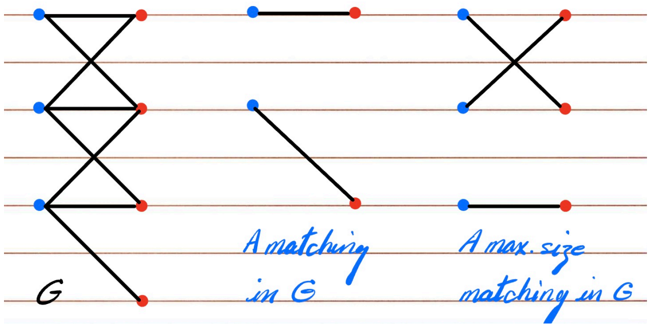
Network Flow Applications

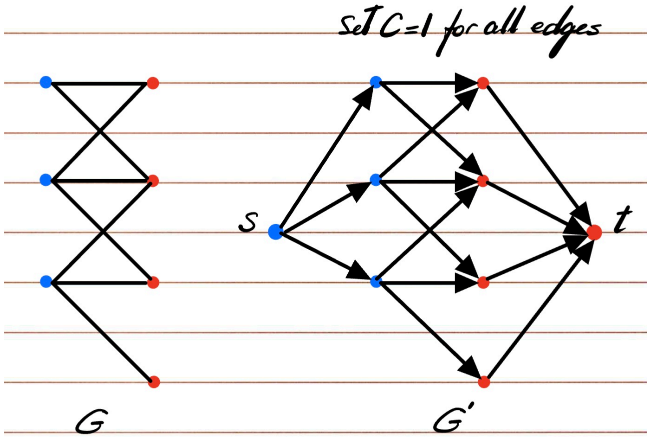
Julius March 20, 2024

1 Bipartite Matching Problem

Bipartite Graph: A graph $G = (V, E)$ is bipartite if V can be partitioned into two sets $V = X \cup Y$ such that every edge in E has one endpoint in X and the other in Y.
Matching: A matching in a graph $G = (V, E)$ is a set of edges $M \subseteq E$ such that each node appears in at most one edge in M.

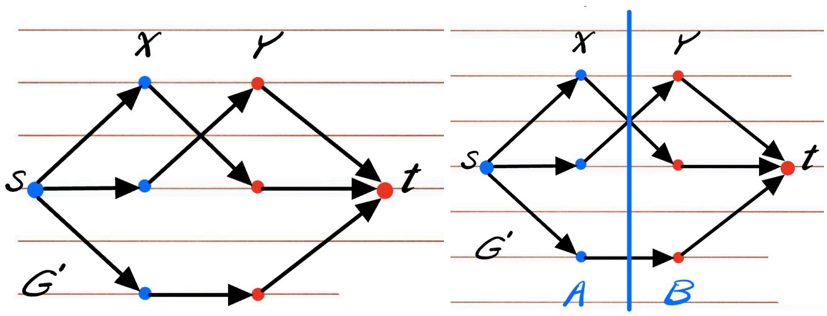


Given a bipartite graph G, how to find a matching of largest possible size?
Design a flow network G' that will have a flow value v(f) = k iff there exists a matching of size k in G. Moreover, flow f in G' should identify the matching M in G.



Solution: find the max flow in G', then edges carrying flow between sets X and Y will correspond to our max size matching in G. To prove this, we will show that G' will have a flow of value k iff G has a matching of size k.

Proof. If we have a matching of size k in G, we can find an s-t flow f of value k in G'.
Just set Capacity = 1 for all edges and use the matching of size k, we will find k independent s-t paths each with capacity 1, thus there is a flow of value k.



