Lecture notes for Feb 8, 2023
Finding strongly connected components and topological ordering of digraphs

Chun-Hung Liu

February 8, 2023

A *strongly connected component* of a digraph D is a maximal induced subgraph B of D satisfying that for any two vertices x, y in B, there exist a directed path in D from x to y and a directed path in D from y to x.

We proved the following two results last time.

Proposition 1 *Let D be a digraph.*

1. *$\{V(M) : M$ is a strongly connected component$\}$ is a partition of D.*
2. *Every directed cycle in D is contained in some strongly connected component of D.*
3. *Every strongly connected component of D containing at least two vertices contains a directed cycle.*

Corollary 2 *A loopless digraph is acyclic if and only if every its strongly connected component has exactly one vertex.*

1 Finding strongly connected components

We can apply DFS to digraph in a very similar way, except that we only traverse edges from tails to heads. Then the rooted tree constructed in this way has the property that any tree path between two vertices u, v, where u is an ancestor of v, is a directed path from u to v. It makes us be able to

find all strongly connected components of a digraph in a very similar way as we find all blocks of a graph.

Before we state the algorithm, we remark that the definition of back edges with respect to a rooted tree in a digraph is slightly different from the ones for graphs because of the directions. Let T be a rooted tree in a digraph D. We say an edge e in $E(D) - E(T)$ is a *back edge* if the tail of e is a proper descendant of the head of e. And we say an edge e in $E(D) - E(T)$ is a *forward edge* if the head of e is a proper descendant of the tail of e.

Recall that the rooted tree that we will construct by DFS in D has the property that any tree path between two comparable vertices is a directed path from the ancestor to the descendant. So every back edge e together with the tree path connecting the ends of e is a directed cycle C in D, so C is contained a strongly connected component of D.

Unlike the DFS tree of undirected graphs, it is possible to have an edge of a digraph between two incomparable vertices in the DFS tree. But such an edge (x, y) must satisfy that the tail x is added into the tree later than the head y.

A digraph D is *simple* if its edge-set is set of ordered pairs of vertices of D with distinct entries. That is, D has no edge whose tail equals its head, and D has no two edges that have the same tail and the same head. But two edges with the same ends but with different directions are allowed.

Now we state the algorithm.

==========

An algorithm for finding strongly connected components
Input: A simple digraph D.
Output: The set of the vertex-sets of all strongly connected components of D.
Procedure:

Step 0: Set all edges and vertices of D as "unmarked". Set $\mathcal{B} = \emptyset$.

Step 1: Terminate the algorithm if all vertices of D are marked. Otherwise, do the following: Pick an unmarked vertex of D and call it r. Set T to be the rooted tree $(\{r\}, \emptyset)$ rooted at r. Set S to be the sequence (r). Set $B_r = \{r\}$. Set $f(r)$ to be the minimum positive integer that is not in the image of f. Set $g(r) = f(r)$.

Step 2: If S is empty, then do Step 1. Otherwise, do the following: Say the last entry of S is v. If there exists an unmarked edge of D with tail v, then do Step 2-1; otherwise, do Step 2-2.

Step 2-1 : Pick an unmarked edge e of D with tail v. Mark e. Let u be the head of e.

* If $u \notin V(T)$ and u is unmarked, then add e into T, add u into S as the last entry, set $B_u = \{u\}$, define $f(u)$ to be the minimum positive integer that is not in the image of f, define $g(u) = f(u)$, and then repeat Step 2.
* If $u \in V(T)$, then redefine $g(v)$ to be $\min\{g(v), f(u)\}$, and then repeat Step 2.

Step 2-2:

* If $g(v) = f(v)$, then mark all vertices in B_v, add B_v into $\mathcal{B}$, and then remove v from S and repeat Step 2.
* If $g(v) < f(v)$, then v has the parent in T, called the parent p, and we set $B_p = B_p \cup B_v$, redefine $g(p)$ to be $\min\{g(p), g(v)\}$, and then remove v from S and repeat Step 2.

==========

Lemma 3 *The following properties are preserved during the entire process:*

1. *For every vertex v of G, all vertices in the current B_v are descendants of v in the current tree T.*
2. *The marked vertices of D are exactly the vertices contained in members of $\mathcal{B}$.*

Proof. It is straightforward to verify that these properties are preserved. ■

Lemma 4 *Let v be a vertex of D. Let a_v be the ancestor of v such that B_{a_v} is added into $\mathcal{B}$ when a_v is about to be removed from S, and subject to this property, a_v is as far from r as possible. (Note that a_v exists since r is a candidate for a_v.) Then when a_v is about to be removed from S, v is contained in B_{a_v} and B_{a_v} is added into $\mathcal{B}$.*

