Finding Strongly Connected Components

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

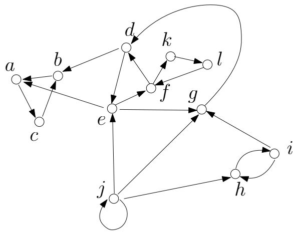
Strongly Connected Component

Let $G = (V, E)$ be a directed simple graph.

A **strongly connected component** (SCC) of G is a subset S of V s.t.

- for any two vertices $u, v \in S$, G has a path from u to v and a path from v to u;
- S is maximal in the sense that we cannot put any more vertex into S without breaking the above property.

Example



- $\{a, b, c\}$ is an SCC.
- $\{a, b, c, d\}$ is not an SCC.
- $\{d, e, f, k, l\}$ is not an SCC (because we can still add vertex g).
- $\{e, d, f, k, l, g\}$ is an SCC.

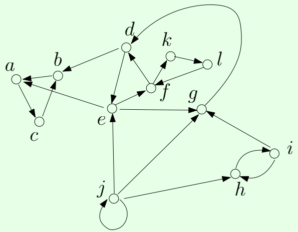
SCCs are Disjoint

Lemma 1: Suppose that S_1 and S_2 are both SCCs of G. Then, $S_1 \cap S_2 = \emptyset$.

The proof is easy and left to you.

Given a directed graph $G = (V, E)$, the goal of the **strongly connected components problem** is to divide V into disjoint subsets, each being an SCC.

Example:



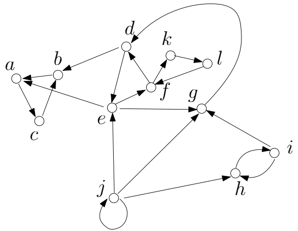
We should output: $\{a, b, c\}$, $\{d, e, f, g, k, l\}$, $\{h, i\}$, and $\{j\}$.

Algorithm

Step 1: Run DFS on G, and list the vertices by the order they turn black (i.e., popped from the stack).

If vertex $u \in V$ is the i-th turning black, we **label** u with i.

Example



Start DFS from e, re-start from h, and then another re-start from j.
The following is a possible turn-black order: $c, b, a, d, g, l, k, f, e, i, h, j$.

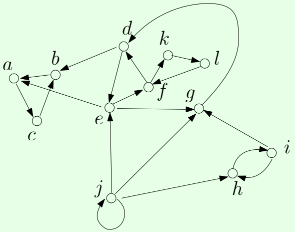
- Note: the order is not unique.

The label of c is 1.
The label of g is 5.
The label of i is 10.
The label of j is 12.

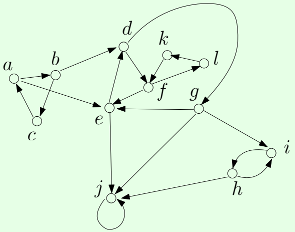
Algorithm

Step 2: Obtain the reverse graph G^{rev} by reversing the directions of all the edges in G.

Example:



Input graph



Reverse graph

Algorithm

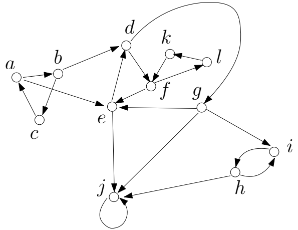
Step 3: Perform DFS on G^{rev} by obeying the following rules:

- **Rule 1:** Start at the vertex with the largest label.
- **Rule 2:** When a restart is needed, do so from the white vertex with the largest label.

Output the set of vertices in each DFS-tree as an SCC.

Example

Vertices in ascending order of label: $c, b, a, d, g, l, k, f, e, i, h, j$.
Reverse graph G^{rev}:



Start DFS from j, which finishes immediately and discovers only j.

- First SCC: $\{j\}$

Restart from h, which finishes after discovering h and g

- Second SCC: $\{g, h\}$

Restart from e, which finishes after discovering e, d, g, f, l, and k

- Third SCC: $\{e, d, g, f, l, k\}$

Restart from a, which finishes after discovering a, b, and c.

- Fourth SCC: $\{a, b, c\}$

Analysis

Theorem: Our SCC algorithm finishes in $O(|V| + |E|)$ time.

The proof is left as a regular exercise.

Next, we will prove that the algorithm correctly returns all the SCCs.

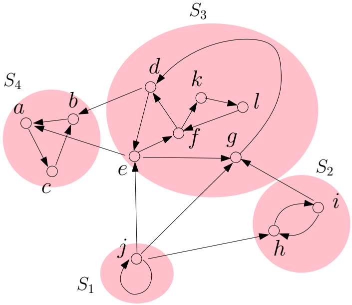
SCC Graph

Suppose that the input graph G has SCCs $S_1, S_2, ..., S_t$ for some $t \geq 1$.

