DFS
Strongly Connected Components
Strongly Connected Components

- $G$ is strongly connected if every pair $(u, v)$ of vertices in $G$ is reachable from one another.
- A **strongly connected component** (SCC) of $G$ is a maximal set of vertices $C \subseteq V$ such that for all $u, v \in C$, both $u \sim v$ and $v \sim u$ exist.
Component Graph

- $G^{SCC} = (V^{SCC}, E^{SCC})$.
- $V^{SCC}$ has one vertex for each SCC in $G$.
- $E^{SCC}$ has an edge if there’s an edge between the corresponding SCC’s in $G$.
- $G^{SCC}$ for the example considered:

![Graph Diagram]

graphs-2 - 3
$G^{	ext{SCC}}$ is a DAG

**Lemma 1**
Let $C$ and $C'$ be distinct SCC's in $G$, let $u, v \in C$, $u', v' \in C'$, and suppose there is a path $u \sim u'$ in $G$. Then there cannot also be a path $v' \sim v$ in $G$.

**Proof:**
- Suppose there is a path $v' \sim v$ in $G$.
- Then there are paths $u \sim u' \sim v'$ and $v' \sim v \sim u$ in $G$.
- Therefore, $u$ and $v'$ are reachable from each other, so they are not in separate SCC's.
Simple SCC Algorithms

- Computing the **transitive closure** of the adjacency matrix of the graph.

- We want to compute a sequence of $n = |V|$ matrices.
- For matrix $A_k \ (1 \leq k \leq n)$:
  Entry $i,j$ indicates whether or not there is a path between vertex $i$ and $j$ going only through vertices from the set $\{1,2,\ldots,k\}$.

- The adjacency matrix ($A_0$) tells us if there is a single edge path between $i$ and $j$ ($= a$ path not going through any other vertices).
- The last matrix in the series, $A_n$, should tell us if there is any path between $i$ and $j$. 
Suppose we have computed $A_0, A_1, \ldots, A_{k-1}$. We can now compute

$$A_k[i,j] = A_{k-1}[i,j] \text{ or } (A_{k-1}[i,k] \text{ and } A_{k-1}[k,j])$$

$\Rightarrow$ one way to implement this algorithms is:

$|V|$ passes, each pass makes one check for each pair of vertices ($|V|^2$).

$\Rightarrow$ Total Running Time: $\Theta(|V|^3)$.
Transpose of a Directed Graph

- $G^T = \text{transpose}$ of directed $G$.
  - $G^T = (V, E^T), E^T = \{(u, v) : (v, u) \in E\}$. 
  - $G^T$ is $G$ with all edges reversed.
- Can create $G^T$ in $\Theta(V + E)$ time.
- $G$ and $G^T$ have the same SCC’s. ($u$ and $v$ are reachable from each other in $G$ if and only if reachable from each other in $G^T$.)

Algorithm to determine SCCs

\( \text{SCC}(G) \)
1. call \( \text{DFS}(G) \) to compute finishing times \( f[u] \) for all \( u \)
2. compute \( G^T \)
3. call \( \text{DFS}(G^T) \), but in the main loop, consider vertices in order of decreasing \( f[u] \) (as computed in first DFS)
4. output the vertices in each tree of the depth-first forest formed in second DFS as a separate SCC

**Time:** \( \Theta(V + E) \).
Example

\( G \)

a \(\rightarrow\) b

13/14 \(\rightarrow\) 11/16

e \(\rightarrow\) 12/15

f \(\rightarrow\) 3/4

g \(\rightarrow\) 2/7

h

c \(\rightarrow\) 1/10

d \(\rightarrow\) 8/9

h
Example

$G^T$

Graph with nodes labeled a, b, c, d, e, f, g, h.
Example
How does it work?

Intuition for proof:
- In the 1st DFS, for every tree root, we find all the vertices it can reach.
- In the 2nd DFS, we make the same vertices be tree roots (by choosing highest numbered vertices to start with).
- In the 2nd DFS, we find all the vertices a root can reach in $G^T$; $\Rightarrow$ these are vertices that can reach the root in $G$.
- $\Rightarrow$ all these vertices can reach each other (through the root).
- $\Rightarrow$ each tree contains all vertices that can reach each other (= SCC).

- **Notation:**
  - $d[u]$ and $f[u]$ always refer to first DFS.
  - Extend notation for $d$ and $f$ to sets of vertices $U \subseteq V$:
    - $d(U) = \min_{u \in U} \{d[u]\}$ (earliest discovery time)
    - $f(U) = \max_{u \in U} \{f[u]\}$ (latest finishing time)
SCCs and DFS finishing times

Lemma 2
Let $C$ and $C'$ be distinct SCC’s in $G = (V, E)$. Suppose there is an edge $(u, v) \in E$ such that $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Proof:
- **Case 1: $d(C) < d(C')$**
  - Let $x$ be the first vertex discovered in $C$.
  - At time $d[x]$, all vertices in $C$ and $C'$ are not yet discovered.
  - There exist paths from $x$ to all vertices in $C$ and $C'$.
  - All vertices in $C$ and $C'$ are descendants of $x$ in DFS tree.
  - $f[x] = f(C) > f(C')$. 
SCCs and DFS finishing times

Lemma 2
Let $C$ and $C'$ be distinct SCC's in $G = (V, E)$. Suppose there is an edge $(u, v) \in E$ such that $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Proof:

- Case 2: $d(C) > d(C')$
  - Let $y$ be the first vertex discovered in $C'$.
  - At time $d[y]$, all vertices in $C$ and $C'$ are not there yet discovered.
  - There is a path from $y$ to each vertex in $C' \Rightarrow$ all vertices in $C'$ become descendants of $y \Rightarrow f[y] = f(C')$.
  - By lemma 1, since there is an edge $(u, v)$, there is no path from $C'$ to $C$.
  - No vertex in $C$ is reachable from $y$.
  - Thus, at time $f[y]$, all vertices in $C$ are still not discovered.
  - Therefore, for all $w \in C, f[w] > f[y]$, which implies that $f(C) > f(C')$. 

![](image.png)
SCCs and DFS finishing times

**Corollary 1**
Let $C$ and $C'$ be distinct SCC’s in $G = (V, E)$. Suppose there is an edge $(v, u) \in E^T$, where $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

**Proof:**
- $(v, u) \in E^T \Rightarrow (u, v) \in E$.
- Since SCC’s of $G$ and $G^T$ are the same, $f(C') < f(C)$, by Lemma 2.
Correctness of SCC

- When we do the second DFS, on $G^T$, start with SCC $C$ such that $f(C)$ is maximum.
  - The second DFS starts from some $x \in C$, and it visits all vertices in $C$.
  - Corollary 1 says that since $f(C) > f(C')$ for all $C \neq C'$, there are no edges from $C$ to $C'$ in $G^T$.
  - Therefore, DFS will visit *only* vertices in $C$.
  - Which means that the DFS tree rooted at $x$ contains exactly the vertices of $C$. 
The next root chosen in the second DFS is in SCC $C'$ such that $f(C')$ is maximum over all SCC’s other than $C$.

- DFS visits all vertices in $C$, but the only edges out of $C'$ go to $C$, which we’ve already visited.
- Therefore, the only tree edges will be to vertices in $C'$.

We can continue the process.

Each time we choose a root for the second DFS, it can reach only:
- vertices in its SCC (through tree edges).
- vertices in SCC’s already visited in second DFS (but not through tree edges).