1. Computing Blocking Flows

(a) Why? Edmonds-Karp algorithm: remove augmenting paths with smallest number of edges increases length of such path. $O(n)$ blocking flows give maxflow.

(b) Have a DAG, want to find a flow so there are no more paths going along direction of edges.

(c) Difference with maxflow: can still have augmenting paths that take edges backwards.

(d) $O(nm)$ time algorithm: find path, saturate edge with minimum remaining capacity on that edge.

(e) Faster algorithm due to Sleator and Tarjan: $O(m \log n)$ per blocking flow, $O(mn \log n)$ total. Reason for invention of dynamic trees.

(f) Idea: if vertex has path of non-saturated edges out of it, reuse it until no longer usable.

(g) Invariant: each vertex currently considering one edge out of it.

(h) DAG ensures we have a forest.

(i) Recall: dynamic tree, represent real tree as a series of virtual trees, each corresponding to a path.

(j) Always try to extend forest at root of current node.

(k) If reaches $t$, push flow equaling to minimum remaining capacity.

(l) Need to decrease remaining capacities on all edges from $s$ to $t$ (which is at root) in the real tree.

(m) Operations needed on virtual trees: findmin, decrease range.

(n) Flow Decomposition: figure out which edge leaving $s$ goes to which edge entering $t$. Can be done by finding blocking flow in flow graph.