STRONGLY CONNECTED GRAPHS

(directed) graph s.t. every vertex can be reached from every vertex.
(directed) graph s.t. every vertex can be reached from every vertex.

\[ \begin{cases} 
\text{e.g., 2 vertices can reach each other on a subgraph that is a simple cycle} \\
\text{using edges once} 
\end{cases} \]
(directed) graph s.t. every vertex can be reached from every vertex.
(directed) graph s.t. every vertex can be reached from every vertex.

$$\{\text{Not necessarily true that every 2 vertices reach each other with a simple cycle.}\}$$
STRONGLY CONNECTED COMPONENTS

- No longer strongly connected
- Chain reaction: several vertices no longer mutually reachable
- We get groups (components) each of which is strongly connected
The new graph must be a DAG (any cycle would merge components)
RECOGNIZING
STRONGLY CONNECTED COMPONENTS

Part 1: useful properties
Consider any two SCC: A & B

CASE 1: either they're unrelated
CASE 2: or precisely one links to the other
Which vertices will finish first in a DFS?

If CASE 1, whichever is explored first finishes first.

If CASE 2:
Either B gets explored first (and it can't find A)
or A gets explored first, so some a_i will find all B before finishing

Result: \{all A\} finishes after \{all B\}
or \{some of A\} finishes after \{all B\} after \{some of A\}
Consider any two SCC: A & B

**CASE 1:** either they're unrelated

**CASE 2:** or precisely one links to the other

Which vertices will finish first in a DFS?

Examples for **CASE 2**

1. $a_4 \ a_2 \ a_3 \ a_1 \ x \ b_1 \ b_2 \ b_3 \rightarrow$ DFS started at $x$ or $b_1$ or $a_4 \rightarrow a_2 \rightarrow x$
2. $a_2 \ a_3 \ a_1 \ a_4 \ x \ b_1 \ b_2 \ b_3 \rightarrow$ DFS started at $x$ or $b_1$ or $a_2 \rightarrow x$
3. $a_3 \ a_4 \ a_2 \ a_1 \ x \ b_2 \ b_3 \ b_1 \rightarrow$ DFS started at $b_2$, then $a_3 \rightarrow a_4 \rightarrow a_2$
4. $a_2 \ a_1 \ x \ b_1 \ b_2 \ b_3 \ a_3 \ a_4 \rightarrow$ DFS started at $a_2 \rightarrow a_3$
DFS finishing times (focusing on $A \Rightarrow B$)

\{all $A$\} ... \{all $B$\}

(same if $A, B$ unrelated)

w.l.o.g.

or \{some of $A$\} ... \{all $B$\} ... \{some of $A$\}

\[ \text{not without going via other } A \text{ that finished after } B \]

: nested groups

quasi-topologically sorted

(backwards arrows only within SCC)
Finding all strongly connected components

Part 2: algorithm

what we see

what we want
We don't know any of these groups yet

DFS(G)
finishing times

To us, \( a_2, a, x, b, b_1, b_2, c, b_3, d, a_3, a_4, e \)
looks like \( v_9, v_1, v_4, v_11, v_8, v_2, v_3, v_7, v_5, v_{10}, v_6 \)

: nested groups
quasi-topologically sorted
(back→arrows only within SCC)
Run DFS on $G^T$, processing vertices in order of finishing times obtained in DFS(G)

- Start DFS($G^T$) on some $a_i$ (whatever finished last)
- Only $A$ can link to $a_i$ in $G$

$\Rightarrow a_i$ will find $A$ and nothing else in $G^T$

- DFS($G^T$) will continue on next unmarked vertex $\Rightarrow$ some $x_i$: finds only $X$ ... etc

: nested groups

quasi-topologically sorted

(back\-arrows only within SCC)
Run DFS on $G^T$, processing vertices in order of finishing times obtained in DFS($G$)

- Start DFS($G^T$) on some $a_i$ (whatever finished last)
- Only $A$ can link to $a_i$ in $G$
  - $a_i$ will find $A$ and nothing else in $G^T$
- DFS($G^T$) will continue on next unmarked vertex $\Rightarrow$ some $x_i$ finds only $x$

- nested groups
  - quasi-topologically sorted
- (back←arrows only within SCC)
Run DFS on $G^T$, processing vertices in order of finishing times obtained in DFS($G$)

- Start DFS($G^T$) on some $a_i$ (whatever finished last)
- Only $A$ can link to $a_i$ in $G$.
  \[ a_i \] will find $A$ and nothing else in $G^T$
- DFS($G^T$) will continue on next unmarked vertex
  \[ \ldots \text{some } x_i : \text{ finds only } X \ldots \text{ etc} \]

DFS($G$) finishing times

\[ A \times B \times C \times B \times D \times A \times E \]

\[ a_2 a_1 \quad x \quad b_1 b_2 \quad c \quad b_3 \quad d \quad a_3 a_4 \quad e \]

: nested groups
quasi-topologically sorted
(back\leftarrow-arrows only within SCC)
Run DFS on $G^T$, processing vertices in order of finishing times obtained in DFS($G$)

- Start DFS($G^T$) on some $a_i$ (whatever finished last)
- Only $A$ can link to $a_i$ in $G$
  - $a_i$ will find $A$ and nothing else in $G^T$
- DFS($G^T$) will continue on next unmarked vertex $\Rightarrow$ some $X_i$ finds only $X_{...}$ etc.

DFS($G$)
finishing times

$A \times XBCBDAE$

$\underline{a_2a_1} \times b_1b_2c b_3d a_3a_4e$

quasi-topologically sorted (back-arrows only within SCC)
Run DFS on $G^T$, processing vertices in order of finishing times obtained in DFS($G$)

- Start DFS($G^T$) on some $a_i$ (whatever finished last)
- Only $A$ can link to $a_i$ in $G$
  - $a_i$ will find $A$ and nothing else in $G^T$
- DFS($G^T$) will continue on next unmarked vertex
  - some $X_i$ finds only $X$

: nested groups
quasi-topologically sorted
(back-arrows only within SCC)
Finding Strongly Connected Components

(Another example)
DFS from arbitrary vertex

DFS sequence:
1 → 2 → 8 → 3 → 5
3 → 4
8 → 10 → 9 → 12
2
1 → 6 → 11
7
finished
Finishing times: 7 1 6 11 2 8 10 9 12 3 4 5

DFS(G^T) in this order finished first

DFS from arbitrary vertex
Correctness

Claim:
When processing a vertex $u$ (e.g. 2) in $\text{DFS}(G^T)$, such that $u$ is the leftmost vertex of $\text{SCC}(u)$, $u$ will find all of $\text{SCC}(u)$ and nothing else.

Proof:

part 1: By induction, $\text{SCC}(u)$ will be unmarked

\[ \text{nothing left of } u \text{ has marked } \text{SCC}(u) \]

part 2: Suppose $\exists$ edge $\text{SCC}(u) \rightarrow x$ in $G^T$, s.t. $x \not\in \text{SCC}(u)$, unmarked, & right of $u$.

Then, $\exists$ edge from $x$ to $\text{SCC}(u)$ in $G$.

A path from $\text{SCC}(u)$ to $x$ in $G$ [otherwise $x \in \text{SCC}(u)$, contradiction]

Then in $\text{DFS}(G)$ $x$ will finish after $u$ [ $x$ left of $u$: contradiction]

$G$ trivial IF $\text{SCC}(u)$ found first; ELSE: $x$ not done until it finds $\text{SCC}(u)$