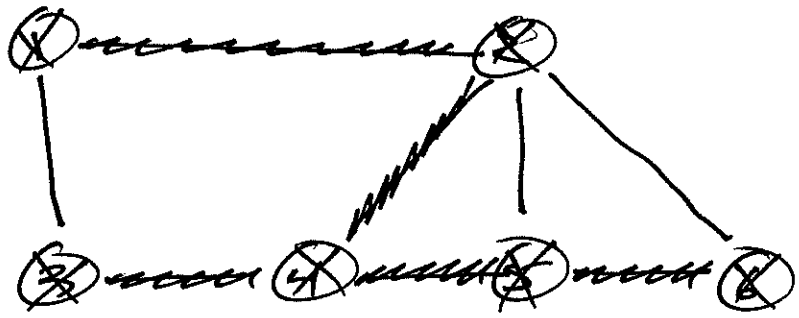
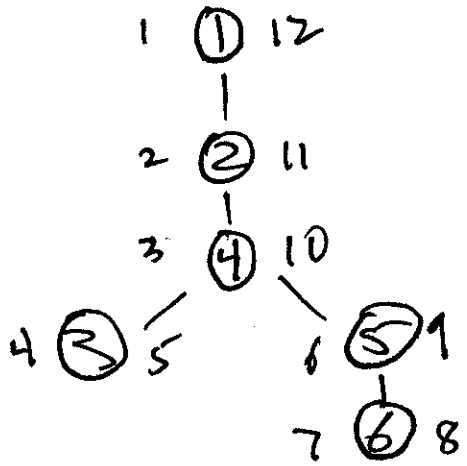
CSE 101-01 4-20-23

Ex. (same)

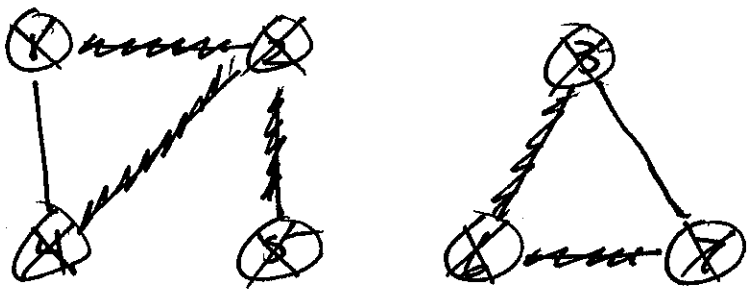


	adj	d	f	p
1	2 3	1	12	N
2	1 4 5 6	2	11	1
3	1 4	4	5	4
4	2 3 5	3	10	2
5	2 4 6	6	9	4
6	2 5	7	8	5

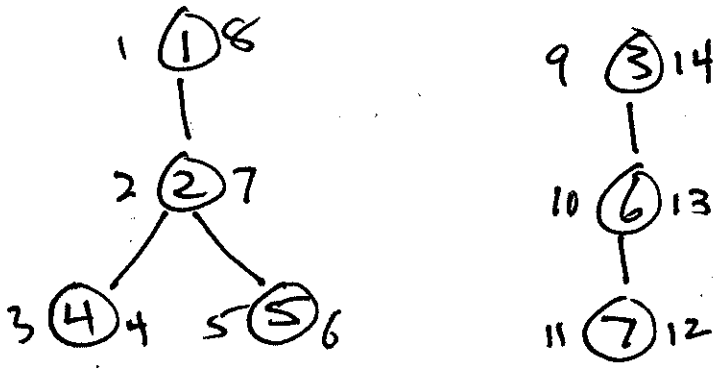
Forest:



Ex.

