

Network Optimization (Fall 2008) — Brief Solutions to the Final Exam

- Let S_r be the set of nodes i with $d(i) = r$. Then the arcs $(i, j) \in A$ with $i \in S_r$ and $j \in S_{r+1}$ will all be forward arcs in an s - t cut for each $0 \leq r < d(t)$, assuming $d(t) \geq 1$. Let $|S_r| = n_r$ (the number of nodes at distance r from node s). Since all arcs have unit capacity, the number of forward arcs in this cut, which is equal to the capacity of the cut, has to be at least v^* , the maximum flow value. On the other hand, there can be at most $n_r \times n_{r+1}$ forward arcs in this cut. Hence, we must have

$$n_r \times n_{r+1} \geq v^*, \quad \text{which gives } \min\{n_r, n_{r+1}\} \geq \sqrt{v^*}.$$

It follows that for every *other* r from among $1, \dots, n_{d(t)}$, we must have $n_r \geq \sqrt{v^*}$. Adding the number of nodes in the sets S_r for every other r gives a lower bound on the total number of nodes n . Hence,

$$n \geq (d(t)/2) \sqrt{v^*}, \quad \text{which gives the desired inequality.}$$

- Find a minimum s - t cut $[S, \bar{S}]$. The capacity of this cut is equal to the values of both the maximum flows \mathbf{x}' and \mathbf{x}'' . It must be the case that the two maximum flows coincide on the forward arcs in this min cut (we assume all $l_{ij} = 0$ here, and hence all forward arcs in the minimum cut are saturated in *any* maximum flow). As such, we can create a new maximum flow \mathbf{x} by setting $x_{ij} = x'_{ij}$ for all arcs (i, j) with $i, j \in S$, $x_{ij} = x''_{ij}$ for all arcs (i, j) with $i, j \in \bar{S}$, and $x_{ij} = x'_{ij} = x''_{ij}$ for all arcs $(i, j) \in [S, \bar{S}]$.
- We first prove a *dual* version of the optimality conditions for minimum spanning trees (MST) of a connected graph G .

Lemma 1. *Let H be a subgraph of the network G containing an MST of G . Then $H \setminus \{e\}$ also contains an MST of G if and only if H contains a cycle W such that e is a costliest edge in W .*

Proof: (\Rightarrow) Let T be an MST of G contained in $H \setminus \{e\}$. Consider the cycle W obtained by joining the unique path P in T between the nodes of e and the edge e itself. It must be the case that $c(e) \geq c(e') \forall e' \in P$, else we can replace e' with e to obtain a spanning tree T' with lower total cost than T . Here, $c(e)$ is the cost of the edge e .

(\Leftarrow) Let T be an MST of G contained in H . The case when $e \notin T$ is trivial, so we assume $e \in T$. Consider the cut obtained by removing e from T . The cycle W must contain at least one other edge $e' \in T$ in this cut, and $c(e) \geq c(e')$ by assumption. Then we can get another MST T' by replacing e with e' , and T' will be contained in $H \setminus \{e\}$. \square

The given algorithm essentially builds an MST by satisfying these dual optimality conditions. The algorithm obviously returns a spanning tree, say T . To be correct, it must consider every edge, though (i.e., the algorithm must not stop even if it has a spanning tree before it has considered adding every edge). Hence, any edge $e \notin T$ has been

excluded because it was the costliest edge in some cycle of the original graph. The remaining subgraph (of the original graph) will contain an MST by the dual optimality conditions.

The default version of Kruskal's algorithm runs in $O(mn)$ time, with $O(m \log m)$ time for sorting the arc capacities in the beginning. Even though we save the effort to sort all the arc costs, we have to spend more time to find the costliest arc in each cycle identified. Using default methods, this step will take $O(m)$ effort for each arc, and hence achieve an overall running time of $O(m^2)$. Using better data structures, we can achieve a worst-case running time for this algorithm that is a factor of $\log n$ more than that for the improved implementation of Kruskal's algorithm.

4. Consider the residual network G' obtained from $G(\mathbf{x}^*)$ by increasing u_{ij} by one unit. Apply the cycle cancelling algorithm (or steps of any other minimum cost flow algorithm) to the modified residual network. Notice that \mathbf{x}^* will be a feasible solution to the new network as well, and hence will give an upper bound to the objective function value. If there are no negative cycles in G' , then \mathbf{x}^* is optimal for G' as well, and hence (i, j) is not upward critical. If there is a negative cycle in G' , then we can get another solution \mathbf{x}' for G' by augmenting along this cycle, to obtain a strictly smaller total cost than \mathbf{x}^* . In this case, (i, j) is upward critical.
5. Both statements can be shown to be TRUE.
 - (a) Let L and L' be the sorted lists of costs of all arcs in the MSTs T and T' of G , respectively. We argue that $L = L'$. Let the lists of costs be $L = [c_1, c_2, \dots, c_{n-1}]$ and $L' = [c'_1, c'_2, \dots, c'_{n-1}]$. Also, let the lists of the corresponding edges be $E = [e_1, e_2, \dots, e_{n-1}]$ and $E' = [e'_1, e'_2, \dots, e'_{n-1}]$, respectively. Let i be the smallest index such that $e_i \neq e'_i$. WLOG, assume $c_i \leq c'_i$. Consider adding e_i to T' . In the trivial case where we still get a tree, $e_i \in T'$, and we could just reorder the edge list L' such that $e_i = e'_i$. Hence, let us assume that $e_i \notin T'$. In this case, there will be a unique path P connecting the end points of e_i in T' . Consider all edges in P that are in T' , but not in T . There must be at least one such edge, else we get a cycle in T . By our assumption about the order of edges, all these edges must have costs $\geq c'_i$, as edges with smaller costs are common to both trees. Consider the arc from this set with the smallest cost, say e'' . By path optimality conditions for T' , this arc should have cost $c'' \leq c_i$. Hence we should have $c'' = c_i$, and we can substitute e_i for e'' in T' to get a different tree with the *same* sorted list of costs L . We can continue this substitution process until we get $L' = L$. In other words, any two MSTs of a graph G will have the same list of sorted costs.
 - (b) We start with some feasible flow, which could be found by solving a maximum flow problem. Since all $b(i)$'s are even by assumption, all the arc capacities in the network created for this maximum flow problem will also be even. Hence the feasible flow found will also be even, as each augmenting path will have an even capacity. Subsequently, each residual capacity will also be even, and hence every negative cycle identified in the residual network will have an even (bottleneck) capacity. Each subsequent augmentation will maintain the residual capacities as even numbers.