CS 124 Section #10 **Network Flows** 4/12/15

1 Network Flow

Recall the following useful facts:

- Any cut is at least as large as every flow
- There is some flow which has at least the value of some cut (via Ford-Fulkerson)
- Therefore, the max-cut and min-flow are equal
- If all capacities are integers, then there is an integer solution to max-flow (also via Ford-Fulkerson)

2 Easier Problems

Exercise. *How might you handle the maximum flow problem with multiple sources and/or sinks?*

Solution.
Suppose the sources are $s_1, ..., s_k$ and the sinks are $t_1, ..., t_\ell$. Add a super-source s with an edge into every source s_i with capacity ∞, and add a super-sink t with an edge from every sink t_j with capacity ∞. Find the maximum flow in this new graph.

Exercise. *How might you hangle the maximum flow problem if the vertices also have capacities?*

Solution.
Replace every vertex v with two vertices v_{in} and v_{out}. Connect them with an edge with the capacity of vertex v. Connect all edges into v into v_{in}, and connect all edges out of v out of v_{out}. Find the maximum flow in this new graph.

Note that there are $O(n)$ vertices and $O(n+m)$ edges in the new graphs for both exercises.

In the following 2 exercises, suppose we are given a maximum flow in a graph $G = (V, E)$ with source s, sink t, and integer capacities. (That is, we're given both the maximum flow value, as well as the amount of flow that goes along each edge to achieve that value.)

Exercise. *Now the capacity of a given edge e is increased by 1. Give a linear time $O(|V| + |E|)$ algorithm for computing the new maximum flow.*

Solution.
If we incremented the capacity of edge e by 1, there are two possible effects on the overall flow of the graph: increase by 1 or stay the same. To determine what effect, we must determine whether or not there is a path

in the residual graph that would allow an additional 1 unit of flow to pass through the network, given the changed capacity of e. To find find the residual graph, we can subtract the amount of flow sent through at the previous optimal by the capacities of the edges, creating the residual graph. We can therefore run a depth-first search (DFS) on the graph, from the source s to the sink t. We stipulate tha the DFS must only traverse along edges whose weight in the residual graph is strictly graters than 0. That is, additional flow is able to be sent down these edges.

Note that if a path from s to t such that each edge in the residual graph exists, it must pass through e; otherwise the optimal solution would have sent flow down that path when calculating the original optimal solution.

If we are able to find a path using our DFS that fulfills the conditions outlined above, we can increment the maximum flow over the graph by 1. If we want to find the updated flows along each of the edges, we could also keep track of which edges we are traversing in our DFS. Otherwise, if we could not find such a path, we do not change the maximum flow.

Exercise. *Similarly, give a linear time $O(|V| + |E|)$ algorithm for computing the new maximum flow if the capacity of a given edge e is decreased by 1.*

Solution.
Let G' be an undirected, unweighted graph whose edges are the edges with non-zero flow on them in the given max flow f. Do a DFS from s and another DFS from v in G'. In this way we can find a path p_1 from s to u and p_2 from v to t used by the flow f. Thus the concatenated path $p_1 \circ (u, v) \circ p_2$ is a path used by the max flow f. Decrease the flow in f on all edges in this path by 1 to form a new flow f' with one less total flow value. Note f' obeys the capacity constraints (since in particular we decremented one unit of flow on the edge (u, v)). Now form the residual graph for f' and try to find an augmenting path.

3 Trickier Problems

Exercise. *Suppose you are given a $u \times v$ matrix A with real entries. Find an efficient algorithm to either construct a $u \times v$ matrix B with integer entries such that the sum of the entries in each row/column of A is equal to the sum of the entries of each row/column of B, or output that this is impossible.*

Solution.
If any of the row/column sums for A are not integers, this is clearly impossible. This may be checked in $O(uv)$ time. Otherwise, we claim this is always possible.

Firstly, we may reduce to the case when entries in A are in $[0, 1]$ (subtracting/adding an integer to any particular entry doesn't change the property that the row/column sums are integers). This takes time $O(uv)$.

Construct a graph G, which has a source vertex s, a sink vertex t, vertices $a_1, ..., a_u$ corresponding to each row, and vertices $b_1, ..., b_v$ corresponding to each column. Connect s to each a_i with the corresponding row sum, and connect each b_j to t with the corresponding column sum. Connect each a_i to every b_j with capacity 1.

Note that the flow that saturates all edges out of s and into t, and sends flow A_{ij} from edge a_i to b_j is a max-flow. Indeed, its value is the sum of the entries in the matrix A, which is the same value as the cut that separates s from the remainder of the graph (and the cut that separates t from the remainder of the graph).

Run Edmonds-Karp on this graph. Since the capacities are integers, this will result in an integer max-flow. If there is a flow of 1 from a_i to b_j, set $B_{ij} = 1$ and otherwise set $B_{ij} = 0$. Since this is a maximum flow, it saturates the edges out of s and into t (these cuts have the same value as the max-flow by the above paragraph). This implies B has the same row and column sums as A.

Our graph has $O(u + v)$ vertices and $O(uv)$ edges, so running Edmonds-Karp takes time $O(u^2v^2(u + v))$.

Exercise. *A standard deck of 52 cards is split into 13 piles of four cards each. Show that no matter how the deck is split, it is possible to select one card from each pile in such a way that the thirteen cards selected contain all thirteen ranks.*

Solution.
Construct a graph G which has a source vertex s, a sink vertex t, one vertex a_i for each rank i, and one vertex b_j for each pile j. Connect s to every a_i with capacity 1, and connect evert b_j to t with capacity 1. If there is a card of rank i in pile j, connect a_i to b_j with capacity 1000000.

Suppose we can find an integer max-flow of value at least 13. Then this flow saturates all edges out of s, so every vertex a_i must send flow exactly 1 to some vertex b_j. Pick rank i from pile j. This flow also saturates all edges into t, so every vertex b_j receives flow exactly 1, and thus this corresponds to a valid selection.

To show this max-flow exists, it suffices to show that the min-cut has value at least 13. Suppose a min-cut splits the graph into parts $\{s\} \cup U \cup V$ and $\{t\} \cup W \cup X$, where $U, W \subseteq \{a_1, ..., a_{13}\}$ and $V, W \subseteq \{b_1, ..., b_{13}\}$. It's clear that for this to be a min-cut, all edges crossing it must be out of s or into t (the other capacities are far too big!). This implies the min-cut has size $|W| + |V|$. This also implies that our graph has no edges from U to X - hence every edge out of U must be into V. In particular, $|V| \geq |U|$ - since all cards with ranks in U belong to piles in V. The $|V|$ piles must have at least $|V|$ distinct ranks, so there need to be connections to at least $|V|$ nodes in U, which means that $|V| \geq |U|$.

Putting it all together, out min-cut has size $|W| + |V| \geq |W| + |U| = |\{a_1, ..., a_{13}\}| = 13$, as desired.