CSE 101 11-10-20

- quizzes today
- ext. PA3 1 last day
  Wed. 10 PM
- PA4 ext. 2 days

Theorem (White Path)

Let $x, y \in V(G)$. Then y is a descendant of x (so (2) holds in Paren. Thm) if and only if at time $d[x]$, G contains an x-y path consisting entirely of white vertices.

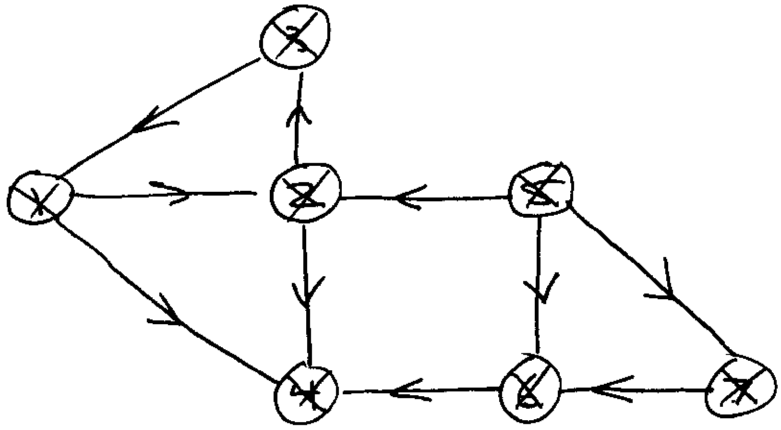
Edge Classification

- Tree Edges: belong to DFS-forest G_p
- Back Edges: Join vertex to an ancestor in G_p
- Forward Edges: join a vertex to a descendant in G_p (other than a child)
- Cross Edges: join either (1) cousin to cousin in same tree, or (2) tree to tree.

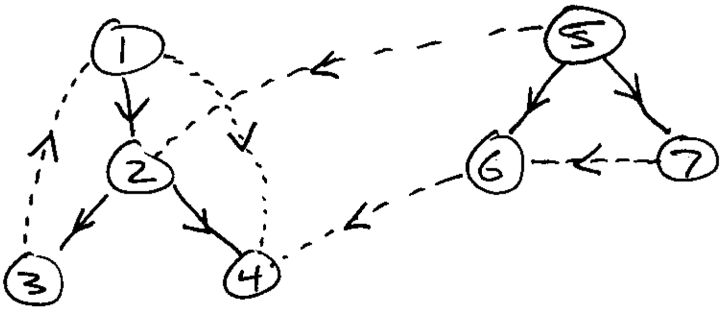
<u>Note</u>: in undirected case, there is no distinction between Back & Forward. all called Back edges

<u>Theorem</u>
In an undir. Graph, there are no cross edges.

Ex.



DFS Forest:



Tree edges: (1, 2), (2, 3), (2, 4) (5, 6), (5, 7)

Back edges: (3, 1)

Forward edges: (1, 4)

cross edges: (7, 6) (cousin - cousin)

(5, 2) (6, 4) (tree - tree)

Exercise

- Run DFS on more examples & do edge classification.

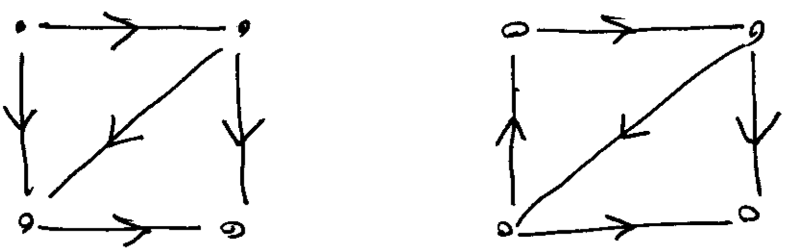
- Modify DFS() & Visit() so as to classify edges as you go, i.e. Print: (x, y) is a ____ edge

tree
back
forward
or cross

23.4 Topological Sort

Defn
A digraph G is <u>acyclic</u> iff it contains no ~~cyc~~ directed cycles.

Ex.



acyclic non-acyclic

Thm
A digraph contains a directed cycle if and only if DFS produces a back edge

i.e. $\boxed{\text{non-acyclic iff back edge}}$

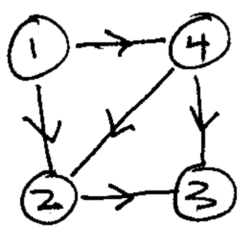
i.e. $\boxed{\text{acyclic iff no back edge}}$

A digraph that is acyclic is called a <u>Directed Acyclic Graph</u> (DAG).

