(M. Dietzfelbinger, 2020-01-09)

3.4 Notes about Kosaraju's algorithm

Definition 3.4.1 (Strongly connected components)**.** *Let $G = (V, E)$ be a directed graph. For nodes $v, w \in V$ define $v \rightsquigarrow w$ if there is a path that leads from v to w. Define $v \leftrightsquigarrow w$ if $v \rightsquigarrow w$ and $w \rightsquigarrow v$. Note that $\leftrightsquigarrow$ is an equivalence relation (reflexive, symmetric, transitive). The equivalence classes are called* ***strongly connected components*** *(sccs) of G.*

One easily sees that $v \leftrightsquigarrow w$ holds if and only if there is a cycle $v = v_0, v_1, \ldots, w = v_i, v_{i+1}, \ldots, v_r = v$ that contains both v and w.

A graph G is called *strongly connected* if it has only one strongly connected component, i. e., if $v \leftrightsquigarrow w$ holds for all $v, w \in V$.

We need a fundamental lemma on DFS.

Lemma 3.4.2 (White Path Theorem)**.** *If we carry out DFS in any digraph, the following holds: w is a descendant of v if and only if at the point in time when* explore(v) *is called, there is a path of new (i. e., "white") nodes from v to w.*

(The *proof was given in class.*)

Corollary 3.4.3. *When* DFS(G) *is run on $G = (V, E)$, then for each strongly connected component $C \subseteq V$ we have that all nodes of C are in one DFS tree. (However, one tree may contain several strongly connected components.)*

Proof: Consider C and let w be the first node in C that is discovered in the course of the DFS. When explore(w) starts, all nodes in C are new. We have that each node v in C is on a cycle with w, which means that all nodes on this cycle also belong to C, and that the path from w to v along the cycle is a „white" path when explore(w) starts. By Lemma 3.4.2, v becomes a descendant of w. – Thus, all nodes in C are in the same tree as w. $\square$

Consider Algorithm 1, called **Kosaraju's algorithm**. (An example is given in class.)

Theorem 3.4.4. *Kosaraju's algorithm outputs the strongly connected components of G. Its running time is $O(|V| + |E|)$.*

Proof: We don't have to say much about the running time. DFS has running time $O(|V| + |E|)$, and the reverse graph can be computed in time $O(|V| + |E|)$.

The only interesting point is correctness.

We first observe that G and G^{R} have the same strongly connected components, since the cycles in G correspond to the cycles in G^{R} in reversed order.

Algorithm 1: scc(G) (Kosaraju's algorithm)

```
// $G = (V, E)$ is a digraph.
// Goal: Find the strongly connected components of $G$.
(1) Do DFS($G$), finding post($v$) for all nodes in $G$;
(2) Calculate $G^{\mathsf{R}}$, by reversing all edges of $G$;
(3) Do DFS($G^{\mathsf{R}}$), where in the outer loop
      nodes are checked in order of decreasing post numbers (from (1));
(4) output the node sets of the DFS trees from (3).
```

Corollary 3.4.3, applied to G^{R}, now says that each scc is contained in one of the trees built in part (3). So there are at most as many trees as scc's. It remains to show that each scc forms one tree by itself. For this, we show that each scc contains a root node. Then there must be (at least) as many DFS trees as scc's, which means that each tree contains exactly one scc.

Lemma 3.4.5. *Let C be an arbitrary scc of G, and let w be the node in C with the largest* post *number. Then w is a tree root in the DFS in part* (3) *of the algorithm.*

Proof: By the properties of DFS, w is a tree root in the DFS in G^{R} if and only if it is unreachable in G^{R} from any node u that is checked before w in the outer loop of the DFS. By the algorithm, exactly the nodes u with higher post numbers (from (1)) than w are checked there. Thus, we must show that in G^{R} w is unreachable from nodes u with a higher post number (from (1)). Equivalently, we must show that no node u with a higher post number than w is reachable from w in G, or:

Claim: If $w = v_0, v_1, \ldots, v_t = u$ is a path in G, then $\text{post}(u) \le \text{post}(w)$.

Proof of Claim: We may assume $u \ne w$. We consider the point in time in the first DFS (1) in G when explore(w) is called. There are three cases:

Case 1: All nodes on the path $w = v_0, v_1, \ldots, v_t = u$ are new. – By Lemma 3.4.2 (the White Path Theorem) node u becomes a descendant of w, hence $\text{post}(u) < \text{post}(w)$.

Case 2: One of the nodes v_i on the path is active. – This means that the path of currently active nodes runs through v_i and ends at w (these edges are tree edges). Together with the piece $w = v_0, v_1, \ldots, v_i$ we get a cycle on which we have w and v_i. This means that w and v_i are equivalent in the sense of being mutually reachable, and hence $v_i \in C$. However, explore(v_i) is active when explore(w) starts, and hence $\text{post}(v_i) > \text{post}(w)$. This contradicts the choice of w as the node with highest post number in C. Hence this case cannot occur.

Case 3: One of the nodes v_i on the path is finished, and no node is active. – By the explore(v_i) procedure, we must have that also v_{i+1} is finished, and v_{i+2}, and so on, and finally that $v_t = u$ must be finished when explore(w) starts. This implies $\text{post}(u) < \text{post}(w)$.

This finishes the proof of Theorem 3.4.4. □