The **SCC graph** G^{scc} is defined as follows:

- Each vertex in G^{scc} is a distinct SCC in G.
- For every two distinct vertices (a.k.a. SCCs) S_i and S_j $(1 \leq i, j \leq t)$, G^{scc} has an edge from S_i to S_j if some vertex in S_i can reach a vertex in S_j in G.

Example



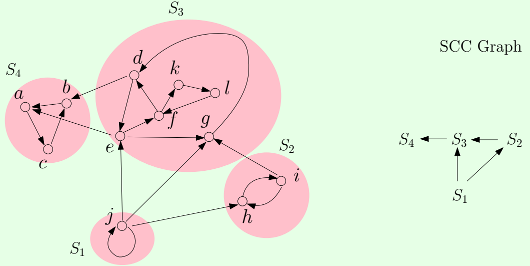
SCC Graph

$$\begin{tikzcd} S_4 & S_3 \arrow[l] & S_2 \arrow[l] \\ & S_1 \arrow[u] \arrow[ur] & \end{tikzcd}$$

SCC Graph

For each SCC S_i ($i \in [1, t]$), define

$$\textbf{label}(S_i) = \max_{v \in S_i} \text{label of } v$$



Vertices in ascending order of label: $c, b, a, d, g, l, k, f, e, i, h, j$.
$label(S_1) = 12$, $label(S_2) = 11$, $label(S_3) = 9$, $label(S_4) = 3$

SCC Graph

Lemma 2: If SCC S_i (for some $i \in [1, t]$) has an edge to SCC S_j (for some $j \in [1, t]$) in G^{scc}, then $label(S_i) > label(S_j)$.

Proof: Let u be the first vertex in $S_i \cup S_j$ that turns gray in DFS (i.e., u is the first vertex in $S_i \cup S_j$ discovered by DFS).

- If $u \in S_i$, u has a white path to every vertex in $S_i \cup S_j$. By the white path theorem, u turns black after all the vertices in S_j and is the last vertex in S_i turning black. This implies $label(S_i) > label(S_j)$.
- If $u \in S_j$, u has a white path to every vertex in S_j but no white path to any vertex in S_i. By the white path theorem, u turns black after all the vertices in S_j and before every vertex in S_i. This again implies $label(S_i) > label(S_j)$.

$\square$

SCC Graph

Henceforth, we arrange $S_1, S_2, ..., S_t$ such that

$$label(S_1) > label(S_2) > ... > label(S_t).$$

Lemma 3: Fix any $i \in [1, t]$. Consider any vertex $u \in S_i$. In G^{rev} (i.e., the reverse graph), if (v, u) is an incoming edge of u and yet $v \notin S_i$, then v belongs to some S_j with $j > i$.

Proof: As (v, u) is in G^{rev}, G has an edge from u to v. Hence, S_i has an edge to S_j in G^{scc}.

By Lemma 2, $label(S_i) > label(S_j)$, which means $j > i$. □

Correctness

Lemma 4: Consider the DFS on G^{rev} (in Step 3 of our algorithm). For each $i \in [1, t]$, S_i is exactly the set of vertices in the i-th DFS-tree produced.

Proof: We will prove the claim by induction on i.

Consider $i = 1$. Let u be the vertex in S_1 having the largest label; u is the root of the first DFS-tree. Consider the beginning moment of the first DFS on G^{rev}.

- As S_1 is an SCC, u has a white path to every other vertex in S_1.
- By Lemma 3, u has no white path to any vertex outside S_1.

By the white path theorem, all and only the vertices in S_1 are descendants of u in the first DFS tree. The claim thus holds for $i = 1$.

Correctness

Proof (cont.): Assuming that the claim holds for $i = k - 1$ (where $k \geq 2$), next we prove its correctness for $i = k$. Let u be the vertex in S_k having the largest label; u is the root of the k-th DFS-tree. Consider the beginning moment of the k-th DFS on G^{rev}.

- All the vertices in $S_1, S_2, ..., S_{k-1}$ are black.
- As S_k is an SCC, u has a white path to every other vertex in S_k.
- By Lemma 3, u has no white path to any vertex in $S_{k+1}, S_{k+2}, ..., S_t$.

By the white path theorem, all and only the vertices in S_k are descendants of u in the k-th DFS tree. The claim thus holds for $i = k$. $\square$