6.854 Advanced Algorithms

Lecture 10: 09/29/2006 Lecturer: David Karger

Scribes: Huy Nguyen, Edwin Chen and Lyric Doshi

10.1 Summary of last lecture

Max flow

1. Augmenting paths
2. O(mf) = O(mn) on simple unit capacity graph
3. $O(m^2U)$ for max capacity U – “pseudopolynomial”
4. Maximum capacity augmenting path
5. Scaling $O(m^2 logU)$ – “weakly polynomial”

10.2 Shortest Augmenting Path – Edmonds-Karp

Intuition: In residual graph, if the source and the sink are very far apart, then not much flow is left in the residual graph. Intuitively, it is more likely that there exists a small-valued cut separating s and t.

How might we do our augmenting path search to make the source and sink far apart?

Because we don't want short paths, we try to get rid of them by augmenting along short paths. Then only long augmenting paths remain.

Unlike flow whose values are arbitrarily large numbers, augmenting path lengths are only from 1 to n so they are independent of the actual value of the flow.

To guarantee this approach will terminate, we need a notion of progress.

Lemma 1 *Under shortest augmenting path, d(s,i) and d(i,t) (d=distance measured by the number of edges) in the residual graph are non decreasing for any i.*

Proof: We will use contradiction. Supppose that among the vertices that got closer to s, consider the one closest to s after the change, i. After the change, there's a new shortest path from s to i. i has a predecessor j. j did **not** get closer. i can only get closer if the edge (j,i) is new. But this can happen if and only if the augmenting path sent flow from i to j. This implies i was closer than j to s before the augmentation and it got even closer to s after. This is a contradiction because we assume that j is the current predecessor of i. ■

Lemma 2 *There are at most $\frac{mn}{2}$ augmentations.*

Proof: Consider edge (i, j) which is saturated by given shortest augmenting path. Before the next use of (i, j) to augment the residual graph, we must augment over (j, i). When we augment from i to j, j is farther than i from the source. Similarly, when we augment from j to i, i is farther j from the source. By the previous lemma, d(s, i) and d(s, j) are non-decreasing. Therefore, d(s, i) must have increased by at least 2. That can happen at most n/2 times. Once d(s,i) ¿ n, no augmenting path can reach i. Because there are m edges, we conclude there will be $\frac{mn}{2}$ augmentations overall. ■

Corollary 1 *Runtime for shortest augmenting path is $O(m^2n)$ – "strongly polynomial"*

Insight: Improve by making your work count.

How to take advantage of doing a full BFS? Can we do more than one augmenting path with one BFS? When we do BFS, we get a bunch of layers in the graph. We could find multiple paths through the layers of the graph and augment along many of them at once.

10.3 Blocking Flows - Dinic

Definition 1 ***layered graph:*** *the graph divided into layers by distances from the sink.*

Definition 2 ***admissible arc:*** *an arc that goes from the layer at distance d to the layer at distance d - 1.*

Definition 3 ***admissible path:*** *path made of admissible arcs.*

Find a blocking flow that saturates at least one arc on every admissible path.

Claim 1 *Each blocking flow increases the distance between the source and the sink.*

Proof: Any remaining path must use a nonadmissible edge (this includes any new edges, which must go backwards) so every remaining path is longer than current shortest path. ■

Corollary 2 *n blocking flows yield a max flow.*

10.4 Variations on maxflow

1. Multiple sources and sinks.

   Add a super source with infinite capacity edges to all the original sources. Likewise, add a super sink with infinite capacity edges coming in from all the original sinks.

2. Feasibility of specific supplies s_i at sources and demands d_i at sinks.

   Same idea as last example: At the source side, add edges from the super source with capacities s_i, and at the sink side, add edges from the super sink with capacities d_i. We have thus converted a feasibility problem to an optimization problem. If this is a feasible problem, the maximum flow must be equal to the sum of s_i, and it will fill all the new edges.

3. Bipartite Matching

   We are given M men and W women, and a list of compatible pairs. The problem is to make as many simultaneous matches as possible, where each man and woman is matched at most once.

4. Perfect Matching

   Same problem, but matching all the vertices: Add a source and a sink. Then connect the source to all the men with unit capacity edges, and connect all the women to the sink with unit capacity edges.

10.5 Unit Capacity Blocking Flows

Blocking flows is easier than Max flow in the sense that when we augment along an edge, we don't have to worry about the newly introduced edge because we know it is not admissible.

We define the following primitive operations :

1. **Advance:** follow some edge forward from the current vertex, adding the traversed edge to the current path from the source.

2. **Retreat:** there is no outgoing edge from the current vertex, so traverse backward the last edge on the current path.

   Once a vertex is blocked, it stays that way, so delete an edge as we traverse it backwards.

Eventually a combination of advances and retreats brings us to the sink, and the current path becomes an augmenting path. Augment it. All the edges on the path get used up, so we can delete all edges on the path.

We delete each edge at most once. We advance once per edge because after we advance, we either reach the sink or retreat, and we delete the edge in either case.

Hence, we have $O(m)$ time for blocking flow, and $O(mn)$ max flow for a unit capacity graph.

This argument breaks if we have a capacitated graph, since we no longer augment along an edge once. Instead, when we augment, only one edge gets destroyed. Therefore we get a bound of $O(m^2)$ for blocking flow and consequently $O(m^2n)$ for max flow.