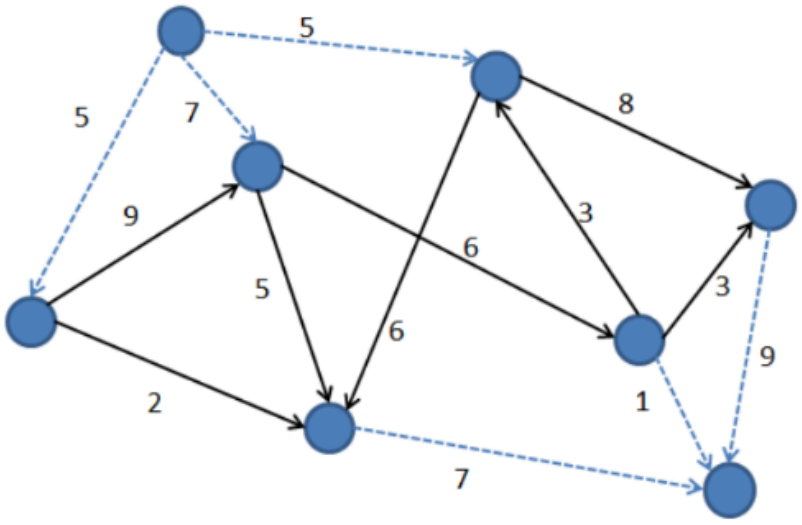
(i) Transform this graph into an instance of max-flow problem.

Solution:

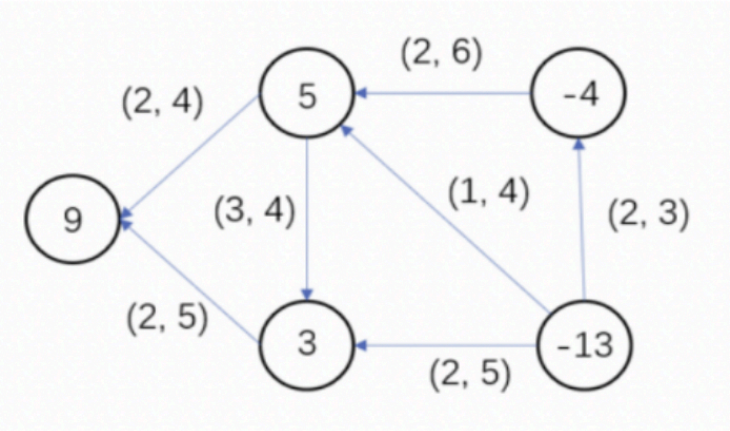


(ii) Now, assume that each edge of G has a constraint of lower bound of 1 unit, i.e., one unit must flow along all edges. Find the new instance of max-flow problem that includes the lower bound constraint.

Solution:



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.



(a) Reduce the Feasible Circulation with Lower Bounds problem to a Feasible Circulation problem without lower bounds.

Solution

i. Assign flow to each edge e to satisfy the lower bound ℓe.

ii. At each node v, calculate Lv = f_in(v) - f_out(v), followed by d'v = dv - Lv to update the demand.

iii. At each edge, calculate c'e = ce - ℓe to update the capacity.

After these steps, we'll have an updated network G', where the demand values and edge capacities are adjusted according to the assigned flow and lower bounds.

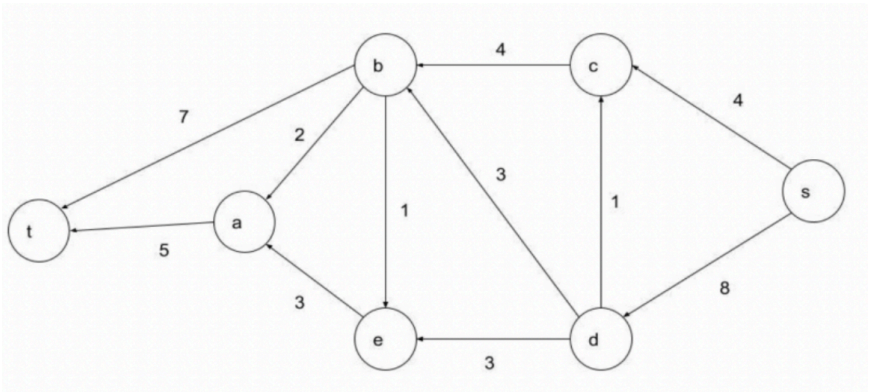
Rubric (8 pts)

•4 pts: If all edge capacities are correctly calculated

•4 pts: If all demand values are correctly calculated

(b) Reduce the Feasible Circulation problem obtained in part (a) to a Max Flow problem in a Flow Network.

