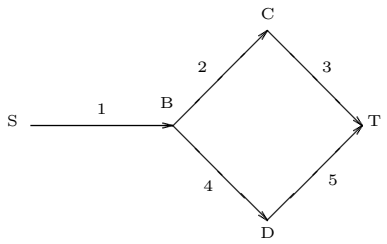
6.854 Advanced Algorithms **Petar Maymounkov**

Problem Set 4 (October 7, 2005)

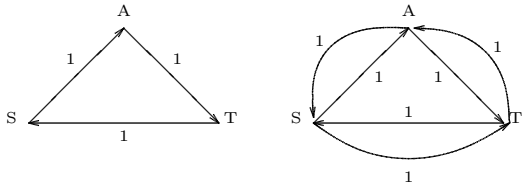
With: Benjamin Rossman, Oren Weimann, and Pouya Kheradpour

Problem 1a. FALSE



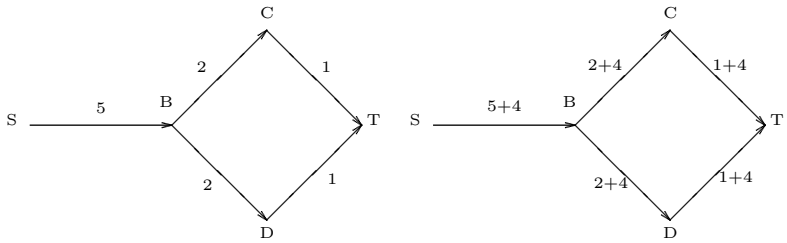
Two different max flows for example are SBCT and SBDT of flow 1.

Problem 1b. FALSE



Max flow in original graph is 1, and in derivative graph is 2.

Problem 1c. FALSE



Min cut in the original graph is $S = \{S, B, C, D\}$, and in the derivative graph is $S = \{S\}$.

Problem 2a. SHORTEST PATH performs n inserts, n delete-mins and m decrease-keys, where the keys are the "current" path lengths. Dial's data structure supports inserts and decrease-keys in $O(1)$ time. Delete-mins are suported in total time D (the maximum distance), because each delete-min scans forward through the array (of size D) to find the first non-empty bucket. Since the minimum element is non-decreasing, each array bucket gets scanned ever only once. Runtime $O(m + n + D) = O(m + D)$.

Problem 2b.

1. Suppose not
2. Then $l_{vw} + d_v - d_w < 0$
3. Then $d_w > d_v + l_{vw}$
4. Contradiction:
5. We know that $d_w \leqq d_v + l_{vw}$, since the shortest path to v appended by the edge vw is a valid path from s to w.

Problem 2c. For any path $s, v_0, \ldots, v_k$, the path length under the reduced distnace works out (after telescoping and observing that $d_s = 0$) to be $l_{sv_0} + l_{v_0v_1} + \cdots + l_{v_{k-1}v_k} - d_{v_k}$.

Since edge weights are non-negative, the shortest possible path has length 0, and in fact this length is achieved if and only if the path is the same a shortest path from s to v_k under the regular edge weights.

Therefore, there is a 1-to-1 correspondance between shortest paths in the regular and reduced edge variants of the graph. And thus the lengths of all shortest paths in the reduced graph are 0.

Problem 2d. We do $\log C$ iterations of SHORTEST PATH over the given graph with edge lengths x^i_{vw} on each iteration i, where:

$$x^i_{vw} = l^i_{vw} + 2d^{i-1}_v - 2d^{i-1}_w,$$

Where l^i_{vw} are the i most significant bits of l_{vw}, and d^i_v is the length of the shortest path to v under edge lengths are l^i_{vw}.

Note that x^i_{vw} is non-negative because:

$$\begin{aligned} x^i_{vw} &= l^i_{vw} + 2d^{i-1}_v - 2d^{i-1}_w \\ &= 2\left(l^{i-1}_{vw} + d^{i-1}_v - d^{i-1}_w\right) + \epsilon, \text{ where } \epsilon \in \{0, 1\} \end{aligned}$$

And the part in the brackets is non-negative as shown in part 2b.

Inductive hypthesis: At the end of step $i-1$ we have knowledge of all d^{i-1}_v.

