Strongly Connected Components

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

Write $u \rightsquigarrow v$ if there is a path from $u$ to $v$ in $G$

Write $u \sim v$ if $u \rightsquigarrow v$ and $v \rightsquigarrow u$

$\sim$ is an equivalence relation:

- $u \sim u$
- $u \sim v$ implies $v \sim u$
- $u \sim v$ and $v \sim w$ implies $u \sim w$

$\sim$’s equivalence classes are called the strongly connected components (SCC’s) of $G$

For $v \in V$, $C(v) := v$’s SCC
The component graph

Idea: collapse each SCC’s into a single node

Formally: component graph $G^{\text{SCC}} = (V^{\text{SCC}}, E^{\text{SCC}})$

$V^{\text{SCC}} = \text{the SCC's } C_1, \ldots, C_k \text{ of } G$

$E^{\text{SCC}} = \{(C_i, C_j) : i \neq j, (u, v) \in E \text{ for some } u \in C_i, v \in C_j\}$
Lemma 1. \( u \sim v \) in \( G \iff C(u) \sim C(v) \) in \( G^{SCC} \)
Lemma 2. $G^{scc}$ is acyclic.

- Suppose there is a cycle.
- By definition, no self loops in $G^{scc}$, so the cycle must contain two distinct nodes, say $C(u)$ and $C(v)$
- Then we have $C(u) \rightsquigarrow C(v)$ and $C(v) \rightsquigarrow C(u)$ in $G^{scc}$
- By Lemma 1, $u \rightsquigarrow v$ and $v \rightsquigarrow u$ in $G$
- Thus, $C(u) = C(v) \Rightarrow \Leftarrow$
- QED
An application

Scheduling with constraints:

- We want to schedule a set of tasks
- Each task is represented by a node in a directed graph $G$
- Edges in $G$ represent scheduling constraints: if
  - $v$ and $w$ are distinct tasks,
  - there is a path from $v$ to $w$ in $G$, and
  - both $v$ and $w$ are performed,
  then $v$ must be performed before $w$
- Each task has a profit associated with it: we want to schedule tasks to maximize profit
A solution:
1. Compute component graph, topologically sorted
2. For each SCC, select the task with maximum profit
3. Output the tasks from Step 2 in the topological order

Profits: $a = 1$, $b = 2$, $c = 3$, etc.

Optimal schedule: $e, d, g, h$
Special case: $G$ is undirected

$$(u, v) \in E \iff (v, u) \in E$$

SCC’s are just called *connected components*

The component graph consists of isolated nodes — no edges between components

Easy to compute: the trees in the DFS forest are the connected components
Computing SCC’s

For a graph $G$, let $G^T$ denote its “transpose” or “reverse” — same as $G$ but with all edges reversed.

$G$ and $G^T$ have the same SCC’s — in fact, $(G^T)^{scc} = (G^{scc})^T$.

Algorithm $SCC(G)$:

1. call $DFS(G)$, and order the nodes $v_1, \ldots, v_n$ in order of decreasing finishing time (as in $TopSort$).

2. compute $G^T$.

3. call $DFS(G^T)$ — but in the top-level loop, process in the order $v_1, \ldots, v_n$.

   the trees in the DFS forest are the SCC’s of $G$.

Running time: $O(|V| + |E|)$
Example:

\[ G : \]

\[ G^T : \]

\[ G^{SCC} : \]
**Notation:** let $f[u]$ be the finish time in the *first* DFS, and let $f(U) := \max\{f[u] : u \in U\}$

**Lemma 3.** Suppose $(C, C') \in E^{sc}$. Then $f(C) > f(C')$

**Proof.** In the first DFS, let $x$ be the first node discovered in $C \cup C'$

**Case 1:** $x \in C$

By the White Path Theorem, all nodes in $C \cup C'$ are descendents of $x$ in the DFS forest

By the Parenthesis Theorem, $f[x] = f(C) > f(C')$
Case 2: \(x \in C'\)

By the White Path Theorem, all nodes in \(C'\) are descendents of \(x\) in the DFS forest.

By Lemma 2, there is no path from \(C'\) to \(C\) in \(G^{scc}\), and so no node in \(C\) is reachable from \(x\) so at time \(f[x]\), all nodes in \(C\) are still white.

\[
\therefore f(C) > f[x] = f(C'). \text{ QED}
\]

**Corollary.** Let \(C, C' \in V^{scc}\), with \(C \neq C'\). If \((u, v) \in E^T\), with \(u \in C\) and \(v \in C'\), then \(f(C) < f(C')\).
Theorem. Algorithm SCC is correct.

Proof. Let $T_1, \ldots, T_\ell$ be the trees of the DFS forest created in step 3
Let $C_1, \ldots, C_k$ be the SCC’s, with $f(C_i) > f(C_{i+1})$

At step 3, we start with a vertex $x_1$ in $C_1$
By White Path Theorem, all nodes in $C_1$ will be in $T_1$
By Corollary, in $G^T$, there are no edges leaving $C_1$
∴ the nodes of $C_1$ are exactly the nodes of $T_1$
Next, we pick a node in $C_2$, and at this time, all nodes in $C_1$ are black, and all nodes in $C_2, \ldots, C_k$ are white.

By White Path Theorem, $T_2$ contains all nodes in $C_2$, and by Corollary, $T_2$ contains no other nodes.

∴ the nodes of $C_2$ are exactly the nodes of $T_2$.

Proceeding by induction, we get $T_i = C_i$ for $i = 1, \ldots, \ell$, and so $k = \ell$. QED
Representation of $G^{\text{scc}}$

- Let $C_1, \ldots, C_k$ be the SCC's
- Number the nodes $1 \ldots k$
- Standard adjacency list representation of $G^{\text{scc}}$
- Also:
  - An array mapping $v \in V$ to $j \in \{1, \ldots, k\}$, where $v \in C_j$
  - An array mapping $j \in \{1, \ldots, k\}$ to a list representation of $C_j$
- This can all be done in time $O(|V| + |E|)$, and we may assume that $C_1, \ldots, C_k$ are already in topological order — in fact Algorithm SCC outputs $C_1, \ldots, C_k$ in topological order
Another application

**Problem:** A graph $G = (V, E)$ is called *semi-connected* if for all $u, v \in V$, $u \rightsquigarrow v$ or $v \rightsquigarrow u$.

Show how to test if $G$ is semi-connected in time $O(|V| + |E|)$.