Conversely, if we have an s-t flow f of value k in G', we can find a matching of size k in G.
Given flow of value k, there must be k edges on cut (A, B) that carry flow. And these k edges can't share any node, i.e. a matching of size k. □

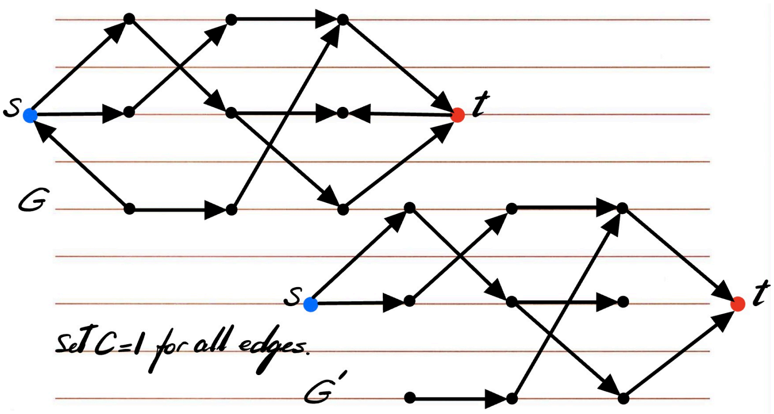
Time Complexity: The time complexity of Ford-Fulkerson algorithm is $O(Cm)$. With all edge capacities are set to 1, we have $C = O(n)$, so the complexity will be $O(nm)$, which is Strongly Polynomial Time. (Running time is bounded by a polynomial in the number of integers in the input.)

2 Edge-Disjoint Paths Problem

A set of paths is edge-disjoint if their edge sets are disjoint, i.e. no two paths share an edge.

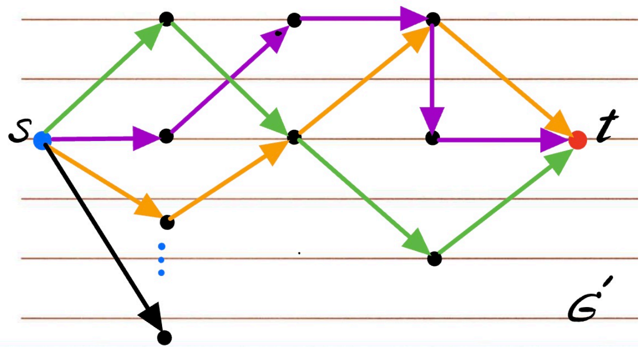
2.1 Directed Graphs

Given a directed graph G = (V, E) and two nodes $s, t \in V$, find the maximum number of edge-disjoint s-t paths in G. Design a flow network G' that will have a flow value v(f) = k iff there exists k edge-disjoint s-t paths in G. Moreover, flow f in G' should identify the edge-disjoint paths in G.



Solution: Find the max flow f in G', then edges carrying flow from s to t will correspond to our edge-disjoint paths in G, and v(f) will equal the maximum number of edge-disjoint s-t paths in G.

Proof. If we have k edge-disjoint s-t paths in G, we can find a flow of value k in G'.
Obviously, since each path can carry 1 unit of flow independently, from s to t in G'.
Conversely, if we have a flow of value k in G', we can find k edge-disjoint s-t paths in G.



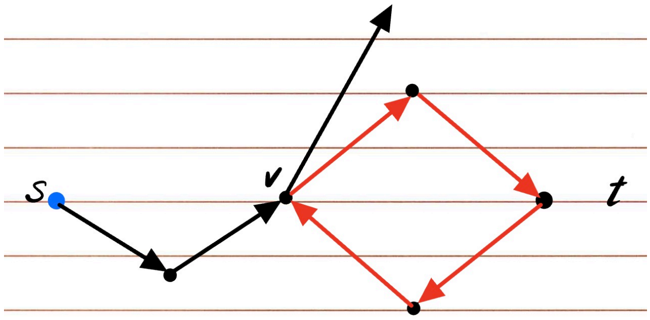
Using conservation of flow, we can follow each unit of flow leaving s and find the s-t path that carries that flow into t. Then remove all edges on that path from G' and repeat this process. Since there are k units of flow leaving s, we can find k edge-disjoint s-t paths. ☐

Time Complexity: The time complexity of Ford-Fulkerson algorithm is $O(Cm)$. With all edge capacities are set to 1, we have $C = O(n)$, so the complexity will be $O(nm)$, which is also Strongly Polynomial Time. (Running time is bounded by a polynomial in the number of integers in the input.)

