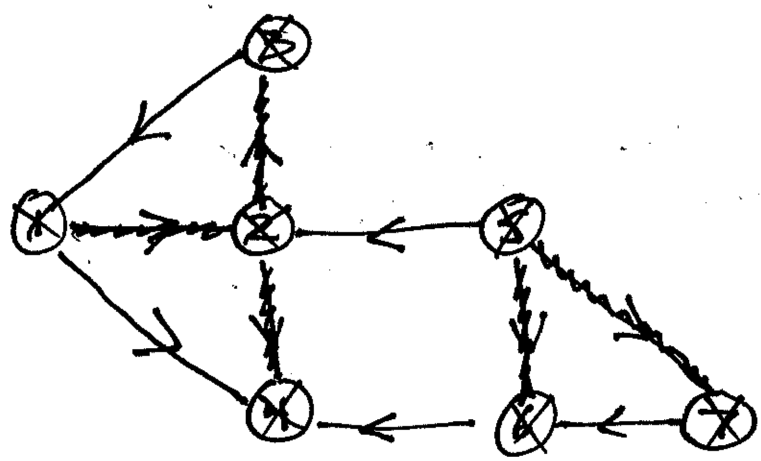
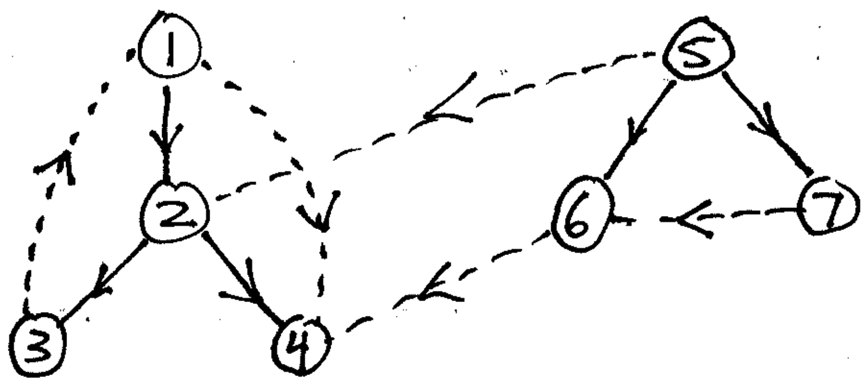
Ex.





tree: (1,2), (2,3), (2,4), (5,6), (5,7)

back: (3,1)

forward: (1,4)

cross: (5,2), (6,4), (7,6)

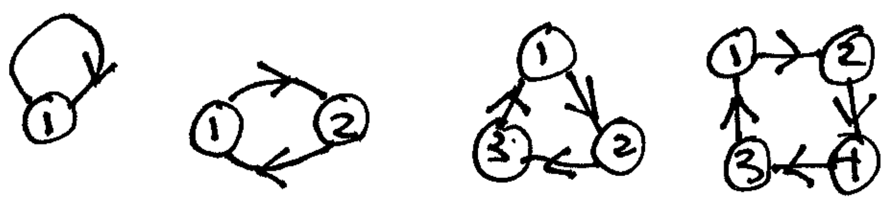
__Exercise:__

modify DFS so it classifies edges as it goes.

See Problem 22.3-12 (3rd ed.) for hints.

__Topological Sort__ (in a digraph)

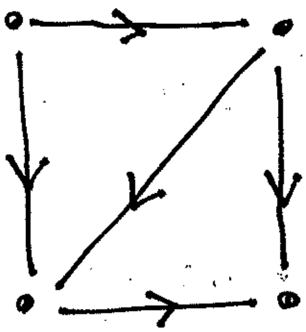
cycles!



