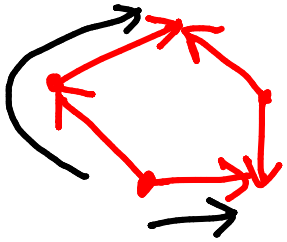
Directed graphs & connectivity

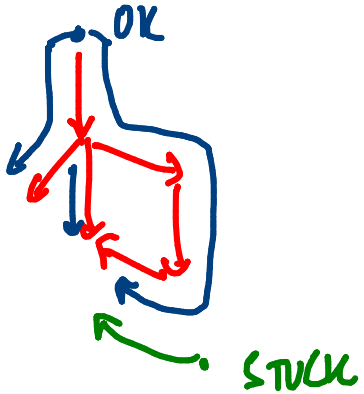
Undirected graphs - BFS/DFS identifie connected components, connected $\stackrel{def}{=}$ one component

What is a good generalization for directed graphs?

Is this "connected"?



No!



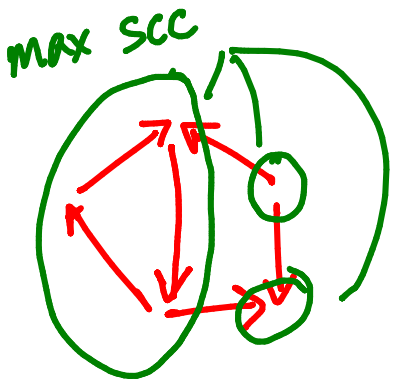
No!

Component: Start anywhere & reach full component

Strongly connected component (scc)

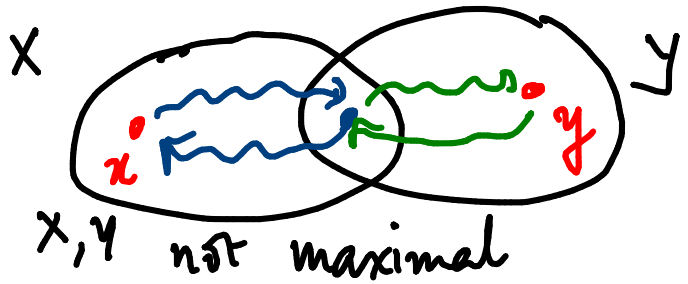
A subset X of V s.t. for any $u, v \in X$ there is a path from u to v and v to u

We are interested in maximal scc's



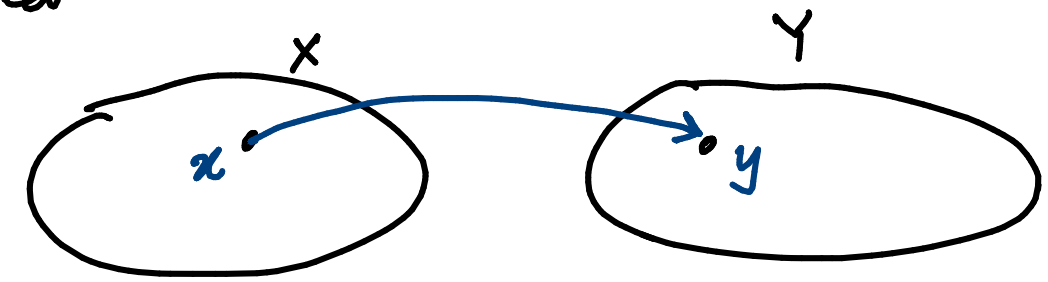
Maximal scc's partition V

- i.e. no two can overlap



Goal: Identify scc's

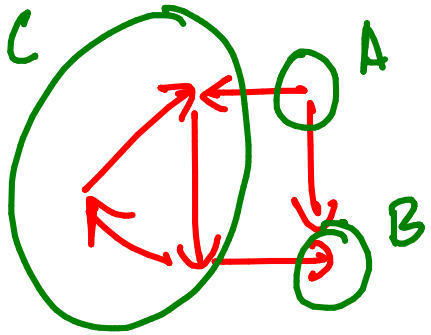
Suppose X, Y are max scc's s.t. they are "connected"

