CSE 202: Design and Analysis of Algorithms Winter 2013

Problem Set 3

Instructor: Kamalika Chaudhuri **Due on:** Tue. Feb 26, 2013

Instructions

- For your proofs, you may use any lower bound, algorithm or data structure from the text or in class, and their correctness and analysis, but please cite the result that you use.
- Problem 1 is worth 18 points. Problem 2 is worth 12 points.
- If you do not prove that your algorithm is correct, we will assume that it is incorrect. If you do not provide an analysis of the running time, we will assume you do not know what the running time is.

Problem 1

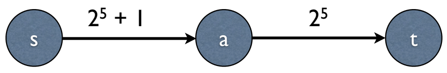
The following statements may or may not be correct. For each statement, if it is correct, provide a short proof of correctness. If it is incorrect, provide a counter-example to show that it is incorrect.

1. In the capacity-scaling algorithm, an edge e can be present in the residual graph $G_f(D)$ in at most one D-scaling phase.
2. The max-flow problem with node capacities is defined as follows. We are given a directed graph $G = (V, E)$ with source s, sink t and capacities $c_v \geq 0$ for all nodes $v \in V$ (but no edge capacities c_e). A flow f is feasible in G if for all v other than s and t, the total flow into node v is at most c_v, and the size of a feasible flow f is the total flow out of s. Given this input, the max-flow problem with node capacities is to compute a feasible flow of maximum size.

   Computing a max-flow with node capacities can be reduced to the regular max-flow problem.
3. Let f and h be a preflow and a labeling compatible with the preflow. Let (u, v) be an edge in the original graph G, and let $h(v) = 4$ and $h(u) = 2$. Then, $f(u, v) = 0$.
4. Suppose you have already computed a maximum (s, t)-flow f in a flow network G with integer capacities. Let k be an arbitrary positive integer, and let e be an arbitrary edge in G whose capacity is at least k. Now suppose we increase the capacity of e by k units. Then, the maximum flow in the updated graph can be computed in $O(|E|k)$ time.
5. If all directed edges in a network have distinct capacities, then there is a unique maximum flow.
6. If we replace each directed edge in a network with two directed edges in opposite directions with the same capacity and connecting the same vertices, then the value of the maximum flow remains unchanged.

Solution

1. False. In the following graph G, the (s, a) edge is present in two D-scaling phases – $D = 2^5$ and $D = 1$.



2. True. The reduction is as follows. We build a graph G', which, for each node v in G with node capacity c_v, has two nodes two nodes v_1 and v_2. All incoming edges to v connect to v_1, and all outgoing edges from v connect from v_2. Finally, add the edge (v_1, v_2) with edge capacity c_v.

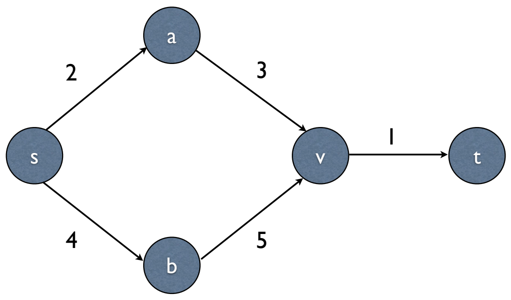
   Now any flow f in G which respects the node capacity constraints can be converted to a flow f' in G' which also respects the edge capacity constraints. To convert f to f', set $f'(u_2, v_1) = f(u, v)$ for any edge $(u, v) \in G$, and $f'(v_1, v_2)$ to be the total incoming flow at node v. As the node capacity constraint on v is satisfied in G, $f'(v_1, v_2) \leq c_v$, and hence f' satisfies the edge capacity constraints in G'. f' also satisfies conservation constraints because f satisfies conservation constraints and because of the way we set $f'(v_1, v_2)$.

   Moreover, any flow f' in G' can be converted to a flow f in G that satisfies the node capacity constraints. To convert f' to f, set $f(u, v) = f'(u_2, v_1)$ for any edge $(u, v) \in G$. f satisfies node capacity constraints because f' satisfies capacity and conservation constraints; this ensures that the total flow through any node v is at most c_v. Finally f satisfies conservation constraints because f' satisfies conservation constraints too. Thus the max flow in G' corresponds to a flow in G of maximum size that respects the node capacity constraints.

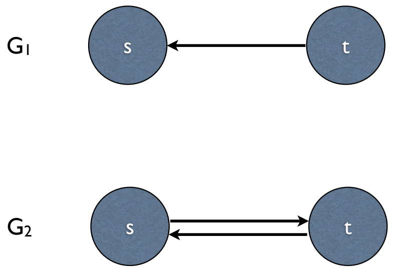
3. True. Proof is by contradiction. If $f(u, v) \neq 0$, then (v, u) is in the residual graph G_f. Note that $h(v) > h(u) + 1$; this violates the compatibility property for the edge $(v, u) \in G_f$.

4. True. Simply compute the residual graph and run the hill-climbing Ford-Fulkerson algorithm from that point. The min $s - t$ cut capacity increases by at most k. Thus the max flow size will increase by at most k. As each iteration of Ford Fulkerson takes $O(|E|)$ time, and each iteration improves the max flow by at least 1, the total running time will be $O(k|E|)$.