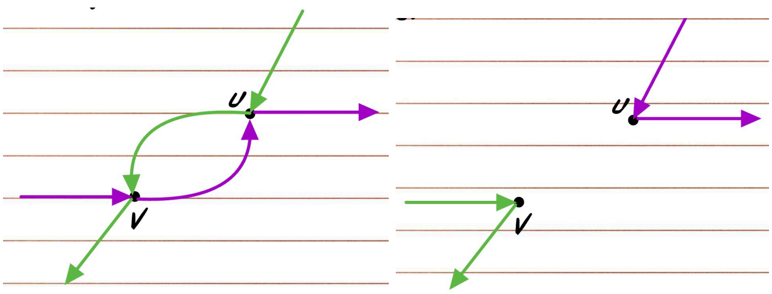
2.2 Undirected graphs

Given an undirected graph G = (V, E) and two nodes $s, t \in V$, find the maximum number of edge-disjoint s-t paths in G.
We can represent each undirected edge uv with 2 directed edges in opposite directions u->v and v->u. However, we will end up with paths that use the edge uv twice, i.e. there might be flow cycle in our found maximum flow.
How to deal with flow cycles?

For each node v in the flow cycle, if we remove edges on the cycle, the flow into v and out of v will decrease by the same amount, i.e. don't violate the conservation of node. Thus, we can remove the cycle and still have a valid flow. And the removed edges won't decrease the total flow value.

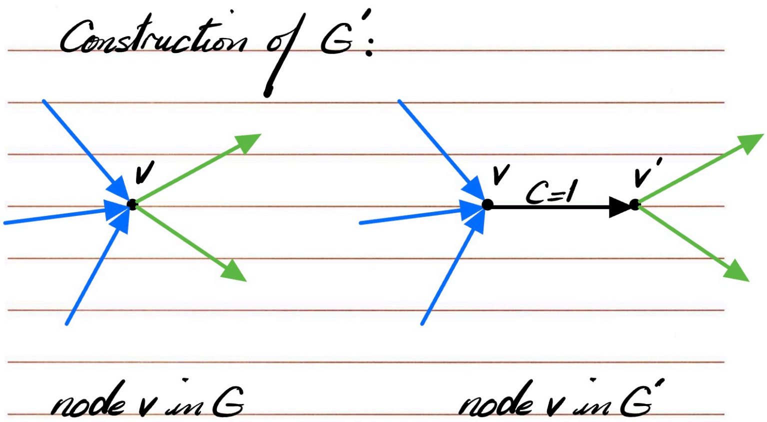


Thus, we can first remove all flow cycles on single edges in the found maximum flow before finding the edge-disjoint paths.



3 Node-Disjoint Paths Problem

A set of paths is node-disjoint if their node sets (except for starting and ending nodes) are disjoint, i.e. no two paths share a node.
Given a directed graph G = (V, E) and two nodes $s, t \in V$, find the maximum number of node-disjoint s-t paths in G. As in edge-disjoint paths problem, the only new challenge here is to design G' such that our paths do not share any nodes.
Recall that the flow network can only constrain use of edges by setting capacity, not nodes. So we can use **Node Splitting**, i.e. split each node v into two nodes v_{in} and v_{out}, and connect them with an edge of capacity 1.



Then we can find the max flow in G' and for each s-t path in max flow, we can just compress the split nodes back to original nodes and get the node-disjoint paths.

4 Circulation & Circulation with Lower Bounds

Circulation Network: a directed graph G = (V, E) with edge capacities c_e and node demands d_v. If $d_v > 0$ node v has a demand of d_v for flow (Sink); If $d_v < 0$, node v has a supply of $|d_v|$ for flow (Source); If $d_v = 0$, node v

is neither a Sink nor Source.
Circulation through edge: circulation f(e) has to follow:
Capacity Constraint:

$$\forall e \in E,\ 0 \leq f(e) \leq c_e$$

Demand Condition:

$$\forall v \in V,\ \sum_{e\ into\ v} f(e) - \sum_{e\ out\ of\ v} f(e) = d_v = f^{in}(v) - f^{out}(v)$$

