Problem Set 4 Solutions

Problem 1. **(a)** False. Consider graph with

$$V = \{s, 1, 2, 3, t\},$$

and

$$E = \{(s, 1), (1, 2), (1, 3), (2, t), (3, t)\}.$$

The capacities are $u(s, 1) = 1$, $u(1, 2) = 5$, $u(1, 3) = 6$, $u(2, t) = 7$, $u(3, t) = 8$. There are at least 2 possible maximal flows: 1) $f(s, 1) = f(1, 2) = f(2, t) = 1$ and 0 in the rest; 2) $f(s, 1) = f(1, 3) = f(3, t) = 1$ and 0 in the rest.

(b) False. Consider graph with $V = \{s, t\}$ and $E = \{(t, s)\}$, the edge having a capacity of 1. Then, the initial graph has a max flow of 0, while the modified graph has a max flow of 1.

(c) False. Consider graph with

$$V = \{s, 1, 2, 3, t\},$$

and

$$E = \{(s, 1), (1, 2), (1, 3), (2, t), (3, t)\}.$$

The capacities are $u(s, 1) = 3$, $u(1, 2) = 1$, $u(1, 3) = 1$, $u(2, t) = 1$, $u(3, t) = 1$. For this graph a min cut is $S = \{s, 1\}$. However, if we add a value of $\lambda = 100$ to the capacity of each edge, then the min cut becomes $S = \{s\}$

Problem 2. Let G be the graph under consideration.

(a) We assume that the array creation takes constant time. There are n insert, m decrease-key and n delete-min operations. The insert and decrease-key operations take $O(1)$ time. Delete-min takes $O(1 + d)$ time, where d is the number of empty buckets skipped during the delete-min operation. The bucket number of the last element deleted is D. Thus the total cost of delete-min is $O(m + D)$.

(b) Given the shortest path P from s to v of length d_v, the path $P + vw$ is known to have length $d_v + l_{vw}$. Therefore

$$d_w \leq d_v + l_{vw} \tag{1}$$

and the reduced edge length l_{vw} is non-negative.

(c) For any path $P = (sv_2, v_2v_3, \ldots, v_{k-1}v_k)$, the total reduced edge length is

$$\begin{aligned} l^d_{sv_2} + \ldots + l^d_{v_{k-1}v_k} &= (l_{sv_2} + d_s - d_{v_2}) + \ldots + (l_{v_{k-1}v_k} + d_{v_{k-1}} - d_{v_k}) \\ &= d_s + (l_{sv_2} + \ldots + l_{v_{k-1}v_k}) - d_{v_k} \end{aligned}$$

Therefore all paths to a vertex v have reduced length as the length minus (constant) d_{v_k}. So the shortest path to v is the same and has length $d_v - d_v = 0$.

(d) The scaling algorithm works as follows. We initially start with edge lengths 0, and distance function $d^1(v) = 0$ for all v. In step k, we shift a bit of the length in each edge, and compute distances d_0^{k+1} with reduced edge lengths based on d^k. We use distance function $d^{k+1} = d^k + d_0^{k+1}$ for the reduced costs in the next step. After $\lceil \log C \rceil$ steps the exact distance will be computed. We will now prove the correctness of this algorithm and analyze its running time.

Consider graph $G' = (V, E)$ constructed from the original graph $G = (V, E)$ with edge length $l'_{vw} = \lfloor l_{vw}/2 \rfloor$. In a scaling step, the distances in G' are used to compute reduced edge length in G. Notice that part (b) and (c) of this problem work for any distance function satisfying (1). So if we define the distance function as distance in G', we still have the same shortest paths in G and G'. This proves the correctness of the algorithm.

The length of shortest paths is 0 in the reduced graph G'. This means that in the original graph G the length of the shortest path is at most n. So Dial's algorithm takes $O(m + n) = O(m)$ time. The total time complexity for $\lceil \log C \rceil$ steps is therefore $O(m \log C)$.