5. False. In the following graph G, all the edge capacities are unique; yet there are at least two max-flows f_1 and f_2 of size 1. Here $f_1(s, a) = f_1(a, v) = f_1(v, t) = 1$, and $f_1(e) = 0$ for all other edges e; $f_2(s, b) = f_2(b, v) = f_2(v, t) = 1$, and $f_2(e) = 0$ for all other edges e.



6. False. Assume that edge capacities are 1. In G_1, the max-flow has size 0, whereas in G_2, the size of the max-flow is 1.



Problem 2

In a public building such as a movie theater, it is important to have a plan of exit in the event of a fire. We will design such an emergency exit plan in this question using max-flows. Suppose a movie theater is represented by a graph $G = (V, E)$, where each room, landing, or other location is represented by a vertex and each corridor or stairway is represented by an edge. Each corridor has an associated capacity c, meaning that at most c people can pass through the corridor at once. Traversing a corridor from one end to the other takes one timestep. (Traversing a room takes zero time.)

1. Suppose all people are initially in a single room s, and there is a single exit t. Show how to use maximum flow to find a fastest way to get everyone out of the building. (Hint: create another graph G' that has vertices to represent each room at each time step.)

2. Show how the same idea can be used when people are initially in multiple locations and there are multiple exits.

3. Finally, suppose that it takes different (but integer) amounts of time to cross different corridors or stairways, and that for each such corridor or stairway e, you are also given an integer $t(e)$ which is the number of seconds required to cross e. Now show how to transform your algorithm in Part (1) to find a fastest way to get everyone out of the building.

Solution

1. Suppose that there are M people that need to be moved out. First, we provide an algorithm to decide if all people can be moved out in T steps. Given this algorithm, we can do a binary search on T between 1 to $|V|M/c$ to find the shortest time in which all the people can move out.

   Our algorithm is as follows: given the graph G, we construct G_T as follows. For each $v \in V$, we make T copies of v: $v_1, \dots, v_T$, where copy v_i corresponds to time step i. For each i, we 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). We then 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. Suppose everyone is in room a initially, and the exit is room b. Then we set the source $s = a_1$, and the sink $t = b_T$.

   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 greater than or equal to the number of people initially at the source. If so, we can move all the people across this graph in T timesteps.

2. We can use the same overall idea: construct a graph G_T, and compute its max flow. If its max flow is greater than or 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 corresponding to the rooms that initially contain all the people, and let U be the sink vertices that correspond to all the exits. 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 $x \in U$) to T with infinite capacities.

3. 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 $w_i + t(v, w)$ with capacity c if there is an edge between v and w in G with transit time $t(v, w)$.

   Correctness. Suppose the number of steps output by the algorithm is T, and initially there are M people. To show the correctness, we need to show the following two statements. First, if there is a way to get everyone out of the building in T steps, then, there exists a flow of size M in G_T. Second, if there exists a flow of size M in G_T, then, there is a way to get everyone out of the building in T steps. We prove these statements for part (a) of the question below; the proofs for parts (b) and (c) are analogous.

   For the first part, consider a schedule for getting everyone out of the building in time T. We show that this schedule can be converted to a valid flow f of size M in G_T. Given the schedule, we build f as follows: if m people are sent through the corridor (v, w) at time i, set $f((v_i, w_{i+1})) = m$. If m' people linger at room v at time i, set $f((v_i, v_{i+1})) = m'$. Set $f((u, v)) = 0$ for any other edge $(u, v) \in G_T$. Since no more than c people can go through a corridor at any time, f satisfies all capacity constraints. Moreover, because the schedule ensures that anyone who enters an intermediate room v other than s and t ultimately leaves it, and because of the way we set $f((v_i, v_{i+1}))$, f also satisfies conservation constraints. Finally, the size of f is the number of people ultimately moving out of room a, which is M. Thus if a valid schedule exists, then, G_T has a max flow of size M.

   For the second part, consider a flow of size M in G_T, and convert it to a schedule as follows: send $f((v_i, w_{i+1}))$ people from room v to room w through the corridor connecting these rooms at time i. This is a valid schedule by construction of G': all people start from the source, and end up at the exit, and people who enter corridor (v, w) at time i leave at time $i + 1$. Moreover, because f satisfies capacity constraints, no corridor has more than c people at any time; because f satisfies conservation constraints, people who enter a room must ultimately leave the room. Finally, this schedule gets everyone out of the building in time T, because everyone is at the exit at time T. Thus, if there is a flow of size M in G_T, then there is a schedule to get everyone out of the building in time T.

   The correctness proof for parts (b) and (c) is analogous.

Running Time. Suppose the movie theater has n rooms and m corridors. Then for a given T, the constructed G_T has $O(Tn)$ nodes and $O(T(m+n))$ edges, regardless of which version of the problem is being solved. The maximum flow value is M and the reduction will work with a maximum edge capacity of $O(M)$. The running time for the max flow problem in iteration T is thus $O(\min((m+n)TM, (m+n)^2T^2 \log M, (m+n)n^2T^3))$, depending on the relative values of M, T, m and n.

We can find the optimal value of T by doing a binary search – first we double t until we can get everyone out of the building in t steps; if the first such value of t we encounter is t^*, then we can do a binary search on the interval $[t^*/2, t^*]$ to get to the correct value of T. This procedure will involve $O(\log T)$ invocations of max flow, each invoked for a $t \leq 2T$. Thus the total running time is:

$$O\left(\log T \cdot \min((m+n)TM, (m+n)^2T^2 \log M, (m+n)n^2T^3)\right)$$