Proof. It is easy to verify this lemma from the description of the algorithm. ■

Lemma 5 *$\mathcal{B}$ is a partition of $V(D)$.*

Proof. By Lemma 4, the union of the members of $\mathcal{B}$ contains $V(D)$. Let v be a vertex of D. Let a_v be the vertex mentioned in Lemma 4. Then no proper ancestor x of a_v satisfies $v \in B_x$ since B_{a_v} is not merged into $B_{p'}$, where p' is the parent of a_v. Hence members of $\mathcal{B}$ are disjoint by property 1 in Lemma 3. So $\mathcal{B}$ is a partition of $V(D)$. ■

Lemma 6 *The following properties are preserved during the entire process:*

1. *g is the function with domain $V(T)$ such that if v is the last entry of the current S, then either*
   - *$g(v) = f(v)$ and there exists no marked edge from a descendant of v in the current tree T to an unmarked vertex u with $f(u) < f(v)$, or*
   - *$g(v)$ equals the smallest i satisfying that there exists a marked edge from a descendant of v in the current tree T to an unmarked vertex u with $f(u) = i < f(v)$.*
2. *For every vertex v of D with $g(v) < f(v)$, there exists a directed path in D from v to u, where u is the vertex with $f(u) = g(v)$.*
3. *There exists no edge of D from $\bigcup_{M\in\mathcal{B}} M$ to $V(D) - \bigcup_{M\in\mathcal{B}} M$.*
4. *For every unmarked vertex v of G with $g(v) < f(v)$, we have that u is unmarked, where u is the vertex with $f(u) = g(v)$.*

Proof. Now we show that property 1 is preserved. Observe that for every vertex x, x is marked when the proper ancestor a_x of x mentioned in Lemma 4 is removed from S. Then it is straightforward to verify that property 1 is preserved by using this observation and the fact that the entires of S form a tree path starting at the root.

Property 2 holds by property 1 by considering the edge from a descendant of v to u and a tree path from v to that descendant. By property 1 and the DFS ordering, property 3 is preserved.

Now we prove that property 4 is preserved. Suppose to the contrary that u is marked. Then u is marked when a_u is removed from S, where a_u is the vertex mentioned in Lemma 4. Since v is unmarked, a_u is not an ancestor

of v. In particular, u is not an ancestor of v. Since $f(u) = g(v) < f(v)$, u was unmarked when v is in S by property 1. So $f(a_u) > f(v)$. It is a contradiction since $f(a_u) \leq f(u) < f(v)$. ■

Lemma 7 *Let v be a vertex of D such that the final $g(v)$ is smaller than $f(v)$. Then there exists a directed path in D from v to a proper ancestor of v.*

Proof. Suppose to the contrary that v is a counterexample of this lemma such that $f(v)$ is as small as possible. Let u be the vertex with $f(u) = g(v)$. By property 2 of Lemma 6, there exists a directed path in D from v to u. And by property 4 of Lemma 6, u is marked no earlier than v. Let x be the vertex of D such that x is marked no earlier than v and there exists a directed path P_v in D from v to x, and subject to this, $f(x)$ is as small as possible. Note that x exists since u is a candidate. So $f(x) \leq f(u) = g(v) < f(v)$. Since v is a counterexample, x is not an ancestor of v. By the minimality of $f(v)$, if $g(x) < f(x)$, then there exists a directed path P_x in D from x to a proper ancestor y of x, so there exists a directed path in $P_v \cup P_x$ from v to a vertex y with $f(y) < f(x)$, and y is marked no earlier than x and hence no earlier than v, contradicting the choice of x. So $g(x) = f(x)$. Since x is not an ancestor of v and $f(x) < f(v)$, x is marked before v is added into S, a contradiction. ■

Lemma 8 *Let B be a strongly connected component of D. Then $V(B) \subseteq M$ for some $M \in \mathcal{B}$.*

Proof. Since $\mathcal{B}$ is a partition of $V(D)$ by Lemma 5, it suffices to show that $V(B)$ only intersects at most one member of $\mathcal{B}$. Suppose to the contrary that there exist distinct $M_1, M_2 \in \mathcal{B}$ with $V(B) \cap M_1 \neq \emptyset \neq V(B) \cap M_2$. By symmetry, we may assume that M_1 is added into $\mathcal{B}$ earlier than M_2. Let X be the union of all members of $\mathcal{B}$ when M_1 is just added into $\mathcal{B}$. Let $Y = V(D) - X$. Then $M_1 \subseteq X$ and $M_2 \subseteq Y$. And there exists no edge of D from X to Y by Property 2 in Lemma 6. Since B is a strongly connected component of D, there exists a directed path P in D from $V(B) \cap M_1 \subseteq X$ to $V(B) \cap M_2 \subseteq Y$. Since $\{X, Y\}$ is a partition of $V(D)$, P contains an edge from X to Y, a contradiction. ■