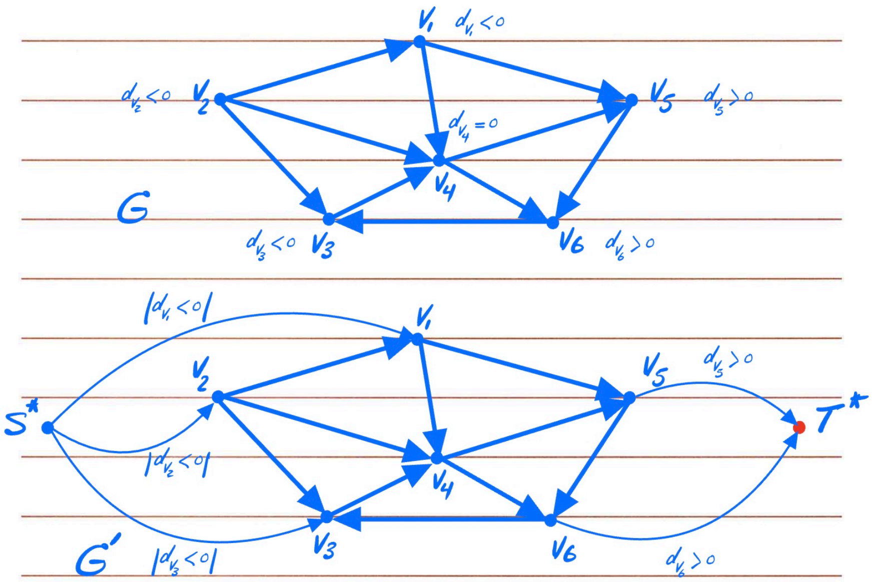
Fact: if there is a feasible circulation with demands $\{d_v\}$, then $\sum_v d_v = 0$

Proof. Consider edge $e = (v, u)$ with circulation $f(e)$, then $f(e)$ contributes to $\sum_v d_v = \sum_v (f^{in}(v) - f^{out}(v))$ once as a positive flow value into u and once as a negative flow value out of v. So all $f(e)$ will cancel out. ☐

Then with $\sum_v d_v = 0 = \sum_v (f^{in}(v) - f^{out}(v))$, we can get **Total Demand Value** $D = \sum_{v:\ d_v > 0} d_v = \sum_{v:\ d_v < 0} (-d_v)$

4.1 Circulation Problem

Given a circulation network G = (V, E) with edge capacities c_e and node demands d_v, is there a feasible circulation in G?
We will reduce the Feasible Circulation problem to Max Flow Problem by constructing a Super Source S^* connecting to all nodes v in demand $d_v < 0$ with capacity $|d_v|$ and a Super Sink T^* connecting from all nodes v in demand $d_v > 0$ with capacity $|d_v|$.



Then Find the Max Flow f in G': if v(f) < D, then there is no feasible circulation with demands $\{d_v\}$ exists in G. If v(f) = D, then we can find a feasible circulation with demands $\{d_v\}$ in G. (Note: s-t cut closest to S^*, T^* each have a capacity of D $\Rightarrow v(f) \leq D$)

Proof. If there is a feasible circulation with demands $\{d_v\}$ in G, we can find a flow of value D in G': Just set flow value equals capacity $|d_v|$ for all added edges in G', we will find D units of flow from S^* to T^*.
Conversely, if there is a max flow of value D in G', we can find a feasible circulation f with demand values $\{d_v\}$ in G: Just remove all edges e from S^* by setting node demand $d_v = -f(e)$ and all edges into T^* by setting node demand $d_v = f(e)$. Then we will have a feasible circulation with demands $\{d_v\}$ in G. ☐