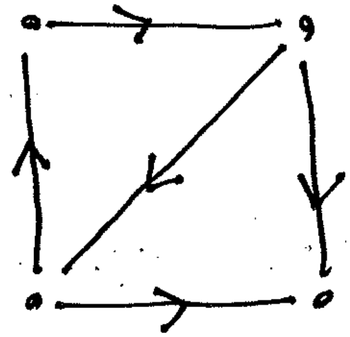
length 1 2 3 4

__Defn__ a digraph G is called __acyclic__ iff it contains no cycles.

Ex.



acyclic



not acyclic

Lemma

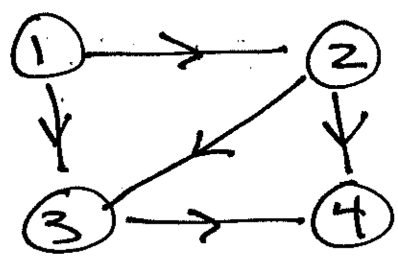
A digraph G is acyclic iff DFS(G) yields no back edges.

We call an acyclic digraph a DAG (directed acyclic graph).

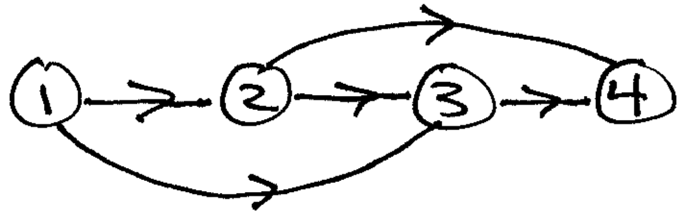
Defn

Let G be a DAG. A <u>topological sort</u> of $V(G)$ is a linear order on $V(G)$ such that if $(x,y) \in E(G)$ then x appears before y in the linear order.

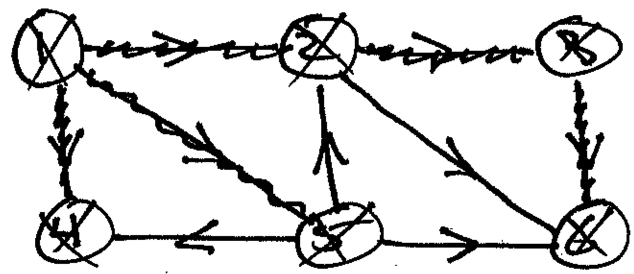
Ex



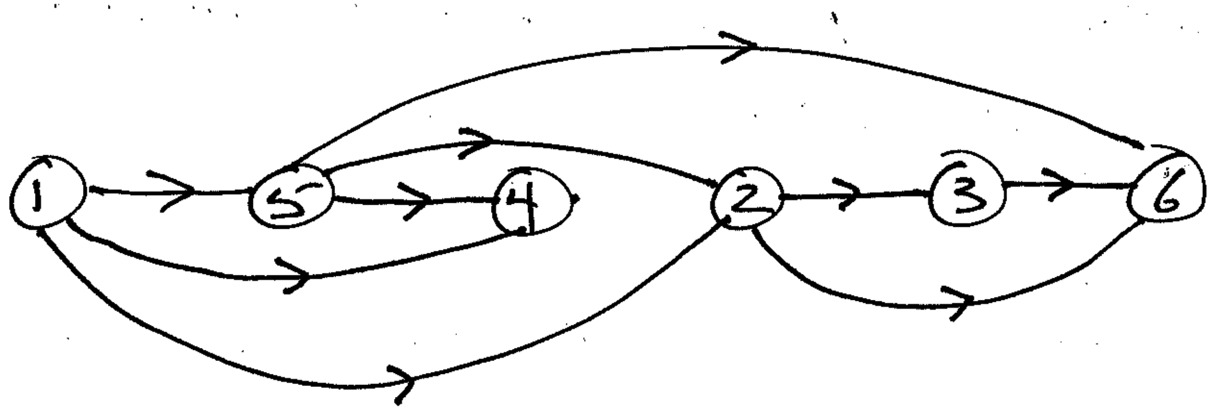
Top. sort



Ex.



