COS 528 Spring 2013 Tarjan

Analysis of the Highest-Label Push-Relabel Algorithm

This version of the push-relabel algorithm maintains an array of buckets indexed by label. We call a vertex *active* if it has positive excess and is not the sink. Bucket k contains the active vertices having label k. The algorithm maintains the index of the highest non-empty bucket and always does a push or relabel step at a vertex of highest label. Maintaining the index takes $O(n^2)$ time overall, since the index increases only by as much as a label increases.

Cheriyan and Mehlhorn found a beautiful potential-based analysis of this algorithm. To count non-saturating pushes, we define the potential of an active vertex to be the number of vertices (with positive excess or not, and including itself) with its label or a smaller label. We define the total potential to be the sum of the vertex potentials. A non-saturating push from a vertex of highest label decreases the potential by at least the number of vertices with the label. A saturating push can create a new vertex with positive excess. Such a push increases the potential by at most n. A label increase of 1 can also increase the potential by at most n. The total increase in potential over the entire algorithm is thus $O(n^2m)$, dominated by the increase caused by saturating pushes. Define a *phase* to consist of all non-saturating pushes that occur during a maximal interval in which the maximum label of a positive-excess vertex does not change: a phase ends either when a vertex is relabeled or when all vertices of highest label push all their excess to the next lower level. There is at most one non-saturating push per active vertex per phase, and at most $4n^2$ phases. (Why?) This gives an $O(n^3)$ bound on the number of non-saturating pushes, but there is a better bound. We call a non-saturating push *big* if there are at least k active vertices of maximum label when it occurs, and *small* otherwise. The number of small pushes is at most k per phase, at most $4kn^2$ in total. Each big push reduces the potential by at least k. Hence there are $O(n^2m/k)$ big pushes. To balance these amounts, we choose $k = m^{1/2}$. Then the total number of non-saturating pushes, as well as the running time of the algorithm, is $O(n^2m^{1/2})$).