4.2 Circulation with Lower Bounds Problem

Circulation Network with Lower Bounds: a directed graph G = (V, E) with edge capacities c_e, lower bound constrains on flow l_e and node demands d_v.
Circulation through edge: circulation f(e) has to follow:
Capacity Constraint:

$$\forall e \in E,\ l_e \leq f(e) \leq c_e$$

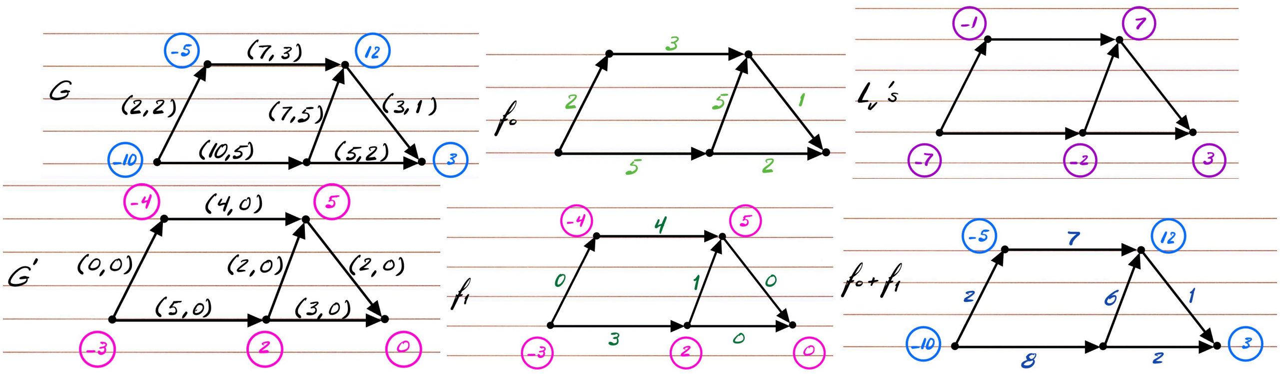
Demand Condition:

$$\forall v \in V,\ \sum_{e\ into\ v} f(e) - \sum_{e\ out\ of\ v} f(e) = d_v = f^{in}(v) - f^{out}(v)$$

Given a circulation network G with edge capacities c_e, lower bounds l_e and node demands d_v, is there a feasible circulation in G?
we will reduce this to Feasible Circulation Problem by finding feasible circulation in two passes.
Pass 1: Push flow f_0 through G, where $f_0(e) = l_e$ to satisfy all lower bounds l_e.
Pass 2: Construct G' with remaining capacity $c_e' = c_e - l_e$ and updated demands

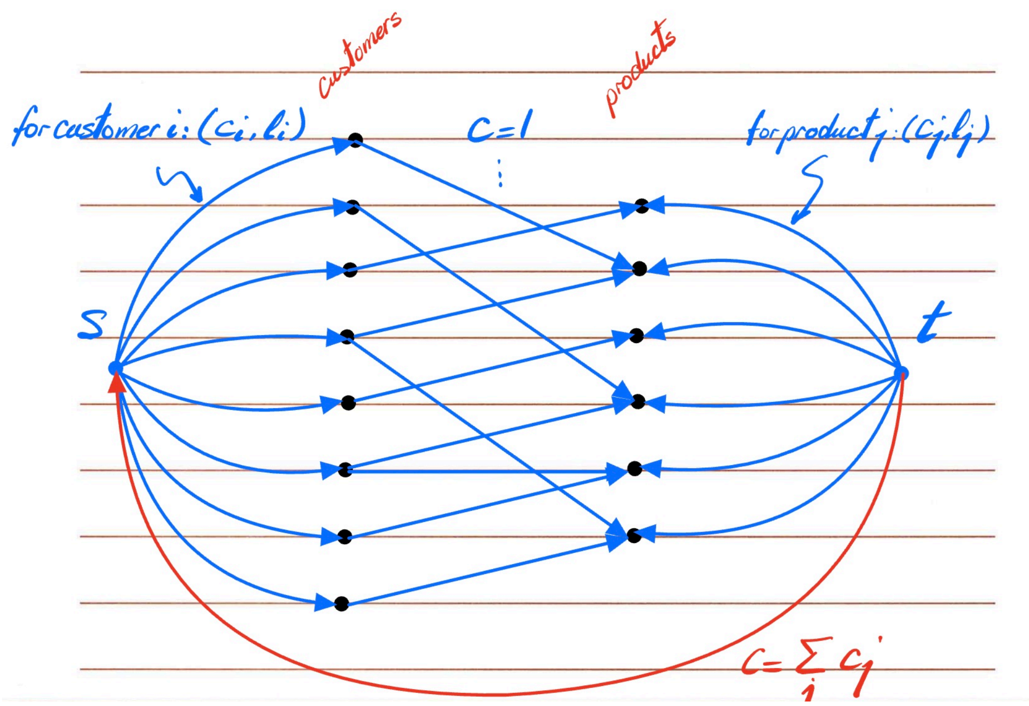
$$d_v' = d_v - L_v = d_v - \Big(\sum_{e \text{ into } v} l_e - \sum_{e \text{ out of } v} l_e \Big)$$

