CSE 347 Analysis of Algorithms February 11, 2019

Lecture 8 Summary

Instructor: Brendan Juba

1. *Reductions* convert new problems into problems we've previously solved. We'll see how to use reductions to sophisticated algorithms to solve new problems in the next part of the course.
2. Example "sophisticated" problem: maximum flow (from s to t)
   (a) Given a directed graph G with integer weights ("capacities") $c(e)$ on the edges $e \in E$ and start and destination vertices s and t
   (b) A *flow* f in G is an assignment of nonnegative values to the edges such that $f(e)$ is at most the capacity of e and for all vertices except s and t, the total assignment to edges entering is equal to the assignment exiting that vertex.
   (c) We wish to find a flow in G with the maximum total flow leaving s (this is the *value* of the flow).
3. We build up a solution by repeatedly finding *augmenting paths* in the *residual graph* given by the flow
   (a) The residual graph has capacities representing the remaining capacity on each edge (with the given flow f) and "backwards edges" with capacity equal to the amount of flow on the corresponding edge.
   (b) An augmenting path is a path in the residual graph in which every edge has positive capacity
   (c) We compute the *"bottleneck"* of a path – the minimum capacity edge traversed in the residual graph – and *"augment"* a flow by pushing the bottleneck amount along that path. This produces another valid flow.
4. The Ford-Fulkerson Algorithm
   (a) While there is any augmenting path, augment the current flow using that path.
   (b) Note: the capacities remain integers, so the value of the flow increases by at least 1 on each iteration.
   (c) The algorithm therefore must terminate within $C = \sum_{e \text{ out of } s} c(e)$ iterations
   (d) Since depth-first search can find the paths in time $O(|E|)$ (if we have no isolated vertices) and computing the bottleneck and augment operations also can be done in time $O(|E|)$, the algorithm runs in $O(C|E|)$ time.
5. We will show correctness using a structural bound involving *"cuts"* in G (next time)
   (a) A *s-t cut* is a partition of V into A and B such that $s \in A$ and $t \in B$
   (b) The *capacity* of the cut (A, B) is the total capacity of edges crossing from A to B
   (c) We will show that for any cut (A, B), the value of the flow is the net amount crossing from A to B. This will bound the optimal value by the cut's capacity.