Top. sort: (1, 5, 4, 2, 3, 6)



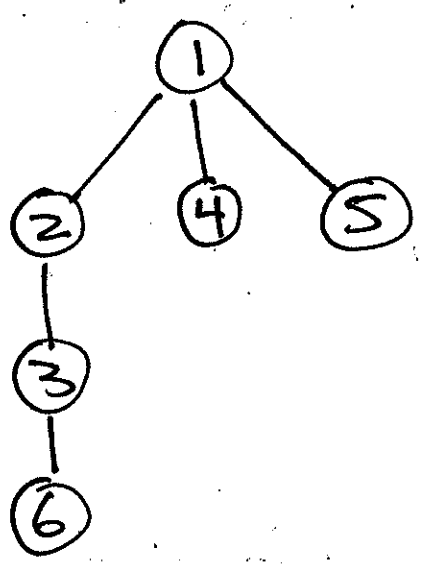
To find a top. sort in a DAG:

- Run DFS(G), as vertices finish push them onto a stack: S

When Done, S (top to bottom) is a topological sort.

Ex.

Forrest!



stack

1
5
4
2
3
6

strongly connected components

Defn Let G be a digraph. We say $y \in V$ is *reachable* from $x \in V$ iff G contains a directed x-y path

Defn

We call G strongly connected iff for all $x, y \in V(G)$: x is reachable from y, and y from x.

Defn

We say $U \subseteq V(G)$ is strongly connected iff for all $x, y \in U$: x is reachable from y and y from x.

Defn.

We call $U \subseteq V(G)$ a

strongly connected component

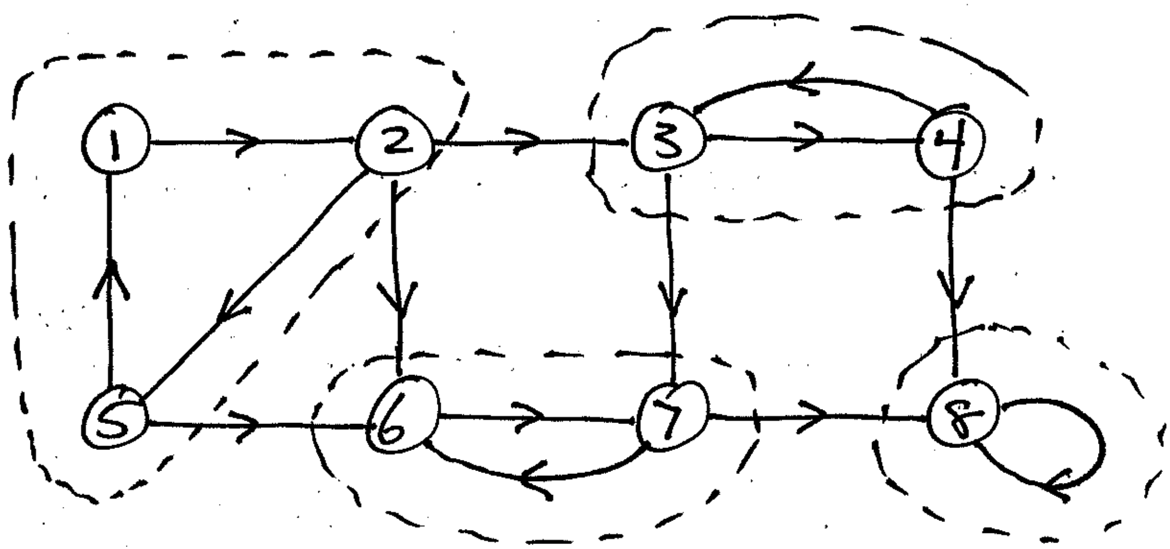
(SCC) of G iff

(1) U is strongly connected

and

(2) U is maximal w.r.t. (1)

Ex.



Problem find the strongly conn. components of a digraph

- Run DFS(G), as vertices finish, push them onto a stack S.
- Compute G^T (G transpose)
- Run DFS(G^T), execute main loop in order given by S.

When done, trees in resulting DFS forrest span the SCCs of G.