, where $L_v = f_0^{in}(v) - f_0^{out}(v)$ is called flow imbalance at node v. Find a feasible circulation f_1 in G' if it exists. If there is no feasible circulation in G', then there is no feasible circulation in G. Otherwise, feasible circulation in G is $f = f_0 + f_1$
Simple Example: (x,y) indicates edge capacity of x and lower bound of y



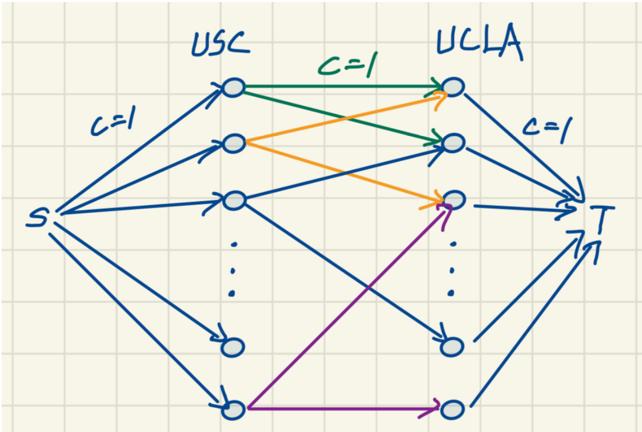
5 Survey Design Problem

Input: Information on which customer purchased which products; Maximum and minimum number of questions to send to customer i (c_i, l_i); Maximum and minimum number of questions to ask about a product j (c_j, l_j)
We want to reduce this problem to a circulation with lower bounds problem by just adding a Source node S and a Sink node T, here the key is what's the **demand value of S and T**?
Remember there is no restriction on the number of total questions asked, so we can't set a specific demand value for S and T, instead we can just **add one edge from T to S** with capacity ∞ or $\sum_j c_j$. Then we can solve this problem by finding a feasible circulation with lower bounds in G.



6 Maximum Matching with Advantage

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 with tennis rating (a positive number, where a higher number can be interpreted to mean a better player) of the i^{th} member of USC's team is t_i and the tennis rating for the k^{th} 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. Our goal is to make as many matches as possible in which the USC player has a higher tennis rating than his or her opponent. Use network flow to give an algorithm to decide which matches to arrange to achieve this objective.
Construct the following flow network G such that G can have a flow of value k iff we can set up k tournament matches with Superiority:

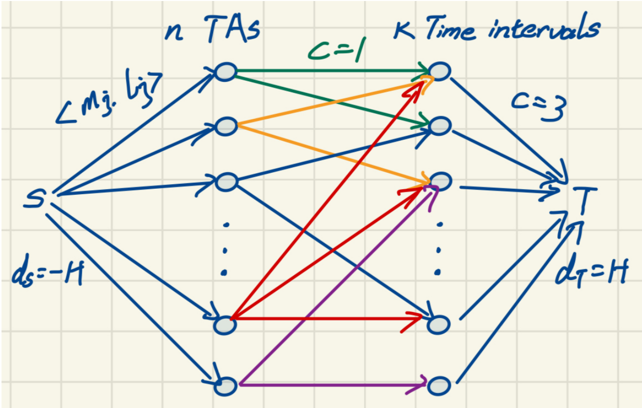


There are n nodes representing USC players and n nodes representing UCLA players, a source and a sink node. There are edges with capacity 1 from source to each USC player, from each UCLA player to sink and from each USC player to UCLA player if his rating is higher than the other.
Then we can find Max Flow f in G. v(f) will be the number of games in which the USC player has a higher ranking than the UCLA player. The edges carrying flow from a USC player to a UCLA player identify such matches. If v(f) < n, then the rest of the players can be matched randomly.

