Stefanie Shidoosh

CS 180: Homework #7

1. Develop an algorithm to solve the water flow problem using Siddarth's solution and the Union-Find algorithm in O(nlogn) time.

Each break point is a tree consisting of itself, initially
Insert the first interval of maxflow to a rooted tree that includes all break points between *i* and *j*
    Union all break points between *i* and *j*
Recursively follow the breakpoint of each next interval in the order of flow to its respective root
    The root determines the specified interval the respective breakpoint belongs to
    Union the resulting tree to the preexisting one, such that the structure satisfies the properties needed for *O(logn)* time for Union-Find algorithm:
        The root of the smaller tree to point to the larger tree; this maintenance allows for traversal of the tree to any node is less than log(2n) steps.
Once completed for all breakpoints i through j, the resulting tree will have the correct intervals that can now be used to produce the graph
Sort the nodes of the tree in increasing order with respect to the begin time of the interval, and the result will be the breakpoints to produce desired graph for maximum water flow for each interval

The total time complexity of this algorithm is $O(nlogn) + O(log(2n))$, which results in an overall time complexity of $O(nlogn)$ with sorting as the most significant operation.

Stefanie Shidoosh

2. In an undirected graph $G = (V, E)$, the edge s-t connectivity is the minimum number of edges to disconnect two vertices s and t. Using max flow, prove that the s-t connectivity equals the maximum number of edge disjoint paths between s and t. (Hint: replace G by creating parallel edges and assign edge capacities 1. This observation is called Menger Theorem and will help you copying from the book or the internet if you are so inclined)

Pick an arbitrary edge e_0 in E(G). Replace that edge with two parallel directed edges to construct a new graph G_0. Consider the max-flow min-cut theorem with respect to G_0 and assign a capacity of 1 to every edge. Let k be the edge connectivity of G. Suppose that by eliminating a set of k edges will generate the subgraphs of G_0, G_1 and G_2, such that one of the vertices, v_1, of e_0, will be in G_1 and another vertex, v_2, will be in G_2. Using the max-flow min-cut theorem again, removing a set of k edges that is applied for $d_0(v_1, v_2)$, and $c_0 \leq d_0(v_1, v_2) \leq k$. This is a contradiction as c_0 is not less than k minimum amount of edges to be removed, and therefore the minimum number of edges to disconnect two vertices s and t must be *equal to* the maximum number of edge disjoint paths between s and t.

Stefanie Shidoosh

3. The s-t vertex-connectivity of undirected graph $G = (V, E)$, is the minimum number of nodes in order to disconnect two vertices s and t. Show that the s-t vertex-connectivity equals the number of vertex-disjoint paths between s and t. (Hint: split a vertex into two vertices)

Suppose there exists n edge-disjoint s-t paths in G. To disconnect vertices s and t, at least one edge must be deleted from each of these n paths. Suppose we can disconnect s and t by removing k edges; we aim to show that there will be k vertex-disjoint paths between s and t to prove that s-t vertex-connectivity equals the number of vertex-disjoint paths. By Max-Flow Min-Cut theorem, k units of flow can flow from s to t. Starting at s, there will always be an outgoing saturated edge by conservation, when considering the maximum flow network. Continue following saturated edges until final destination t is visited, marking each vertex visited and removing cycles. Reducing the flow across the path to 0 to create a new flow, which will have the value $k - 1$. Therefore, by induction, s-t vertex-connectivity equals the number of vertex-disjoint paths between s and t can be concluded.

Stefanie Shidoosh

4. Suppose we are given the maximum flow in a flow network $G = (V, E)$ with source s, sink t, and integer capacities. Suppose the capacity of a single edge is increased by one. Give an $O(n + m)$ algorithm for updating the maximum flow, where G has n vertices and m edges.

Construct the residual graph on the original flow capacities
Assign a capacity of 1 in addition to the previous capacity to the edge that has been increased.
Use BFS to search for an augmenting path p
    If p exists, update the flow
    Else, flow is unchanged
This is done once as p exists and increases flow by the maximum increase, which is 1

This algorithm has a time complexity of $O(n + m)$ as this is the time is takes to construct the residual graph to find p.