(e) If a base b representation is used, there are $\lceil \log_b C \rceil$ scaling steps. The maximum distance D in each shortest path computation is bounded tightly by $n(b - 1)$. Thus the time complexity of our scaling algorithm is $O((m + n(b - 1)) \cdot \log_b C)$. If we set $b = 2 + m/n$ we achieve $O(m \log_{2+m/n} C)$ running time.

Problem 3. **(a)** We give an algorithm to decide if all people can be moved out in t steps. Now, we can increment t to find the shortest time in which all the people can move out.

The algorithm is as follows: given G, construct G_t as follows. For each $v \in V$, make t copies of v: $v_1 \ldots v_t$. Construct an edge from v_i to v_{i+1} at time t with infinite capacity (people can just stay in rooms at a time step). Construct an edge from v_i to w_{i+1} with capacity C if there exists an edge from v to w with capacity C in G.

To test if all the people can get from the source to the sink in t timesteps, we check if the max flow in G_t is equal to the number of people initially at the source. If so, we can move all the people across this graph in t timesteps.

Note that the size of the graph is polynomially large, so the algorithm runs in polynomial time.

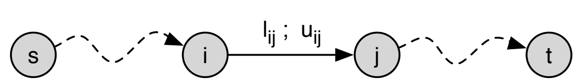
(b) We can use the same overall idea: construct a graph G_t, and compute its max flow. If its max flow is equal to the total number of people we are trying to move, then t time units suffice to move all the people across the graph.

The construction of G_t is the same, except for the following. We create a sink s and source t. Let S be the start vertices, and let T be the sink vertices. We create a link from s to each x_1, for each $x \in S$ with capacity equal to the number of people starting at x. Similarly, we create a link from each x_t (for each $x \in T$) to t with infinite capacities.

(c) Again, the overall idea is the same. But when we construct G_t now, we create edges between the layers in a different way: construct the edge linking v_i to $w_{i+\delta}$ with capacity C if there is an edge between v and w with transit time δ.

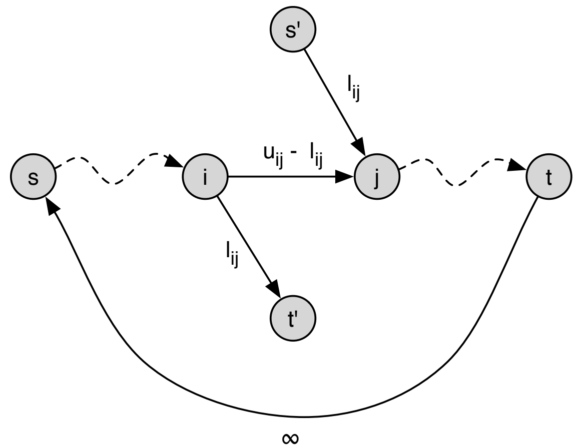
Problem 4. 1. As suggested by the hint, we first construct some feasible flow on the graph G, and then construct a minimum flow using the feasible flow.

We find a feasible as follows. Note that just finding a max flow from s to t is not sufficient as that does not guarantee that the flow flows along the correct edges (to satisfy the minimums). Instead, we modify the original graph G to get a graph G' as follows. For each edge (i, j) in G, we change the capacity $u'_{ij} = u_{ij} - l_{ij}$. Then we drop the minimum capacities on all edges. Thus, in essence, we are forcing a flow of l_{ij} across each of the edges, but we have a surplus and deficit of l_{ij} at i and j respectively. What ways could we get rid of this deficit? Well, we could find a path from a surplus to a deficit node in G'. Or, we could find a path from a surplus node j to the sink t and the source s to the surplus node i. We want to solve this problem using max flows. So, we add to G' a new source s' and a new sink t'. We add edges (i, t') and (s', j) with capacities $u'_{it'} = u'_{s'j} = l_{ij}$[1]. Finally, we add an edge (t, s) (between the original source and sink) with unbounded capacity $u'_{ts} = \infty$. If we start with the graph:



We would end with the graph

[1]We may add more than one edge (i, t') for example, or we can add a single edge with the sums of the relevant capacities. More formally, we could say that $u'_{it'} = \sum_j l_{ij}$ and $u'_{s'j} = \sum_i l_{ij}$.



