Notes on finding Strong Components in a Directed Graph.

Dan Gusfield, Jan. 31, 2013.

I don't like the way the book describes the algorithm to find strong components in a directed graph. So here I am going to rewrite the algorithm (given on page 104, 8 lines from the bottom of the page) to find the strong components of a graph. HOWEVER, in order to understand my version, you still need to read and understand the discussion in the book, up to that point, on how to find strong components. Generally, I like the way the book explains the ideas that lead up to the algorithm on page 104, although I also want to expand on the explanation of Property 1:

First let's define what it means to "visit" a node v": A node v is "visited" precisely when the variable *visited(v)* is set to **true** in Explore.

Then we can restate

Property 1: If the Explore subroutine is started at node u in graph G, then it will terminate precisely when all nodes reachable in G from u have been visited. However, some of those nodes reachable from u in G might have been visited in a prior call of Explore. So the call Explore(G,u) will only visit those unvisited nodes that are reachable from u in G.

Note that if no nodes in a graph H have yet been visited, then the call Explore(H,u) will actually visit all the nodes that are reachable from u in H.

Now we return to the strong component algorithm itself. You can replace what is in the book on page 104 (8 lines up from the bottom) until the end of the chapter (on page 105) with the following:

> 1. Run depth-First search on G^R and assign pre and post numbers to all the nodes. These numbers will be kept and used throughout the algorithm. Lets use the notation $P(v)$ to describe the post number assigned here to a node v.
>
> As explained earlier, the node w with the highest post-number $P(w)$ will be in a source strong component in G^R, and hence will be in a sink strong component in the original G.
>
> Make a copy of graph G, and the post-numbers P, and call that graph H.
>
> Now repeat Step 2 until H is empty:

2. Starting at node w with the highest post-number $P(w)$ in the current H, run a *directed* Explore in H. That is, call Explore(H, w) in the current directed graph H. Note that no nodes in the current H have yet been visited. So, by property 1, that call of Explore from w will visit all and only the nodes in the strong component C in H (and hence in G) that node w is in. So output the nodes in C as the a strong component. Then remove the nodes in C from H, i.e., change the current H.

Now lets look at the example of graph G in Figure 3.9. Graph G^R is shown in Figure 3.10. There are many possible DFSs that could be done in G^R, but lets use that the DFS that assigns the following pre and post numbers:

```
A: 1,2
B: 3,6
C: 7,10
D: 11,12
E: 4,5
F: 8,9
G: 13,14
H: 15,16
I: 17,24
J: 18,23
K: 20,21
L: 19,22
```

The P number is the second number in each pair. You should verify that there is a DFS of G^R that assigns those P numbers.

Using those P numbers, the first Explore in H starts at node I and finds the strong component C containing nodes $\{I, J, K, L, H, G\}$. That strong component is output and those nodes are deleted from H. The next Explore begins at node D and finds the strong component containing only D. From that point the algorithm finds strong components $\{C, F\}$, $\{B, E\}$, $\{A\}$.

Note that my version of the algorithm, and the one in the book, find the strong components in the same order.

You should be clear on three points: 1) No matter what DFS is used in Step 1, exactly the same set of strong components are discovered in Step 2.

2) However, the order in which they are discovered can differ, depending on the DFS that is done in Step 1. 3) Once the the Post numbers P are set in Step 1, the order that the strong components will be found in Step 2 is completely determined.