Inductive step: At step i, we run SHORTEST PATH under edge lengths x^i_{vw}, and let r^i_v denote the length of the shortest path to v. We have that:

$$\begin{aligned} r_v^i &= \arg\min_{\text{paths}} \left(x_{sw_0}^i + x_{w_0w_1}^i + \cdots + x_{w_kv}^i\right) \\ &= \arg\min_{\text{paths}} \left(l_{sw_0}^i + l_{w_0w_1}^i + \cdots + l_{w_kv}^i - 2d_v^{i-1}\right) \\ &= \arg\min_{\text{paths}} \left(l_{sw_0}^i + l_{w_0w_1}^i + \cdots + l_{w_kv}^i\right) - 2d_v^{i-1} \\ &= d_v^i - 2d_v^{i-1} \end{aligned}$$

Therefore,

$$d_v^i = r_v^i + 2d_v^{i-1}$$

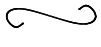
This proves correctness of our algorithm. Next we show that the longest shortest path at iteration i is at most n (under the x_{vw}^i edge length). We have $r_v^i = d_v^i - 2d_v^{i-1}$. Also we know that $d_v^i \leqq 2d_v^{i-1} + n$, because the shortest path to v in the $(i-1)$-st iteration has length at most $2d_v^{i-1} + n$ in the i-th iteration. We conclude that $r_v^i \leqq n$.

Therefore, the running time (using Dial's algorithm) on an iteration is at most $O(m+n)$, and hence the total running time is $O(m \log C)$.

Problem 2e. When we use base b, the number of scaling iterations is $\log_b C$. In this case we set up:

$$x_{vw}^i = l_{vw}^i + bd_v^{i-1} - bd_w^{i-1}$$

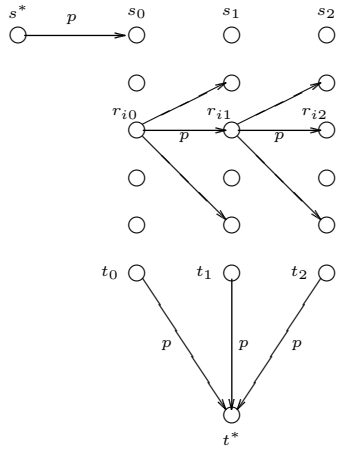
And derive $d_v^i = r_v^i + bd_v^{i-1}$. Hence the length of the longest shortest path becomes bn, and the total running time $O((m+bn)\log_b C)$. We find the optimum $b = m/n$, to get total running time $O(m \log_{m/n} C)$.



Problem 3a. Define a graph H_T, corresponding to T time steps, with vertices s^*, t^* and $r_{i,j}$ for all rooms r_i and times $0 \leqq j \leqq T$, where also $r_0 = s$ and $r_1 = t$. Also let the total number of people be p. Edges:

1. $(s^* s_0)$ of capacity p: Puts all people in the starting room initially
2. $(t_i t^*)$ of capacity p: Release any people who have reached the target room
3. $(r_{i,j}, r_{i,j+1})$ of capacity p: Allow people to remain in a room from one time step to the next
4. For every edge (r_u, r_w) of capacity c_{uw} in the original graph, place an edge $(r_{u,j}, r_{w,j+1})$ of capacity c_{uw} in the timed graph: Allowing the appropriate number of people to traverse that corridor between time steps

An illustration of H_T is given below:



H_t has $O(Tm)$ edges and $O(Tn)$ vertices, where m and n are the edges and vertices in the original graph.

A flow on H_T corresponds to a schedule for letting people outside of the building, and vice-versa (there is a 1-to-1 correspondance). This is so because each edge represents an event that can be true independantly from the remaining edges. In particular, two edges at the same time step represent independent events because they correspond to different corridors, and two edges across times steps are independent because they refer to different times.

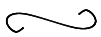
In particular, a p-flow on H_T corresponds to a schedule for sending all people to lunch in at most T time steps.

Our goal is to find the smallest T for which a p-flow exists. This can be done with a binary search on T. Start with $T = 1$. At step T_i run MAX FLOW, if no p-flow exists let $T_{i+1} = 2T_i$, otherwise stop and do a regular binary search between T_{i-1} and T_i. This procedure takes $O(\log T_{\min})$ iterations of MAX FLOW, each taking $O(T_{\min}^3 mn^2)$, for total of $O(mn^2 T_{\min}^3 \log T_{\min})$.

Problem 3b. Let the multiple entry locations were rooms $r_{i_1}, \ldots, r_{i_k}$ and their respective number of people be $a_1, \ldots, a_k$. Simply add edges $(s^*, r_{i_x,0})$ of capacities a_x in H_T for $1 \leqq x \leqq k$.

Let the multiple exit locations be rooms $r_{j_1}, \ldots, r_{j_l}$. Simply add edges $(r_{j_x,z}, t^*)$ of capacity p in H_T for $1 \leqq x \leqq l$ and $0 \leqq x \leqq T$.