Then all we need to do is find a max flow from s' to t'. We saturate all the edges (s', j) if and only if for every node j with a surplus, there are path(s) to t and/or some other deficit node(s) with enough capacity. Thus, we just check, are all these edges saturated? If no, there is no feasible flow on the graph G'. If yes, then there is a feasible flow, AND we know what it is. All we need to do is drop all the extra edges we added, look at the flow f' we found on G', and add the original edge minimums. For example, if we find flow of f'_{ij} across an edge (i, j), then the flow in the original graph is just $f = f'_{ij} + l_{ij}$. Another way of thinking about this is that going from s' to t' in the graph G' is equivalent to going from s to t in the original graph, because we move the flow from (s', j) and (i, t') to the edge (i, j).

Next, we convert the feasbile flow into a minimum flow. Let our feasible flow on any edge from i to j be f_{ij} (in gross flow formulation). We construct a "reverse residual graph" by taking every edge from i to j in the original graph G, and replacing it with two edges in G': one edge from i to j with capacity $u_{ij} - f_{ij}$, and one edge from j to i with capacity $f_{ij} - l_{ij}$. We might get two edges by this construction going from i to j, so we combine the edges by adding the capacities. So the reverse residual graph has an edge from i to j with capacity $u_{ij} - f_{ij} + f_{ji} - l_{ji}$.

Then we compute a normal maximum flow g_{ij} from t to s in G', and add the result back into the flow in G to get a flow f'. When we add the flow back, we decompose g into $g_{ij} = g'_{ij} + h_{ij}$, where $0 \leq g'_{ij} \leq u_{ij} - f_{ij}$ and $0 \leq h_{ij} \leq f_{ji} - l_{ji}$. We add the flow back by adding g' in the forward direction and subtracting out h_{ji} in the backward direction. So the new flow from i to j in our original graph G is

$$f'_{ij} = f_{ij} + g'_{ij} - h_{ji}.$$

The inequality relationships we have are

$$\begin{aligned} l_{ij} &\leq f_{ij} \leq u_{ij} \\ 0 &\leq g'_{ij} \leq u_{ij} - f_{ij} \\ 0 &\leq h_{ij} \leq f_{ji} - l_{ji} \end{aligned}$$

Adding f_{ij} to both sides of $g'_{ij} \leq u_{ij}$ and using the fact that $h_{ji} \geq 0$ tells us that $f'_{ij} \leq u_{ij}$. Adding f_{ij} to the inequality $-h_{ji} \geq l_{ij} - f_{ij}$ and using the fact that $g_{ij} \geq 0$ gives us $f'_{ij} \geq l_{ij}$. Therefore, when we add our flow back, we still get a feasible flow.

This flow f' we compute must be a minimum flow. Otherwise, we could reduce the flow further by finding some path from t to s in G. This corresponds to finding some augmenting path in G' from t to s, suggesting that the flow we computed in wasn't a maximum flow for G'. This is a contradiction, so f' must be a minimum flow.

2. **Claim 0.1** *Let the lower bound on the cut capacity of an s-t cut S be defined as $L(S) = \sum_{(i,j)\in S\times T} l_{ij} - \sum_{(i,j)\in T\times S} u_{ij}$. Then the minimum value of all feasible flows from node s to t equals the $\max_S L(S)$.*