Lemma 9 *Let $M \in \mathcal{B}$. Then there exists a strongly connected component B of D with $M \subseteq V(B)$.*

Proof. It suffices to show that at any moment during the algorithm, for every $v \in V(D)$, B_v is contained in some strongly connected component of D. And it suffices to show that this property is preserved whenever B_v is updated to be $B_v \cup B_c$ when a child c is about to be removed from S. It only happens when $g(c) < f(v)$, which implies that there exists a directed path in D from c to a proper ancestor of c by Lemma 7. In particular it implies that there exists a directed cycle containing the edge (v, c). So v and c lie in the same strongly connected component B of D. Let C_v and C_c be the strongly connected component of D containing the current of B_v and B_c, respectively. Since strongly connected components are pairwise disjoint and $v \in B_v \subseteq V(C_v)$ and $c \in B_c \subseteq V(C_c)$, we have $C_v = B = C_c$. Therefore, once B_v is updated to $B_v \cup B_c$, it is still contained in a strongly connected component B of D. ■

Lemma 10 *$\mathcal{B}$ is the set of vertex-sets of the strongly connected components of D.*

Proof. It immediately follows from Lemmas 8 and 9. ■

Theorem 11 *Given the input digraph D, in linear time, we can find all strongly connected components of D and order them into $C_1, C_2, ..., C_k$ such that for every $1 \leq i \leq k$, there exists no edge of D from $\bigcup_{j=1}^{i} V(C_j)$ to $V(D) - \bigcup_{j=1}^{i} V(C_j)$.*

Proof. By Lemma 10, the output $\mathcal{B}$ consists of the vertex-sets of the strongly connected components of D. And it is easy to see that this algorithm runs in linear time. By taking induced subgraphs of those vertex-sets, we find all strongly connected components of D in linear time. Moreover, let $M_1, M_2, ..., M_k$ be the members of $\mathcal{B}$, ordered by the time that they were added into $\mathcal{B}$. For each $i \in [k]$, let C_i be the strongly component of D with $V(C_i) = M_i$. Then Property 3 in Lemma 6 implies that for every $1 \leq i \leq k$, there exists no edge of D from $\bigcup_{j=1}^{i} M_j = \bigcup_{j=1}^{i} V(C_j)$ to $V(D) - \bigcup_{j=1}^{i} M_j = V(D) - \bigcup_{j=1}^{i} V(C_j)$. ■

2 Topological ordering

A *topological ordering* of a digraph D is an ordering of the vertices $v_1, v_2, ..., v_{|V(D)|}$ of D such that for every $1 \leq i \leq |V(D)|$, every edge of D between $\{v_1, v_2, ..., v_i\}$ and $\{v_{i+1}, v_{i+2}, ..., v_{|V(D)|}\}$ has its tail in $\{v_1, v_2, ..., v_i\}$.

Proposition 12 *Let D be a digraph. If D has a topological ordering, then D is acyclic.*

Proof. Suppose to the contrary that D has a directed cycle C. Let $v_1, v_2, ..., v_{|V(C)|}$ be the vertices of C appearing in the topological ordering of D in the order listed. Since C is a directed cycle, there exists a directed path in C from $\{v_2, v_3, ..., v_{|V(D)|}\}$ to v_1. So there exists an edge e of C from $\{v_2, v_3, ..., v_{|V(D)|}\}$ to v_1. Hence e is an edge of D from $\{x \in V(D) : v_1 \prec x\}$ to $\{x \in V(D) : x \preceq v_1\}$, where $\preceq$ is the topological ordering of $V(D)$, a contradiction. ■

Corollary 13 *A loopless digraph has a topological ordering if and only if it is acyclic. Moreover, given a loopless digraph D, we can find either a topological ordering of D or a directed cycle in D in linear time.*

Proof. Let D be a loopless digraph. We can remove parallel edges from D to make it simple in linear time. Run the linear time algorithm mentioned in Theorem 11 to obtain all strongly connected components of D and the corresponding ordering. Check the size of each strongly connected components (which can be done in linear time). If every strongly connected component contains exactly one vertex, then we obtain a topological ordering of D by reversing the order of the strongly connected components, which takes linear time; if some strongly connected component B contains at least two vertices, then we can find a directed cycle in B by doing DFS in B, which takes linear time. (Note that if D is acyclic, then Corollary 2 implies that every strongly connected component of D has exactly one vertex, so there exists a topological ordering by the former; together with Proposition 12, we know that D has a topological ordering if and only if D is acyclic.) Therefore, in linear time, we can find either a topological ordering of D or a directed cycle in D.
■