Lecture notes for Feb 6, 2023
Finding blocks and introduction of strongly connected components of digraphs

Chun-Hung Liu

February 6, 2023

1 Finding blocks (same in the notes for Jan 30)

We go through the details of the following algorithm and the proof of correctness of the algorithm today. Detailed proofs can be found in the notes.
==
An algorithm for finding all blocks
Input: A connected simple graph G with $|V(G)| \geq 2$.
Output: The set of all blocks of G.
Procedure:

Step 0: Pick a vertex r of $V(G)$.

Step 1: Set T to be the rooted tree $(\{r\}, \emptyset)$ rooted at r. Set all edges of G as "unmarked". Set S to be the sequence (r). Set $B_r = \{r\}$ and $\mathcal{B} = \emptyset$. Set $f(r) = 1$ and $g(r) = 1$.

(It is helpful to keep in mind that S and T will be constructed in the same say as they were constructed in DFS, and the following statements hold during the entire process:

(i) For every vertex v of G, once $f(v)$ is defined, the value $f(v)$ will never be changed; but $g(v)$ might be frequently updated.

(ii) f is a bijection from the current $V(T)$ to $[|V(T)|]$ indicating the ordering of the vertices of G added into T.

(iii) 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 no back edge for the current tree T incident with v is marked, or
- $*$ $g(v)$ equals the smallest i satisfying that there exists a back edge for the current tree T between a descendant of v in the current T and the vertex u with $f(u) = i$, and this back edge is marked.

(iv) For every vertex v of G, all vertices in the current B_v are descendants of v in the current tree T.

We will prove that the algorithm really preserves the above properties later.)

Step 2: Terminate the algorithm and output $\mathcal{B}$ if S has no entry. So we may assume that S is non-empty. Say the last entry of S is v. If there exists an unmarked edge of G incident with v, then do Step 2-1; otherwise, do Step 2-2.

Step 2-1 : Pick an unmarked edge e of G incident with v and mark e. Let u be the end of e other than v.

- $*$ If $u \notin V(T)$, 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 $v = r$, then remove v from S and repeat Step 2.
- $*$ Otherwise, let p be the parent of v in T.
  - $\cdot$ If $g(v) = f(v)$, then add $\{v, p\}$ into $\mathcal{B}$, and then remove v from S and repeat Step 2.
  - $\cdot$ If $g(v) = f(p)$, then add $B_v \cup \{p\}$ into $\mathcal{B}$, and then remove v from S and repeat Step 2.

- If $g(v) < f(p)$, then 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.
  (Note that $f(v) \neq g(v) > f(p)$ is possible due to (ii), (iii) and the known fact that there is no crossing edge for a DFS tree T.)

==

Now we prove the correctness of this algorithm. Clearly, properties (i), (ii) and (iv) hold during the entire process.

Lemma 1 *Property (iii) holds during the entire process.*

Lemma 2 *Let v be a vertex of G. Let B be a block of G containing a proper ancestor of v in the final T and a proper descendant of v in the final T. Then the final B_v contains $\{x \in V(B) : x \text{ is a descendant of } v\}$ and is contained in B.*

Lemma 3 *The vertex-set of every block of G is a member of $\mathcal{B}$, and every member of $\mathcal{B}$ is the vertex-set of a block of G.*

Theorem 4 *The blocks of the input graph G can be found in linear time.*

2 Strongly connected components of digraphs

Recall that two vertices u, v are contained in the same component of a graph G if and only if there exists a path in G between u and v. Since paths have directions in digraphs, we need a bit more when considering connectivity of digraphs. 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. Note that the maximality of B implies that the aforementioned both directed paths between x and y are also in B.

Proposition 5 *Let D be a digraph.*

1. *$\{V(M) : M \text{ is a strongly connected component}\}$ 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.*

Proof. By the maximality in the definition of a strongly connected component, every vertex of D is contained in some strongly connected component of D. Suppose that there exist two distinct strongly connected components M_1, M_2 of D share a vertex z. Then the maximality implies that there exist $x \in V(M_1) - V(M_2)$ and $y \in V(M_2) - V(M_1)$. So there exist a path in M_1 from x to z and a path in M_2 from z to y, and hence there exists a path in D from x to y. Similarly, there exists a path in D from y to x. So x, y are contained in the same strongly connected component of D, a contradiction. This proves Statement 1.

Statement 2 is obvious.

If M is a strongly connected component of D containing two vertices a and b, then there exists a closed walk from a to b in D, and the shortest such walk is a directed cycle C in D containing a and b, so Statements 1 and 2 imply that C (and hence a and b) are contained in M. ∎

A digraph is *acyclic* if it does not contain a directed cycle. If a digraph is loopless, then every directed cycle contains at least two vertices. So Statements 2 and 3 of Proposition 5 imply the following.

Corollary 6 *A loopless digraph is acyclic and only if every its strongly connected component has exactly one vertex.*