Proof. If we find k matches in which USC has an advantage, we can then find a flow of value k in G.
Given k matches in which USC has an advantage, we can find k edge disjoint s-t paths (since a given player only participates in one match) in G each one capable of carrying 1 unit of flow. Therefore, we can find a flow of value k in G.
Conversely, if we can find a flow of value k in G, we can then find k matches in which USC has an advantage.
Given a flow of value k in G, we must have k edges that carrying 1 unit of flow from a USC player to a UCLA player. Since there is only one unit of flow coming into each node on the USC side and only one unit of flow that can leave a node on the UCLA side, then this edge cannot share any nodes with other edges carrying flow from the USC side to the UCLA side, so each of these edges identify a pair in which the USC player has a higher ranking than the UCLA player. □

7 Circulation with Lower Bounds

CSCI 570 is a large class with n TAs. Each week TAs must hold office hours in the TA office room. There is a set of k hour-long time intervals $I_1, I_2, \ldots, I_k$ in which the office room is available. The room can accommodate up to 3 TAs at any time. Each TA provides a subset of the time intervals he or she can hold office hours with the minimum requirement of l_j hours per week, and the maximum m_j hours per week. Lastly, the total number of office hours held during the week must be H. Design an algorithm to determine if there is a valid way to schedule the TA's office hours with respect to these constraints
Construct a circulation network G such that G can have a feasible flow iff we can schedule TA office hours with all the given constraints. The construction will involve the following sets of nodes and edges:
Nodes: n nodes representing TAs and k nodes representing the hour-long time intervals, a Source node called s with demand -H and a Sink node called t with deman H.
Edges: n directed edges from s to each of the n TA nodes, the edge connecting to TA j will have capacity m_j and lower bound l_j; A directed edge with capacity 1 from each TA to the set of time intervals in which that TA is available; A directed edge with capacity 3 from each time interval to the sink node t.



Then solve the feasible circulation problem. If there is a feasible circulation in G, we can find a feasible assignment of TAs to office hours. Otherwise, there will not be a feasible assignment of TAs to office hours.

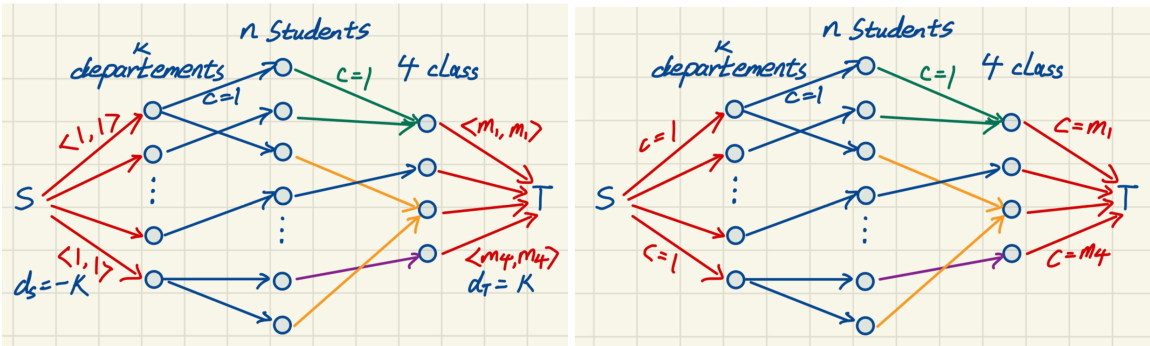
8 Circulation with Same Capacity and Lower Bounds

There are n students in a class. We want to choose a subset of k students as a committee. There has to be m_1 number of freshmen, m_2 number of sophomores, m_3 number of juniors, and m_4 number of seniors in the committee. Each student is from one of k departments, where $k = m_1 + m_2 + m_3 + m_4$. Exactly one student from each department has to be chosen for the committee. We are given a list of students, their home departments, and their class (freshman, sophomore, junior, senior). Describe an efficient algorithm based on network flow techniques to select who should be on the committee such that the above constraints are all satisfied.

