CSE 101 4-28-20

Depth First Search (DFS)

Vertex attributes:

color[x]: w, g, b

P[x]: Parent (Predecessor)

d[x]: discover time

f[x]: finish time

Two functions

DFS(G)

Visit(x) ← recursive

global variable: time $\in \{0, 1, 2, \ldots, 2n\}$

Predecessor Subgraph

$$G_p = (V_p, E_p)$$

$$V_p = V(G)$$

$$E_p = \{ (\pi[x], x) \mid \pi[x] \neq nil \}$$

ordered pair if digraph
un-ordered pair if graph

Note: Since only white vertices are visited, after which they turn g then b, there can be no cycles.

∴ G_p is acyclic

called: DFS forest.

Note:

DFS has no source, could supply a source s and replace main loop (5-7 of DFS) by single call to visit(s)

could also re-write BFS to not have a source.

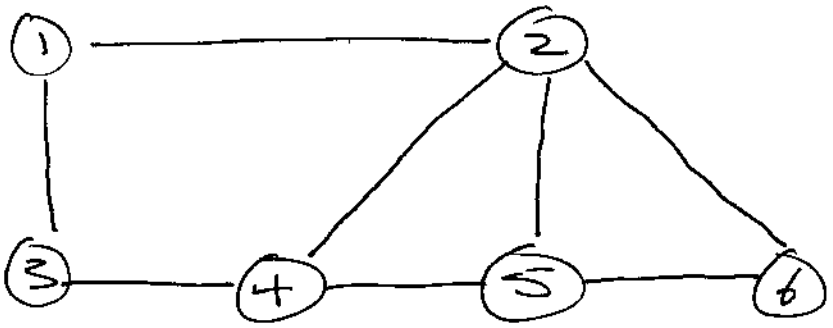
Exercise write the 'other version' of both algorithms.

Note extremes

BFS ——————— DFS

continuum

Ex.

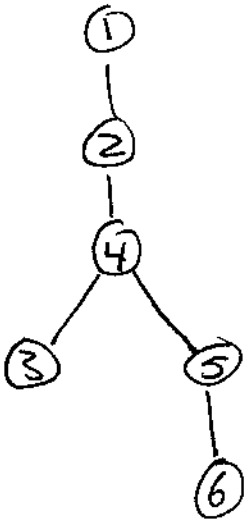


	adj	color	d	f	p
1	2 3	~~w g~~ b	1	12	n
2	1 4 5 6	~~w g~~ b	2	11	~~n~~ 1
3	1 4	~~w g~~ b	4	5	~~n~~ 4
4	2 3 5	~~w g~~ b	3	10	~~n~~ 2
5	2 4 6	~~w g~~ b	6	9	~~n~~ 4
6	2 5	~~w g~~ b	7	8	~~n~~ 5

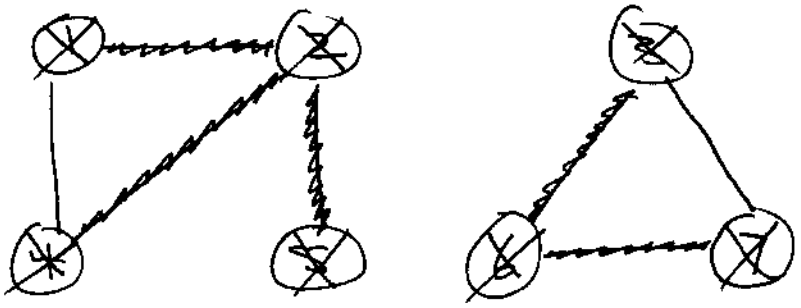
time

~~10~~ 0
~~11~~ 1
12 2
~~3~~
~~4~~
~~5~~
~~6~~
~~7~~
~~8~~
~~9~~

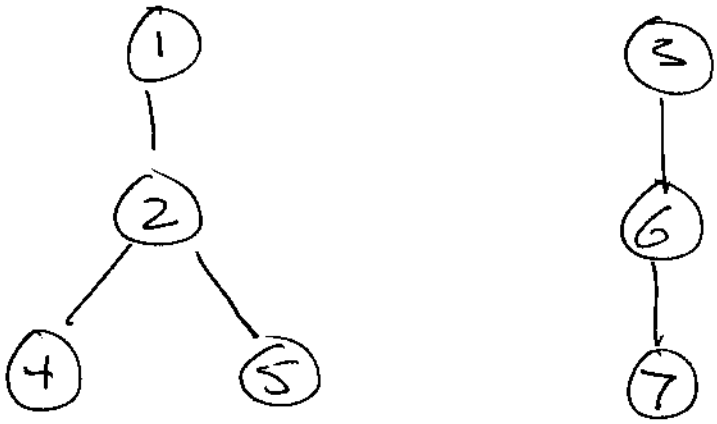
DFS forest



Ex.