Problem 3c. In this scenario, for a corridor (r_u, r_w) with transit time y, introduce edges $(r_{u,i}, r_{w,i+y})$ of capacity c_{uw} for all $i \leqq T - y$. Note that this model takes into account that a corridor of transit time y can have up to $c_{uw}y$ people in it at the same time, where only c_{uw} can enter on each time step.



Problem 4a. Begin by finding a feasible flow, simply using the technique described in class:

- Add new source s^* and new sink t^*
- Replace each edge ij with an edge of capacity $u_{ij} - l_{ij}$
- Add demand edge it^* of capacity l_{ij}
- Add supply edge s^*j of capacity l_{ij}
- Add infinite capacity edge ts (old sink to old source)
- Run MAX FLOW on resulting graph
- This produces a feasible flow f on the original graph

At this point we have our original graph G together with a feasible flow. The next step will produce the minimal flow:

- Create residual graph G_f of G with the feasible flow f
- Subtract u_{ij} capacity from the backwards ji edges of G_f so that we cannot undo the minimum allowable flow on the edges
- Run MAX FLOW on G_f in the opposite direction, i.e. from t to s
- Add the resulting flow g to the initial feasible flow f to get the minimum feasible flow h

The resulting flow is feasible by construction. It is minimal, because if there were a smaller flow, we would have missed an augmenting path which is a contradiction.

Problem 4b. First we show the inequality. For any feasible flow f and cut S: the lower bound on the cut capacity (LBCC) is $\sum_{ij \in S \times T} l_{ij} - \sum_{ji \in T \times S} u_{ji}$. We know that $\sum_{ij \in S \times T} l_{ij}$ is a lower bound on how much f pushes forward through the cut, and $\sum_{ji \in T \times S} u_{ji}$ is an upper bound on how much f pushes backwards through the cut, hence the LBCC is a lower bound of f.

We now show that f is equal to the LBCC of some cut (which would imply that this is the maximum LBCC due to the inequality above).

Start by specifying the edges of the graph G' on which we run the second MAX FLOW procedure. For every edge ij in G, there are two edges in G': (1) ij with capacity $u_{ij} - l_{ij} - \Delta f_{ij}$, and (2) ji with capacity Δf_{ij}, where $\Delta f_{ij} = f_{ij} - l_{ij}$.

Consider the cut defined by all vertices reachable from t in the residual graph of G' after the second run of MAX FLOW (which was in the opposite direction). s is not a member of this set, since there is no augmenting path from t to s.

For every edge ij in G, the residual of G' must have no edge ji which means that it saturated the ji edge of G' of capacity Δf_{ij}. So g_{ij} (with respect to G) is $-\Delta f_{ij}$ (in the s to t direction). On the other hand $f_{ij} = l_{ij} + \Delta f_{ij}$, hence $h_{ij} = f_{ij} + g_{ij} = l_{ij}$.

For every edge ji in G, the residual of G' must have no edge ji which means that it saturated the ji edge of G' of capacity $u_{ji} - l_{ji} - \Delta f_{ji}$. So g_{ji} (with respect to G) is $-(u_{ji} - l_{ji} - \Delta f_{ji})$ (in the s to t direction). On the other hand, $f_{ji} = -(l_{ji} + \Delta f_{ji})$, hence $h_{ji} = f_{ji} + g_{ji} = -u_{ij}$.

Summing over all edges that cross the cut, we get that the flow h is exactly equal to the LBCC of the cut.

Problem 4c. Set up a graph G with vertices $s, t, a_1, b_1, \ldots, a_n, b_n$, and edges:

1. (s, a_i) of capacity 1 and lower limit 0, for all i
2. (b_i, t) of capacity 1 and lower limit 0, for all i
3. (a_i, b_i) of capacity 1 and lower limit 1, for all i
4. (b_i, a_j) of capacity 1 and lower limit 0, for all $a_j - b_i \geqq r_{ij}$

Run MIN FLOW on this graph. Each flow path (of the flow decomposition) corresponds to a student's schedule. (There are no cycles as the graph is acyclic.)

Edges of types 3 and 4 correspond to actaul work being done (attending lecture or commuting between lectures). Edges of type 1 correspond to requiring a new student to enter into the schedule. Edges of type 2 correspond to a student being done with their schedule.

All edges have upper capacities of 1 because it is easily seen that at no point it makes sense to have more than 2 students with overlapping schedules (as one of them could be assigned a different schedule, or simply released from the loop).