CS 512: Comments on Vertex Ordering

16:198:512
Instructor: Wes Cowan

The algorithms we are currently discussing (connectivity, strongly connected components, linearization, etc) all seem to depend heavily on the order in which the vertices are processed. As presented in the book, these algorithms usually hinge on a for-loop over the set of vertices - but this does not specify an ordering to use. However, it turns out that the structural properties of pre/post numbering are such that order, to a large extent, does not matter. This is proven more concretely in the book, based on the idea that the highest post numbers are assigned in source components (why?), but I want to present a small example.

To move away from the visual representation and more towards a data processing notion, consider the following graph (as represented by an adjacency list:

$$\text{Graph} = \{A : [D, G], B : [H], C : [B, E], D : [F], E : [A], F : [B], G : [C], H : [D]\}.$$

Consider the problem of (algorithmically) identifying the strongly connected components of this graph. The first step is to do pre/post numbering on the *reverse* of the graph:

$$\text{Graph}_R = \{A : [E], B : [C, F], C : [G], D : [A, H], E : [C], F : [D], G : [A], H : [B]\}.$$

If the highest post numbers are assigned in source components, by reversing the graph, the highest post numbers are assigned in sink components of the original graph. We can then use this to identify and remove sink components, allowing us to peel off strongly connected components of G.

Consider processing the vertices of Graph_R via depth first search in the following order: A, B, C, D, E, F, G, H.

In this case, we generate pre/post numbers:

	A	B	C	D	E	F	G	H
Pre	1	9	3	11	2	10	4	12
Post	8	16	6	14	7	15	5	13

Ordering the vertices by *decreasing post number* yields: B, F, D, H, A, E, C, G. If we then utilize this vertex ordering on the original Graph, this will identify $[B, H, D, F]$ as the first (sink) strongly connected component. Removing that, the next unprocessed vertex in the ordering is A, and that will yield $[A, G, C, E]$ as the second strongly connected component.

But what if we had started with a different initial ordering? Consider processing the vertices of Graph_R via depth first search in the following order: B, C, F, H, G, E, D, A. (This is not a purely randomized order, but is geometrically reasonable given my own drawing / labeling of the graph.)

In this case, we generate pre/post numbers:

	A	B	C	D	E	F	G	H
Pre	4	1	2	11	5	10	3	12
Post	7	16	9	14	6	15	8	13

Note, this has produced similar, but distinct pre/post numbers. Ordering the vertices by *decreasing post number* yields: B, F, D, H, C, G, E, A. Again, this is similar to but distinct from the vertex ordering that resulted in the original case. Using this vertex ordering to process Graph will identify the first (sink) strongly connected component as $[B, H, D, F]$ and the second strongly connected component as $[C, E, A, G]$.

Note then that either ordering produced the same structural results: two meta nodes $[BDFH]$ (sink) and $[ACEG]$ (source). Different initial vertex process orders may generally change the order of vertices in a s.c. component (as observed here), and may change the resulting order of those components (not observed here, as the example only has two s.c. components), but the overall structure will be the same.