DFS Forest



	adj	d	f	p
1	2 4	1	8	n
2	1 4 5	2	7	1
3	6 7	9	14	n
4	1 2	3	4	2
5	2	5	6	2
6	3 7	10	13	3
7	3 6	11	12	6

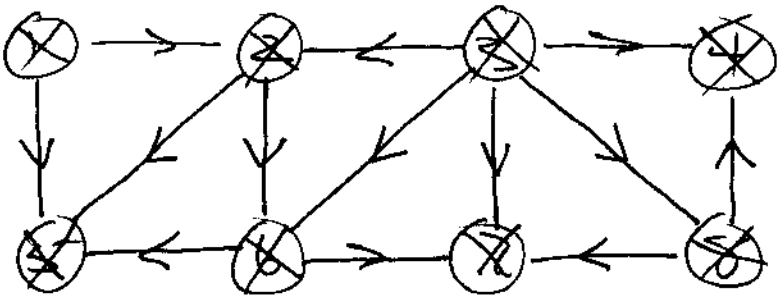
Exercise

re-write DFS so as to determine (1) # of conn. components in an (undirected) graph, and (2) for any $x, y \in V$, determine if they are in same component.

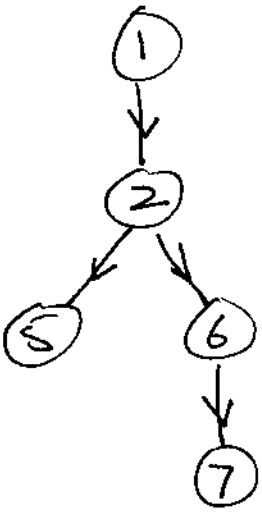
Hint: add another 'global' variable (like time) called count. Increment count from main loop of DFS, each time Visit() is called

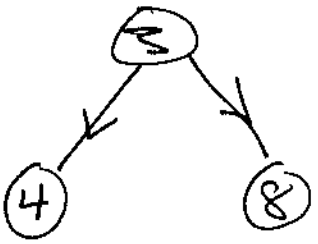
Hint: add a vertex attribute, component[x], stamped on x by Visit(x), value = count.

Ex.



DFS

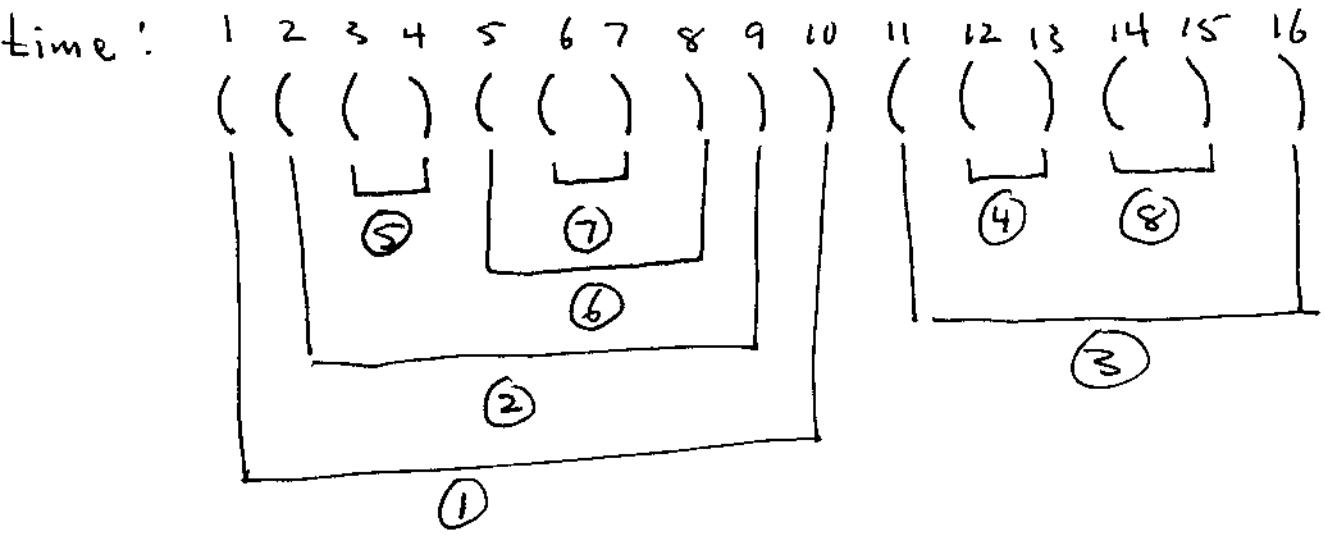




Parenthesis string

"(" discover event

")" finish event



Edge Classification

1. tree edges : belong to G_p
2. back " : join vertex to ancestor
3. forward " : join ancestor to a descendant (other than a child)
4. cross " : all others
   - tree to tree
   - cousin to cousin

Note

classification depends on order in which we execute

- main loop of DFS (5-6
- loop 3-6 in Visit( )

Exercise

Modify DFS so it classifies edges as it goes

Hint: see p. 546-548 & Problem 22.3-4 in CLRS

Sec 23.4 Topological Sort