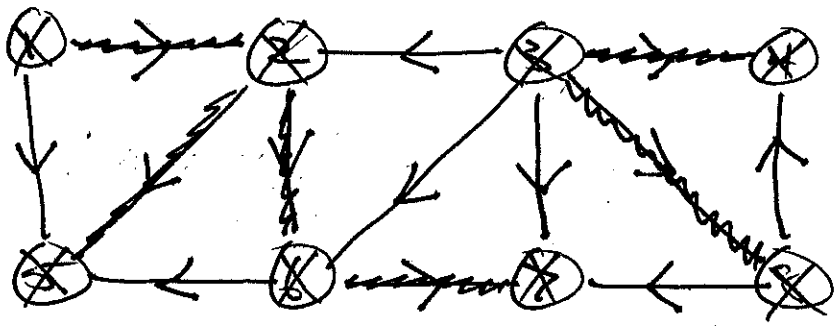


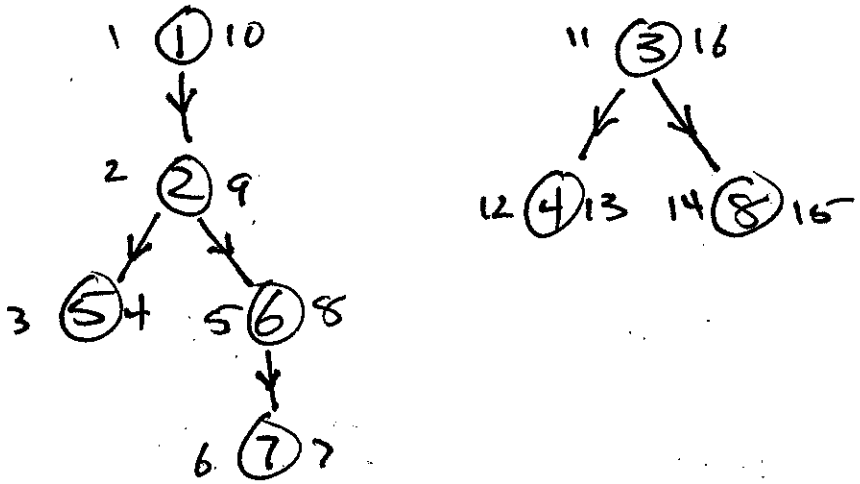
Forest:



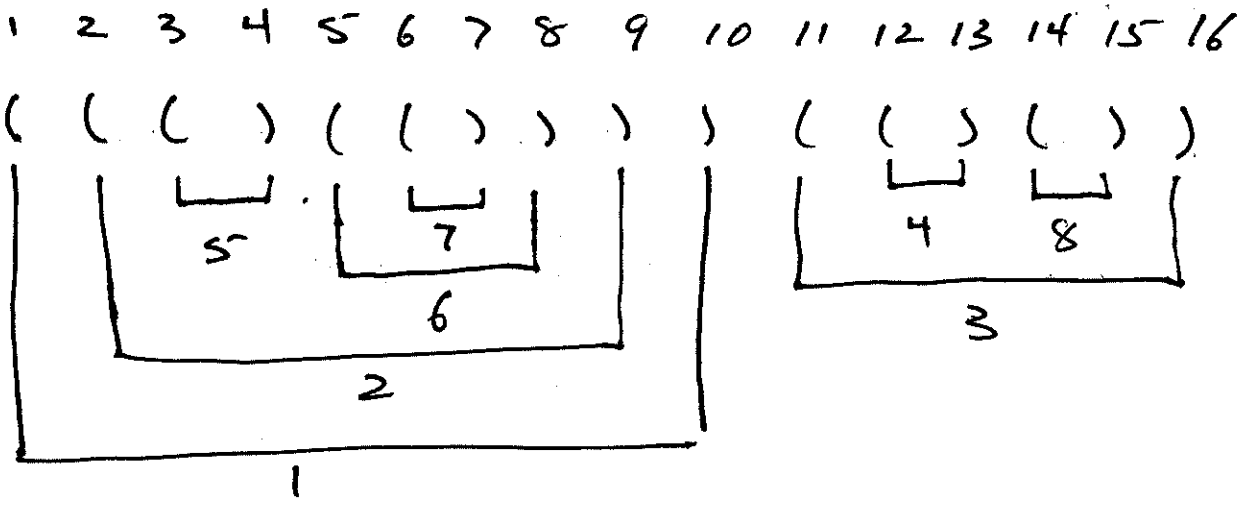
Thm (undirected)
If DFS is run on a graph, then each tree in the DFS forest spans a connected component.

Ex.





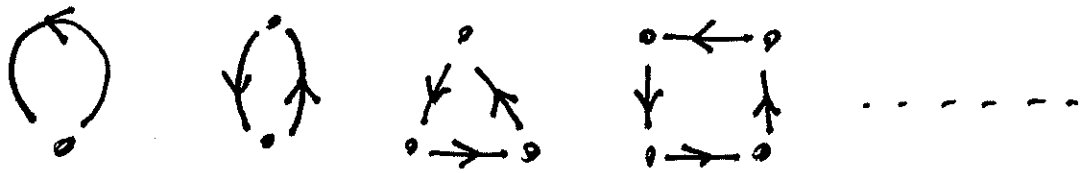
Parenthesis string



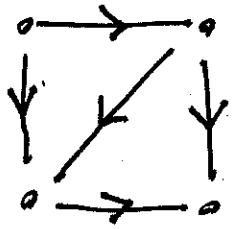
Def^n

Let G be a dig-aph. We say G is **acyclic** (also a DAG) iff G contains no directed cycles.

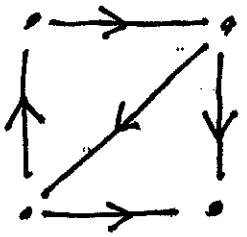
Directed cycles:



Ex.