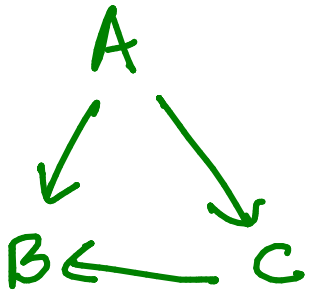


Any connection between X & Y must go from X to Y

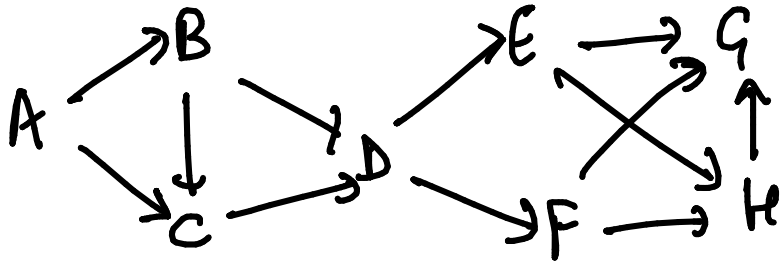
Consider each scc as a vertex

Edges $X \to Y$ if X is connected to Y] DAG





In general



BFS/DFS from $g \in G$ exactly discovers G

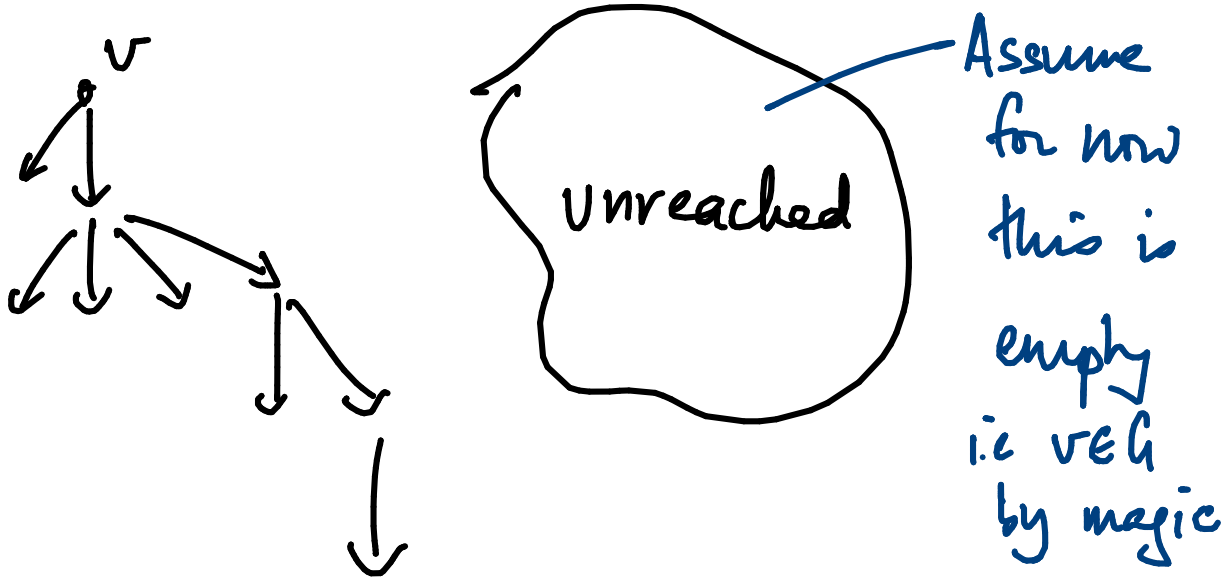
After this, repeat for $h \in H$ – exactly discover H

Process the SCC DAG in reverse topological order

How to process SCC DAG in reverse order?

Reverse all edges in original graph

Run DFS(v) for some $v \in V$



Entry(u), Exit(u) for each u - Exit(v) is maximal

Go back to original graph

Start a DFS at max exit no. → explore "tail" scc

Cannot leave G

Consider highest unmarked exit(u)

This must explore H

etc.