Solution: The max flow network is as follows:



Rubric (8 pts)

•2 pts: Add S and T

•2 pts: Connect S with correct nodes

•2 pts: Connect T with correct nodes

•2 pts: Give correct capacities to edge

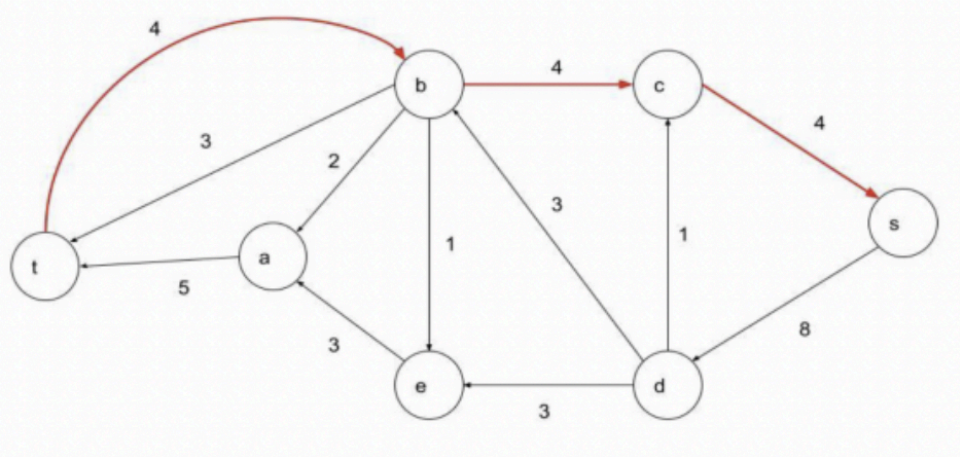
(c) Solve the resulting Max Flow problem and explain whether there is a Feasible Circulation in the original G.

Solution explores three candidate solutions for augmenting paths in the network:

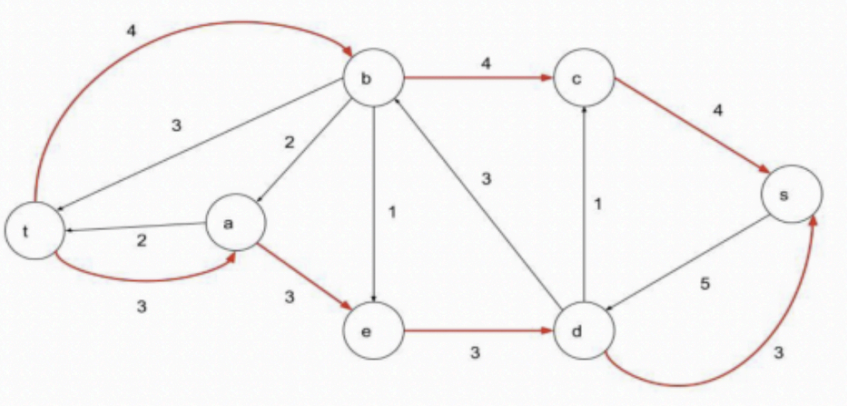
Candidate Solution 1:

