COS 423 Problem Set 5 Due Wednesday April 13 11AM
Spring 2011
Tarjan
No collaboration on Problems 1 and 4; collaboration allowed on 2 and 3.

1. The following "proof" of correctness of the two-way strong components algorithm (Lecture 15) appears in at least two textbooks. The proof is not correct. Find the error: give a graph, and a run of the algorithm on the graph, for which some statement in the proof is false. Brackets are my added clarifications. Double brackets provide a missing explanation; this is not where the bug in the proof is.

"We have claimed that the vertices of a strongly connected component correspond precisely to the vertices of a tree in the spanning forest of the second depth-first search [the backward search]. To see why, observe that if *v* and *w* are vertices in the same strongly connected component, then there are paths in G from *v* to *w* and from *w* to *v*…

Suppose that in the [second] depth-first search of G, we begin a search at some root *x* and reach either *v* or *w*. Since *v* and *w* are reachable from each other, both *v* and *w* will end up in the spanning tree with root *x*. [[This argument is incomplete. Here is the missing part. Suppose *v* or *w* is visited during the search backward from *x*. If any vertex on the cycle consisting of the path from *v* to *w* followed by the path from *w* to *v* is visited during an earlier backward search from a different root, then both *v* and *w* would have been visited during that search (or even earlier) as well. Thus all vertices on the cycle are unvisited when the search from *x* starts, and they will all be visited during the search from *x*, including both *v* and *w*.]]

Now suppose *v* and *w* are in the same spanning tree of the depth-first spanning forest [generated by the backward search]. We must show that *v* and *w* are in the same strongly connected component. Let *x* be the root of the [backward] spanning tree containing *v* and *w*. Since *v* is a descendant of *x*, there exists…a path from *v* to *x*.

In the construction of the [backward] spanning forest, vertex *v* was still unvisited when the depth-first search at *x* was initiated. Thus *x* has a higher [postorder] number than *v*, so in the [forward] depth-first search of G, the recursive call at *v* terminated before the recursive call at *x* did. But in the [forward]… search…, the search [from] *v* could not have started before [the search from] *x*, since the path in *G* from *v* to *x* would then imply that the search [from] *x* would start and end before the search [from] *v* ended.

We conclude that in the [forward] search, *v* is visited during the search [from] *x* and hence *v* is a descendant of *x* in the [forward] depth-first spanning forest… Thus there exists a path from *x* to *v* in *G*. Therefore *x* and *v* are in the same strongly connected component. An identical argument shows that x and w are in the same strongly connected component and hence v and w are in the same strongly connected component, as shown by the path from v to x to w and the path from w to x to v."

2. We can define the blocks of an undirected graph without reference to cut vertices, as follows: The blocks are the subgraphs whose edge sets are the finest partition of the edges such that two edges on a common simple cycle (no repeated vertices) are in the same block of the partition.

(a) Prove that if edges (u, v) and (w, x) are on a simple cycle, and edges (w, x) and (y, z) are on a (possibly different) simple cycle, then there is a simple cycle containing (u, v) and (y, z). Use this to show that two edges are in the same block if and only if they are on a common simple cycle.

In the remaining parts of the problem, assume we are given a connected graph G and we do a depth-first search of G, generating a DFS tree T and directing each edge along the direction of the first advance during the search.

(b) Show that each block has a unique tree arc (x, y) which is the first arc traversed in the block and is such that (i) every vertex in the block other than x is a descendant of y; and (ii) if a vertex v other than x is in the block, then every tree arc on the tree path from y to v is in the block.

(c) Prove that the DFS algorithm for computing blocks presented in lecture is correct. That is, each block receives a distinct number greater than n, and an edge (v, w) is in the block whose number is $\min\{pre(v), pre(w)\}$. Hint: You may wish to assume that, instead of putting vertices on stacks R and S, the algorithm puts the corresponding tree arcs on R and S. This may make it easier to reason about the correctness of the algorithm. You will want to prove that the algorithm maintains an invariant relating the blocks of the edges traversed so far to the contents of the stacks R and S.

3. We define the *strong blocks* of a directed graph to be the subgraphs whose arc sets are the finest partition of the arcs such that two arcs on a common simple cycle are in the same block of the partition.

(a) Give an example to show that (unlike the undirected case (2a)) there is a directed graph in which there are three arcs (u, v), (w, x), and (y, z) such that (u, v) and (w, x) are on a common simple cycle, and (w, x) and (y, z) are on a common simple cycle, but no simple cycle contains both (u, v) and (y, z).

(b) Give an $O(n + m)$-time algorithm to find the strong blocks of a given directed graph. You may use any of the algorithms discussed in class. Hint: prove that the strong blocks can be found using the following approach: First, find the strong components. Next, within each strong component, ignore the arc directions and find the blocks of the resulting undirected graph.

4. A depth-first search of the following directed graph is performed and results in the preorder numbering in the figure. We identify vertices by their preorder numbers.

(a) Identify the type of each arc: tree, forward, back (cycle), or cross.

(b) Give the semi-dominator, relative dominator, and immediate dominator of each vertex. Draw the dominator tree.