CS 161 Fall 2017: Section 9 Solutions

Max Flow Potpurri

(a) Suppose that instead of having a single source and sink s, t respectively, we have multiple sources $S = \{s_1, s_2 \dots s_k\}$ and multiple sinks $T = \{t_1, t_2 \dots t_l\}$. We wish to still find the max flow in the graph from sources to sinks.

Create a meta source node s' connected to all source nodes with edge weight ∞. Likewise, create a meta sink node t' where all sink nodes are connected to t' with weight ∞.

(b) Suppose that in addition to edges having max flow capacities, vertices also have a limit to their capacity; that is, each vertex v_i has capacity c_i. We wish to find the max flow from a source s to sink t in this graph.

Replace each vertex v_i with v_i and v_i', where there is an edge $v_i \to v_i'$ with weight c_i.

(c) Given a solution to max-flow, verify that it is correct in linear time.

Construct the residual graph and try to find an augmenting path.

Expense Settling

You've gone on a trip with k friends, where friend i paid c_i for the group's expenses. You would like to develop an algorithm to ensure that everyone gets paid back fairly, but without going through one person (that is, each person would either pay or receive money, but not both).

Calculate the per person cost, $c = \frac{\sum c_i}{k}$. People who paid more than c need to get paid back, while people who paid less need to pay others. Create a graph with a source node s, sink node t, and one node per person v_i. If $c_i > c$, this person needs to get paid back, and we draw an edge from $v_i \to t$ with weight $c_i - c$. If $c_i < c$, this person needs to pay other people, and we draw an edge from $s \to v_i$ with weight $c - c_i$. We connect all pairs of vertices $v_i \to v_j$ with edge weight ∞. We find the max flow from source to sink in the graph, and the flow along an edge will represent how much people pay one another.

Project Selection

Suppose you have a set of k tasks $t_1 \dots t_k$. There are certain tasks such that t_i is a prerequisite of t_j. Each task also has a reward r_i, which may be negative. Find an optimal subset of tasks to complete to maximize your reward.

Draw an edge from $v_j \to v_i$ with weight ∞ if t_i is a prereq of t_j. If $r_i > 0$, add an edge from $s \to v_i$ with weight r_i. If $r_i < 0$, add an edge from $v_i \to t$ with weight $-r_i$. The min cut containing s is the optimal set of projects.