- First Augmenting Path: s → c → b → t with flow = 4

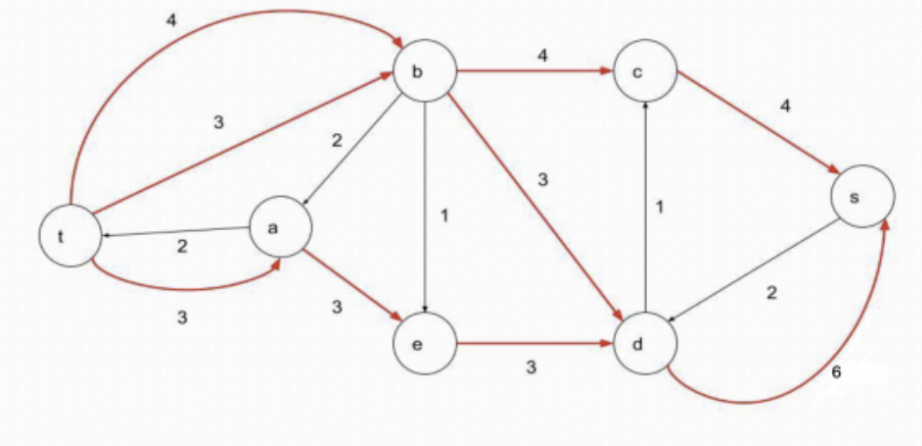
Residual Graph 1:



- Second Augmenting Path: s → d → e → a → t with flow = 3
Residual graph 2:



- Third Augmenting Path: s → d → b → t with flow = 3
Residual graph 3:



Candidate Solution 2:
- First Augmenting Path: s → c → b → t with flow = 4
- Second Augmenting Path: s → d → b → a → t with flow = 2

- Third Augmenting Path: s → d → b → t with flow = 1
- Fourth Augmenting Path: s → d → e → a → t with flow = 3

Candidate Solution 3:
- First Augmenting Path: s → d → e → a → t with flow = 3
- Second Augmenting Path: s → d → b → a → t with flow = 2
- Third Augmenting Path: s → d → c → b → t with flow = 1
- Fourth Augmenting Path: s → c → b → t with flow = 3
- Fifth Augmenting Path: s → d → b → t with flow = 1

All solutions give Max-flow value = 10.

Since the value of Max Flow is less than the total demand value D = 12, there is No Feasible solution in the circulation network G', and therefore there is no feasible circulation in the circulation with lower bounds network G.

Rubric:
- Finding correct Max-Flow and presenting their appropriate residual graphs according to the sequence of augmenting paths: 2 pts
- Correctly concluding "No Feasible Circulation": 1 pt
- Reasoning behind "No Feasible Circulation": 1 pt

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 $A = \{a1, a2, ..., ak\} \subseteq V$. From the starting locations, the groups are taken by a guide on a path through G to some ending location in $B = \{b1, b2, ..., bk\} \subseteq V$. Your goal is to find a path for each group i from the starting location, ai, to any ending location bj such that no two paths share any edges, and no two groups end in the same location bj .
(a) Design an algorithm to find k paths ai → bj that start and end at different vertices, such that they do not share any edges.

(b) Modify your algorithm to find k paths ai → bj that start and end in different locations, such that they do not share any edges or vertices.

Solution:
(a) The complete algorithm is as follows:

1. Create a flow network G′ containing all vertices in V , all directed edges in E with capacity 1, and additionally a source vertex s and a sink vertex t. Connect the source to each starting location with a directed edge (s, ai) and each ending location to the sink with a directed edge (bi t), all with capacity 1.

2. Run Ford-Fulkerson on this network to get a maximum flow f on this network. If |f| = k, then there is a solution; if |f| < k, then there is no solution, so we return FALSE.

3. To extract the paths from ai to bj (as well as which starting location ultimately connects to which ending location), run a depth-first search on the returned max flow f starting from s, tracing a path to t. Remove these edges and repeat k times until we have the k edge disjoint paths. To justify correctness, any argument conveying that there is a flow of size k if and only if there are k edge disjoint paths from A to B is enough.

(b) Duplicate each vertex v into two vertices vin and vout, with a directed edge between them. All edges (u, v) now become (u, vin); all edges (v, w) now become (vout, w). Assign the edge (vin, vout) capacity 1. With this transformation, we now have a graph in which there is a single edge corresponding to each vertex, and thus any paths that formerly shared vertices would be forced to share this edge. Now, we can use the same algorithm as in part (a) on the modified graph to find k edge disjoint paths sharing neither edges nor vertices,if they exist.

Rubric (20 pts)
(a)
• 8 pts: Correct Construction of graph.
• 4 pts: Output k paths.
(b)
• 8 pts: Correct Construction of graph and apply (a) to solve the graph.