Solution 1 reduce to Circulation with Lower Bounds: Construct a circulation network G such that G have a feasible flow iff there is a feasible assignment of students to the committee. The construction will involve the following sets of nodes and edges:

Nodes: 4 nodes representing freshman, sophomore, junior, and senior classes. n nodes representing students, k nodes representing departments, a node called s with demand -k and a node called t with demand k.

Edges: k directed edges from S to each department nodes with capacity 1 and lower bound 1; n directed edges from department to each student with capacity 1; n directed edges from each student to class node with capacity 1; 4 directed edges from each class node to t with capacity m_i and lower bound m_i.



Solution 2 reduce to Max Flow: Construct a flow network G such that G have a flow of value k iff there is a feasible assignment of students to the committee. We only need to modify the edges from S to departments with capacity 1 and edges from class nodes to t with capacity m_i.

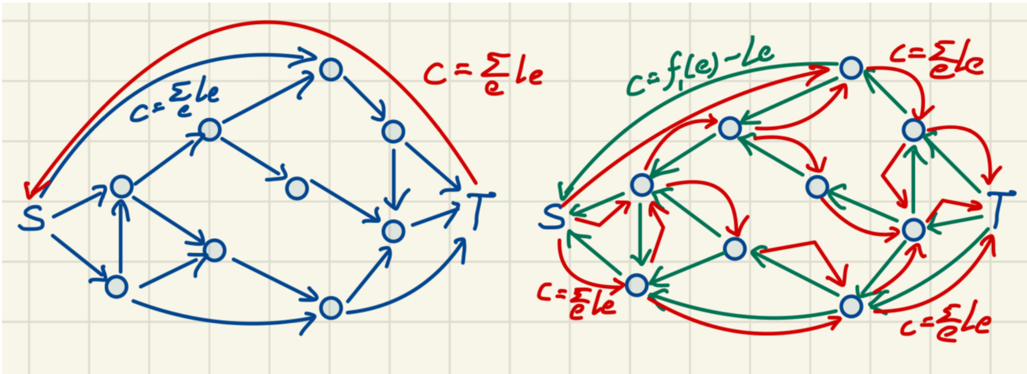
Find the Max Flow f in G. If $v(f) = k$, it means that each edge from s to department is saturated and each edge from four classes to t is also saturated since $m_1 + m_2 + m_3 + m_4 = k$. Then there is a feasible assignment of students to the committee, otherwise there is no such feasible solution.

9 Minimum Flow with Lower Bounds Problem

Given a directed graph $G = (V, E)$ a source node $s \in V$, a sink node $t \in V$, and lower bound l_e for flow on each edge $e \in E$, find a feasible s-t flow of minimum possible value. Note: there are no capacity limits for flow on edges in G. Solve this problem in two passes.

Pass 1: Assign a capacity of $C = \sum_{e \in E} l_e$ to all edges in G and **add an edge from t to s with the same capacity**

C. Find a feasible circulation f_1 in G. Note that C units of capacity will be sufficient to accommodate all minimum flow requirements on all edges, so there will always be a feasible circulation in G. But this feasible circulation may not have the minimum value of flow leaving the source.



Pass 2: Construct G' by reversing the edge directions in G (same nodes and edges but edges are in the opposite direction). Then for each edge e we assign it a capacity of $f_1(e) - l_e$. Note that this amount is the excess flow that can be potentially removed from edge e. However, we also need to use the concept of residual graph to allow us undo some flow. i.e. **For each edge v->u in G', we will also add an edge u->v with capacity C. These edges can help reroute flow if needed.**. We find the max flow f_2 in G'.
Finally, we will find min flow f_{min} by subtracting f_2 from f_1. (i.e. summing up f2 and f1 algebraically) For example, if the flow $f_1(e) = 10$ and $f_2(e) = 3$ (but in the opposite direction), then $f_{min}(e) = 7$.