Notes on Generating Strong Components, §3.6

Let G be an arbitrary directed graph. We split G into strongly connected components, two points v, w being in the same component if you can get from either to the other. Call the components $C_1, \ldots, C_r$. We create a supergraph, thinking of C_i as a superpoint and putting a directed edge from C_i to C_j if there is any directed edge from any $v \in C_i$ to any $w \in C_j$. This supergraph is a DAG. (Reason: If, say, $C_1, C_2, \ldots, C_s, C_1$ formed a cycle you could get from any v in any of those C_i to any other such w and so they would have formed one strongly connected component.) We know that DAGs have topological orders. So we can (and will) but the components in that ordering. That is, the vertices are partitioned into strongly connected components $C_1, \ldots, C_r$ where all directed edges between components go from a smaller indexed component to a larger indexed component.

Now we give an algorithm that will find the components C_i. For $1 \leq i \leq r$ (but we don't know r in advance of course) `STRONGCOMPONENT`$[i]$ will be a queue (its really just a set) which will be the set C_i. We use `COMPONENTCOUNT` to update which component we are working on.

`Step 1:` Apply `DFS`$[G]$ as described in the DAGS/ Top Order notes. Use colors white (not yet found), grey (`DFS`$[v]$ begun) and black (`DFS`$[v]$ ended) as before. Create the stack `GOODORDER` as before. *However*, if $w \in \mathbf{Adj}[v]$ and has color `grey` we do not stop the procedure but simply ignore it. In the end `GOODORDER` will have all of the vertices of G in the reverse order (as it was a stack) of the time they became black. That is, the first v on the stack will be the last v for which `DFS`$[v]$ was completed. (Of course, `GOODORDER` is no longer necessarily a topological order, G may not have one, but it will have some nice properties!)

`Step 2:` Now we apply `DFS` to the graph G^{rev} (the graph with arrows reversed! This is done by using the **InAdj** list rather than the normal (outadjacency) list), taking the vertices in the order `GOODORDER`. For each new vertex v we punch out v and the vertices found by `DFS`$[v]$ as a strongly connected component. Here is the pseudocode:

`LISTSTRONGCOMP` (*Assume `GOODORDER` has already been created *)
(* Initialization *)
`COMPONENTCOUNT` $\leftarrow 0$
For all vertices v
    `COLOR`$[v] \leftarrow$ `white`
(* End Initialization *)
For each vertex v, *using the order* `GOODORDER`
    If `COLOR`$[v] \leftarrow$ `white`

COMPONENTCOUNT $++$
STRONGCOMPONENT[COMPONENTCOUNT] $\leftarrow \emptyset$ (*initiate new component *)
DFSMAKECOMP[v]
EndIf
EndFor

The center of the algorithm is DFSMAKECOMP[v]

DFSMAKECOMP[v]
ADDTOQUEUE[STRONGCOMPONENT[COMPONENTCOUNT], v] (*add v to component *)
For all $w \in \mathbf{InAdj}[v]$
If COLOR[w] $\leftarrow$ white (* New point *)
DFSMAKECOMP[w] (* recursive call *)
EndIf
(* Ignore black and grey vertices *)
Endfor
COLOR[v] $\leftarrow$ black (* v has been completed *)

That does it. It helps to think of DFSMAKECOMP[v] at the start of LISTSTRONGCOMP when all vertices are white. It will set COMPONENTCOUNT $\leftarrow$ 1 and create a queue STRONGCOMPONENT[1] with all the points found. It will explore all vertices w that can reach v in G (equivalent, that are reachable from v in G^{rev}) and call DFSMAKECOMP[w] for those w and all those w will be come black and then when DFSMAKECOMP[v] finishes, v itself becomes black. At this point some (sometimes all, in which case we're done) of the vertices are black but none are grey. When we find another white vertex v' and set COMPONENTCOUNT $\leftarrow$ 2 and call DFSMAKECOMP[v'] and create STRONGCOMPONENT[2]. While doing that it is as if the black vertices (from LISTSTRONGCOMP[v]) weren't there.

Why does it work? The key is the property of GOODORDER. Roughly, GOODORDER will be the v from the earlier components (with smaller i) higher in the stack. Here is the exact result:

Claim: Suppose that in the supergraph we have a directed path from C_a to C_b. Then in the stack GOODORDER the first point we come to from C_a will occur before the first point we come to from C_b. In other words, there will be a vertex of C_a that turns black after all the vertices of C_b turn black.

Argument for Claim: Write the directed path as $C_{i_1}, \ldots, C_{i_s}$ with $i_1 = a$ and $i_s = b$. (There may be many such, just take one.) Now run LISTSTRONGCOMP. Eventually every point becomes grey and then black so there will be a *first time* when any point from any of the $C_{i_1}, \ldots, C_{i_s}$ is found and turns grey. Call that point v. At the time it turns grey all of the other points of

$C_{i_1}, \ldots, C_{i_s}$ are still `white`.

`Case I:` $v \notin C_a$. We now call `DFS`$[v]$. In its operation it will reach all the points reachable from v through white vertices so it particular it will reach all of the points of C_b and all of those points will turn black. But there is no path from v to C_a and so at the end of `DFS`$[v]$ all points of C_a are still white. So actually in this instance all points of C_a will turn black after all points of C_b have turned black.

`Case II:` $v \in C_a$. We now call `DFS`$[v]$. In its operation it will reach all the points reachable from v through white vertices so it particular it will reach all of the points of C_b and all of those points will turn black. But this is all during the operation of `DFS`$[v]$ and v itself only becomes black when `DFS`$[v]$ is over so this is after all of the points of C_b have turned black.

Back to why it works: OK, we've done `DFS`$[v]$ and created `GOODORDER`. We take the first v from that stack, say $v \in C_b$. Can there be any C_a which arrows C_b in the supergraph. No! For by the claim that would put some vertex of C_a above v in `GOODORDER`. That is, there is not edge *into* C_a. We run `LISTSTRONGCOMP`$[v]$, getting all the points from which we can reach v. This includes all the points of C_b (by strong connectivity of C_b) and none others. Thus `STRONGCOMPONENT`$[1]$ will consist of exactly C_b.

What about the general case. Suppose we have already outputted the strong components $C_{b_1}, \ldots, C_{b_s}$. Now say the next point in `GOODORDER` is $v \in C_b$. From the claim there can be no C_a which arrows C_b in the supergraph other than the $C_{b_1}, \ldots, C_{b_s}$. (Why? Had there been one of its points would have been ahead of v in the stack!) At this stage $C_{b_1}, \ldots, C_{b_s}$ are all black. We run `LISTSTRONGCOMP`$[v]$, getting all the points from which we can reach v with the black points excluded. So we get exactly C_b. That is, each time we generate exactly a strong component and in the end we have gotten all of the points so we have generated precisely the strong components.