Acyclic



not acyclic

Edge Classification

Run DFS, then classify edges

Tree: belong to DFS forest

Back: join vertex to an ancestor

Forward: join ancestor to descendant, other than a child

Cross: any other edges

- tree-to-tree
- cousin-to-cousin

Thm

A Digraph G is acyclic iff DFS(G) yields no back edges. Equivalently: G contains a back edge iff it contains a directed cycle.

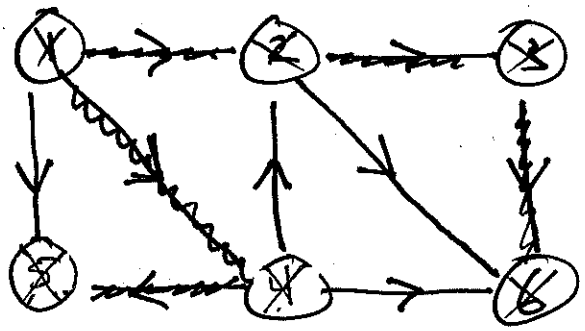
Topological Sort

Let $G = (V, E)$ be a DAG.

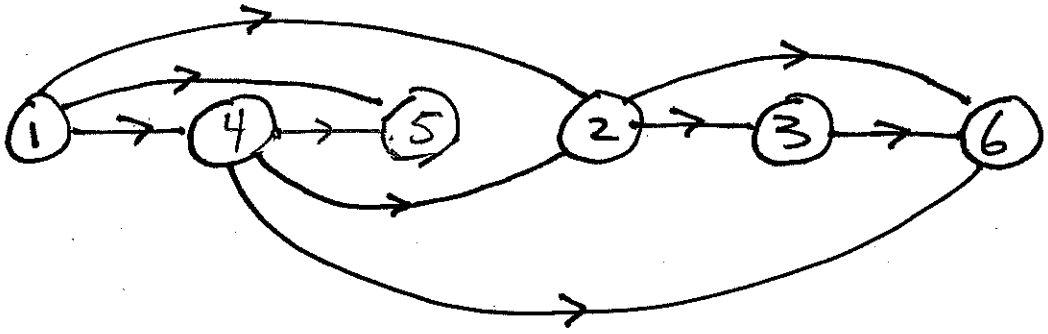
A topological sort of G is a linear ordering of V such that if $(x, y) \in E$, then x is before y in the ordering.

Ex.

G



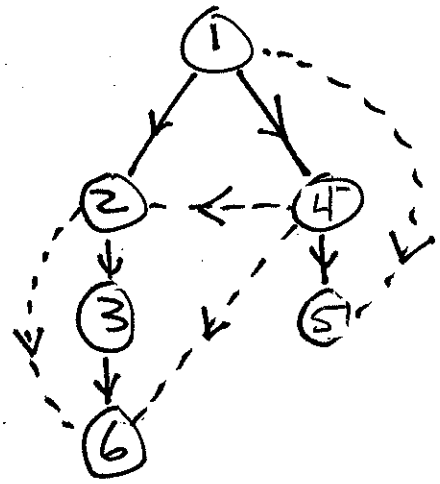
Top. sort $V(G) =$ (1, 4, 5, 2, 3, 6)



Run DFS!

stack

Forest:



1
4
5
2
3
6

Top sort.

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

back:

forward: (1,5), (2,6)

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

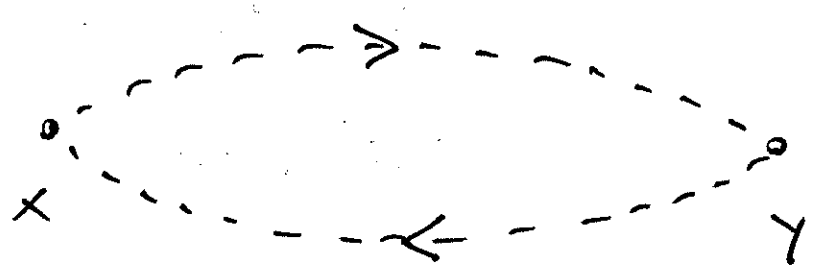
To find a Top. sort:

- run DFS, As vertices, push them onto a stack

Then stack is top. sort.

Strongly connected components

Let $G=(V,E)$ be a digraph. Let $S \subseteq V$. We say S is <u>strongly connected</u> iff for all $x, y \in S$: x is reachable from y and y from x.



If V is strongly connected, we say G is <u>strongly connected</u>.

Let $S \subseteq V$. We say S is a strongly conn. component of G iff

(1) S is strongly connected

and

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

Ex