CSE 5311-501 Lab Assignment 3

Due April 10, 2003

Goals:

Understanding of push-relabel techniques for network flows.

Requirements:

1. Modify the program `preflowPushFIFO.c` to eliminate the test that distinguishes between unsaturated and saturated edges in `preflowPushLoop()`. Your program must compile and execute on OMEGA.

2. Email your code (a ***single*** source file) as an attachment to `yxb4544@omega.uta.edu` before 5:30 pm on April 10. The subject should include your name as recorded by the University.

Getting Started:

1. The input (`stdin`) will consist of the following:

   a. A line with `n` (number of vertices) and `m` (number of edges). Vertices will be numbered from `0` (source) to `n - 1` (sink).
   b. `m` lines, each including the tail, the head, and the capacity for an edge. These are unordered.

2. Your submitted program should only produce output as in the provided program. All other tracing should be disabled.

3. Before making any significant changes, you should add code to the provided program to count the number of times the test `if (edgeTab[k].capacity-edgeTab[k].flow>0)` fails. Your goal is to eliminate the cost of these many failures by separating unsaturated edges from saturated edges (i.e. the failures) in the `edgeTab`.

4. Since all edges with vertex `i` as the tail were stored together using `edgeTab[firstEdge[i]]` through `edgeTab[firstEdge[i+1]-1],` it is convenient to add another table `firstSaturatedEdge` (similar to `firstEdge`) such that unsaturated edges exiting vertex `i` are stored in `edgeTab[firstEdge[i]]` through `edgeTab[firstSaturatedEdge[i]-1]`, while saturated edges exiting vertex `i` are stored in `edgeTab[firstSaturatedEdge[i]]` through `edgeTab[firstEdge[i+1]-1]`. Degenerate cases, where vertex `i` has only unsaturated edges or only saturated edges, may occur.

5. Whenever flow is pushed, an edge may change from unsaturated to saturated (in O(1) time). Similarly, the inverse for the edge may change from saturated to unsaturated. This may be handled by swapping an edge in the saturated part with an edge in the unsaturated part. Since each `edgeTab` entry points at its inverse, it is important that these also get updated.

6. The vertices in the `inQueue` are ‘‘overflowing’’. A vertex removed from `inQueue` will have pushes and lifts until there is no excess.