COS 423 Notes on Fast Versions of the Preflow Push Max-Flow Algorithm
Spring 2009

1. FIFO(Round-Robin) Version

In this version of the algorithm, we maintain a queue of all the active vertices (those with positive excess). We pop the first vertex off of the queue and *discharge* it by applying pushing and relabeling steps to it until its excess is reduced to zero. If a push makes a vertex active, we inject it into the back of the queue.

The running time of this method is $O(nm)$ plus $O(1)$ per nonsaturating push, since the time for queue operations is $O(1)$ per push. (The time for the $O(n^2)$ relabelings is $O(nm)$, the number of saturating pushes and the time they take is $O(nm)$, the time per nonsaturating push is $O(1)$, and the time spent finding edges on which to do pushes is $O(nm)$ plus $O(1)$ per nonsaturating push.) We shall bound the number of nonsaturating pushes, and hence the total running time, by $O(n^3)$. To do this, we define *passes* through the queue as follows. Pass one consists of the discharges done on vertices initially on the queue (once the arcs out of s are saturated). Pass $k+1$ consists of the discharges done on vertices added to the queue during pass k. During each pass, there is at most one discharge per vertex, and thus at most one nonsaturating push per vertex, since such a push reduces the vertex excess to zero. We shall derive an $O(n^2)$ bound on the number of passes, giving the desired $O(n^3)$ bound on the number of nonsaturating pushes. Since there are $O(n^2)$ relabelings, there are $O(n^2)$ passes that do at least one relabeling. To bound the number of passes that do not do a relabeling, let Φ be $\max\{d(v) \mid e(v) > 0\}$. Φ is always at least zero and at most $2n^2$. A pass that does not do a relabeling decreases Φ by at least one. The total increase in Φ over all passes is at most the total label increase, which is at most $2n^2$. Hence the number of passes that do not do a relabeling is $O(n^2)$. Hence the total number of passes is $O(n^2)$.

2. Big-Excess Version(assuming integer arc capacities)

In this version of the algorithm, we maintain a parameter V that is an upper bound on the maximum excess. Initially V is the initial maximum excess, equal to the maximum of the capacities of the arcs leaving the source, say U. Pushes are only done on vertices with excess exceeding $V/2$. We call such vertices *big*. Once there are no big vertices, V is divided by 2 and rounded down to the nearest integer, and pushing continues. Once V is less than one, all excesses must be zero, since the algorithm maintains flow integrality. In order to guarantee that the algorithm always makes forward progress, we must modify the pushing step so that it never creates a vertex excess exceeding V. In particular, when pushing from v to w, the amount of flow moved is $\min\{e(v), c_f(v,w), V - e(w)\}$.

We still need a way to select big vertices for processing. A good method is to select a big vertex of minimum label. With this method, any nonsaturating push moves at least $V/2$ units of flow. Consider the potential function $\Phi = \Sigma (e(v)d(v) / (2V))$. This potential is initially zero, always non-negative, and at most $4n^2$. Any push reduces the potential, a nonsaturating push reduces it by at least one, and the only increases in potential are due to relabelings, which cause a total increase of $O(n^2)$, and changes in V, each of which can cause an increase in Φ by up to $2n^2$ (from $2n^2$ to $4n^2$). Thus the total number of nonsaturating pushes is $O(n^2 \log U)$, and the total running time is $O(n^2 \log U + nm)$, assuming that the overhead to select big vertices for processing is not too large.

Exercise: Describe a way to implement big vertex selection within the claimed time bound.