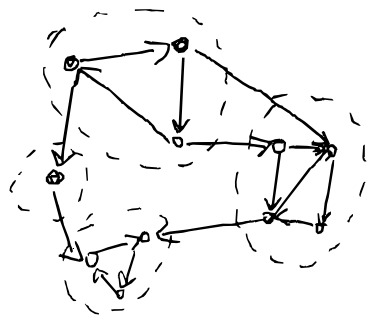
Strongly Connected Components

Connected Components in undirect graphs:
A connect component is a maximal set of vertices that can reach each other.
Can be computed using DFS/BFS in $O(m)$ time.

Strongly connected components in directed graphs (SCC)

A strongly connected component is a maximal set of vertices that can reach each other via directed paths.



Today: SCC in $O(m)$ time.

View each SCC as a supernode, the edges between the SCCs as the edges, denoted this graph by G'

Claim: G' contains no cycles. (G' is a DAG.)

Proof. O.w. a cycle makes vertices in different SCC reachable from each other. Contradiction

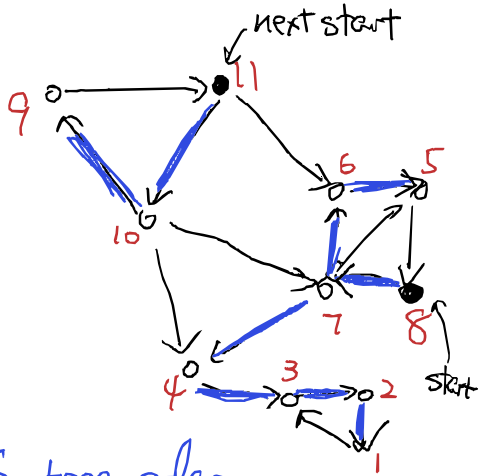
Claim $\Rightarrow$ The SCCs have a topological order.

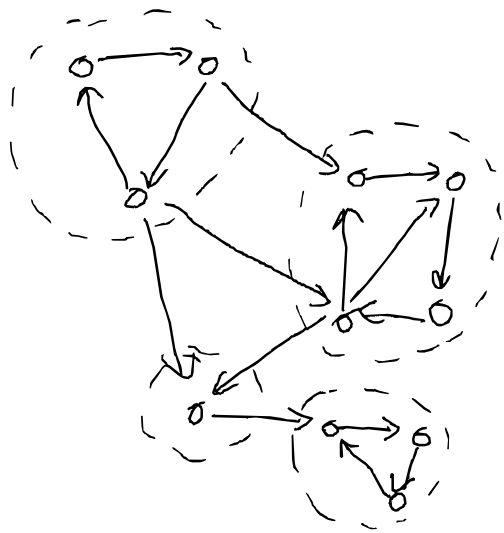
If we could DFS/BFS from some vertex in C_5 and then some in C_4, C_3, C_2, C_1. then we're done.

Run DFS twice

- Repeat
  - find any unvisited vertex u, DFS(u)
- Until all vertices are visited

- List all vertices by the time that DFS exits
  - let $v_1, v_2, \dots, v_n$ be the order from last exit to first exit
- Mark all vertices as unvisited
- For all vertices u, in the order of $(v_1, \dots, v_n)$
  - If u is unvisited,
    - DFS(u) with <u>all edge direction reversed</u>
    - the set of vertice visited during DFS(u) forms an SCC.





DFS tree edges

exit order

Running time : $O(m)$ (two DFS)

Correctness: Prove by induction on #DFS(u) calls for the second DFS.

Induction hypothesis: each of the first i calls identifies an SCC.

Consider the next call DFS(u), for $u = v_k$.

Suppose C be the SCC that contains u.

Lemma 1: DFS(u) visits all vertices in C.

Lemma 2: DFS(u) doesn't visit any vertex not in C.

Proof of Lemma 1: By induction hypothesis, no $x \in C$ was visited in a prior DFS call. u can reach all $x \in C$ via edges with reversed

direction (x and u are in the same SCC). DFS(u) visits all $x \in C$ in this call. □

Proof of Lemma 2:

Let x be a vertex visited by the reversed-edge DFS from $u = v_k$.

- Suppose $x = v_l$, then l must be larger than k, i.e., first DFS leaves x earlier than u.
- There is a reversed-edge path from u to x, i.e., there is a path from x to u.

Want to show: there is a path from u to x.

- When the first DFS leaves x, all nodes reachable from x must have been visited (DFS has at least entered these nodes),

this includes u.

- But DFS leaves u at a later time. DFS(u) has started, but has not returned.

Hence, u is an ancestor of x in the DFS tree. $\exists$ a path from u to x.

Since $\exists$ a path from x to u, x is in C. □

There is also a more sophisticated algorithm that does DFS once.