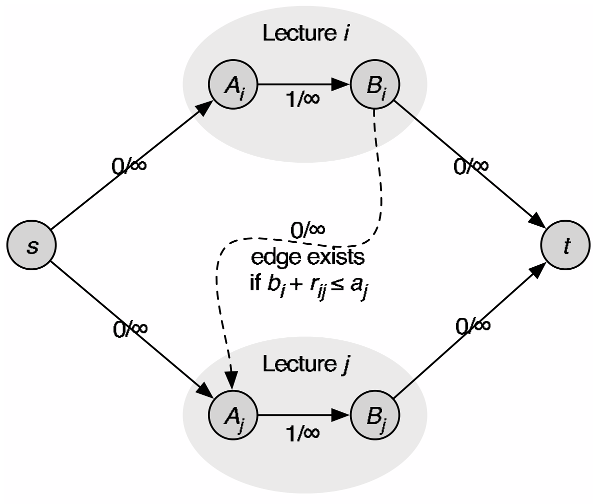
   Proof. We show that for any flow, $f >= \max_S L(S)$ and that min flow $f \not> \max_S L(S)$, which implies that minimum value of all feasible flows from s to t equals $\max_S L(S)$.

   Suppose we have a flow f. Then we want to show that $f \geq \max L(S)$. Instead, we show that $f \geq L(S)$ for any S. Consider any cut S. Since the flow is feasible, we satisfy all the minimum capacities. So, the flow from S to T is at least $\sum_{(i,j)\in S\times T} l_{ij}$. Since the flow is feasible, the most that can be flowing from T to S is bounded by the maximum capacities $\sum_{(i,j)\in T\times S} u_{ij}$. Thus, the net flow across the cut is $\geq \sum_{(i,j)\in S\times T} l_{ij} - \sum_{(i,j)\in T\times S} u_{ij}$. Therefore $f \geq L(S)$.

   Suppose that we have a min flow f. Assume for the sake of contradiction that $f > \max L(S)$. Then for all cuts $f > L(S)$. $f > \sum_{(i,j)\in S\times T} l_{ij} - \sum_{(i,j)\in T\times S} u_{ij}$ implies that either we are sending more flow than the minimum from S to T, or we are not sending the maximum amount back from T to S. Consider the first case. Then our modified residual graph G'_f from part (a) has extra capacity available from T to S on the residual (reverse) edges. In the second case, we also have capacity available from T to S. Since there is capacity available across ALL CUTS going from T to S, there is an augmenting path in G'_f from t to s,[2] and we don't have a min flow. ∎

3. For lecture i, we create two vertices in our graph: A_i to represent the start of the lecture and B_i to represent the end. We draw and edge (A_i, B_i) for every lecture i, with a minimum flow $l(A_i, B_i) = 1$ and a max flow $f(A_i, B_i) = \infty$. We create a source s and a sink t. We create edges (s, A_i) from the source to the start of every class, with unconstrained capacities $l(s, A_i) = 0$ and $f(s, A_i) = \infty$. Similarly, we add edges (B_i, t) from the end of every lecture to the sink, with unconstrained capacities $l(B_i, t) = 0$ and $f(B_i, t) = \infty$. Finally, we introduce edges from the end of lecture i to the start of lecture j if it is possible to commute from lecture i to lecture j. That is, (B_i, A_j) exists if $b_i + r_{ij} \leq a_j$. Again, these edges have unconstrained capacity. The graph looks like:

[2] The max-flow min-cut theorem tells us that $|f| \neq u(S)$ for any S implies that there is an augmenting path in the residual graph.



Now we just solve for the min flow. We claim that the min flow is the minimum number of students necessary to attend all the lectures. Basically, each unit of flow represents a student. We argue correctness by showing that our graph has the same constraints as the students. In particular, a student can only attend two lectures i and j if the commute time allows it. Similarly, we only have an edge in the graph allowing flow between lectures i and j if the commute times allow it. A student can choose any lecture to be the first one he goes to (modeled by edges from s to all lecture starts), and he can choose any lecture to be his last (modeled by edges from lecture ends to t). Finally, the only other constraint we have is that at least one student must be in attendance at each lecture. By creating two vertices for each lecture and having an edge with min capacity of 1 between lecture start and end, we guarantee at least one unit of flow across the lecture—that is, we guarantee that at least one student sits through the entire lecture.