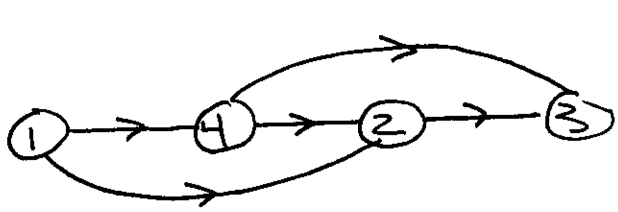
<u>Defn</u>

A <u>Topological sort</u> of $V(G)$, where G is a DAG, is a linear ordering of $V(G)$ such that if $(x,y) \in E$, then x appears before y in the linear ordering.

<u>Ex.</u>



G



TOP. sort of $V(G)$

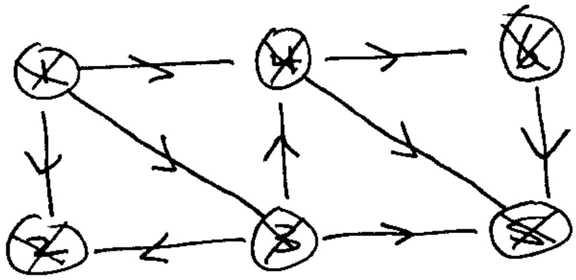
(p. 613 in CLRS)

To find a Top. Sort of a DAG.

- Call DFS(G)
- as vertices finish, push them onto a stack
- when DFS is complete, stack gives the Top. sort (top to bottom.)

  equivalently: sort vertices by decreasing finish time.

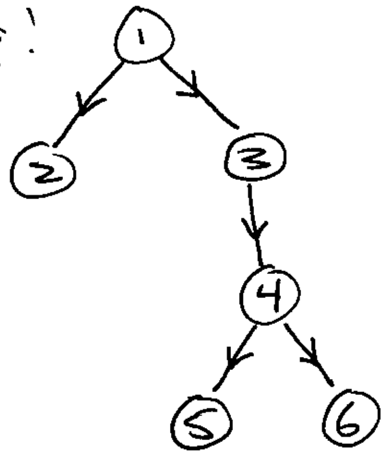
Ex.



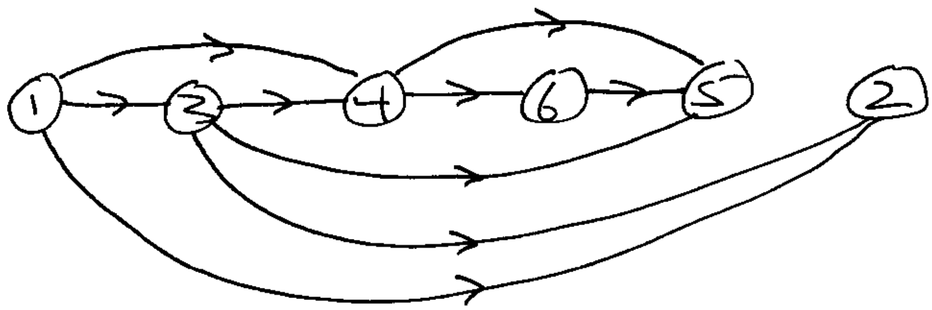
stack

1
3
4
6
5
2

Forest:



Top. sort.



note
are 4
distinct
Top. sort.

(see Thm 22.12 on p. 551 in CLRS)

23.5 Strongly Conn. Components (SCC)

Let G be a digraph. A subset $U \subseteq V(G)$ is called a <u>strongly conn. component</u> of G iff

(1) U is strongly conn. (can get from anywhere to anywhere, and back)

and

(2) U is maximal W.R.T. (1).

<u>Problem</u> find SCCs of G.

(P. 617 in CLRS)

To find SCCs of G:

- Call DFS(G), as vertices finish, push them onto a stack.
- Compute Transpose of G: G^T i.e. reverse all directed edges.
- Call DFS(G^T), in main loop of DFS, process vertices by popping them off the stack, i.e in decreasing finish time (from 1st call) order.

Thm
When this is complete. The trees in the DFS forest (2nd call) span the SCCs of G.

Part 4

$(1, 2, \ldots, n)$

$\downarrow$

$DFS(G, S)$

$\downarrow$

stack: dec. finish time.

$\downarrow$

$DFS(G^T, S)$

$\downarrow$

stack dec. finish time.

can use this to extract tree in DFS forest, & therefore the SCC's.