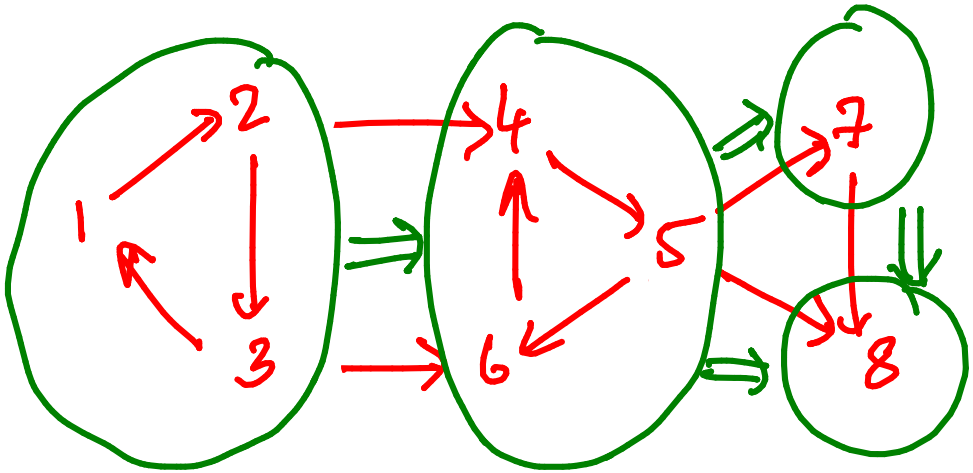
Algorithm — Dasgupta, Papadimitriou & Vazirani

Reverse graph, run DFS, record exit(v) for each $v \in V$

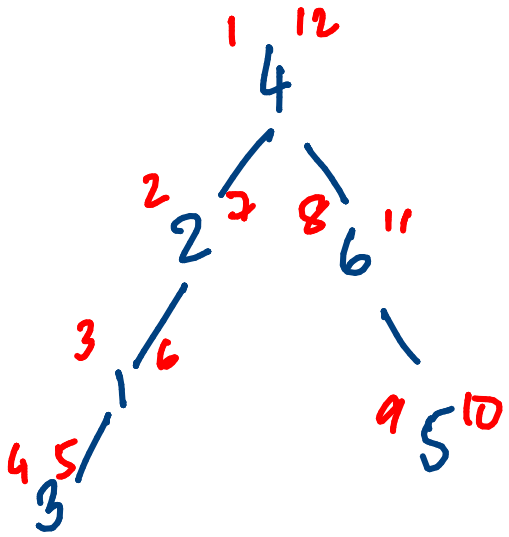
In reverse order of exit(v)

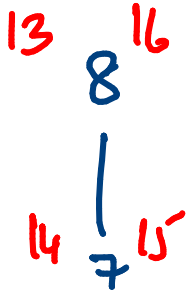
run DFS on original graph

BFS

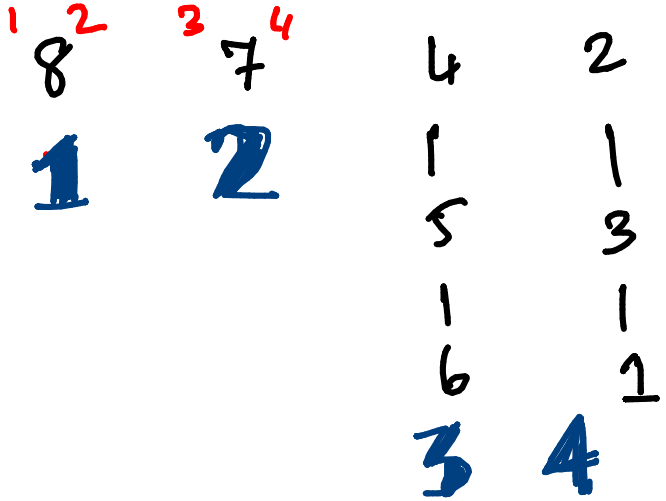


Reverse DFS





8 has max exit(v)

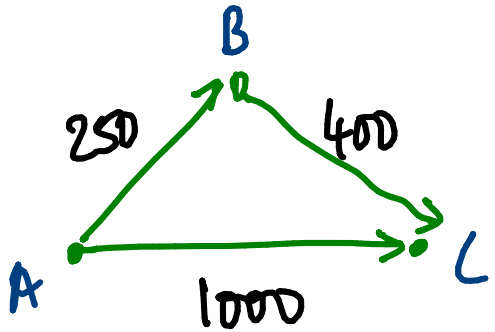


Next

Shortest paths in weighted graphs

Weighted graph - each edge has a "weight"

- cost / distance / time ...



BFS will find this path from A to C