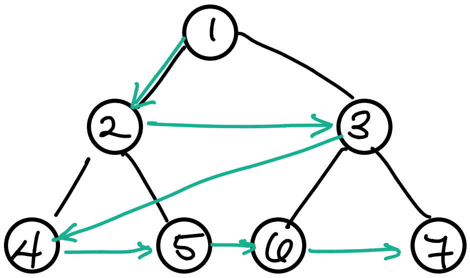
LECTURE 35

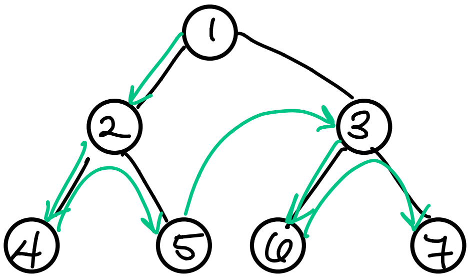
Depth-First Search (DFS)

-BFS explores outward from a source vertex s in all possible directions, adding vertices one layer at a time

DFS differs from BFS in that we sequentially visit verticies until we reach a "dead end" and then backtrack.



BFS



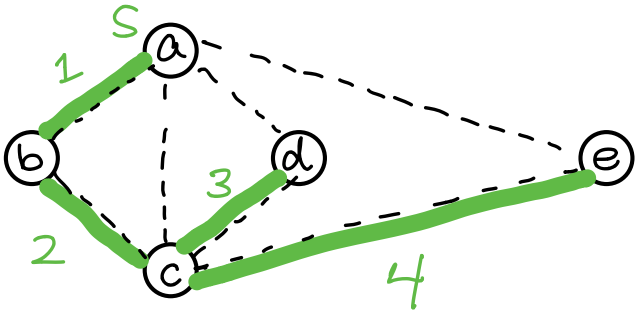
DFS

IDEA: Follow path until you get stuck
-Backtrack until you reach an unexplored neighbor
- Careful not to repeat a vertex.

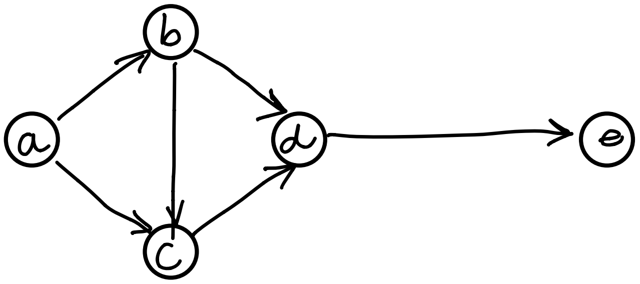
Like BFS, DFS is used for both directed & undirected graphs.

Ex. use DFS to visit all verticies

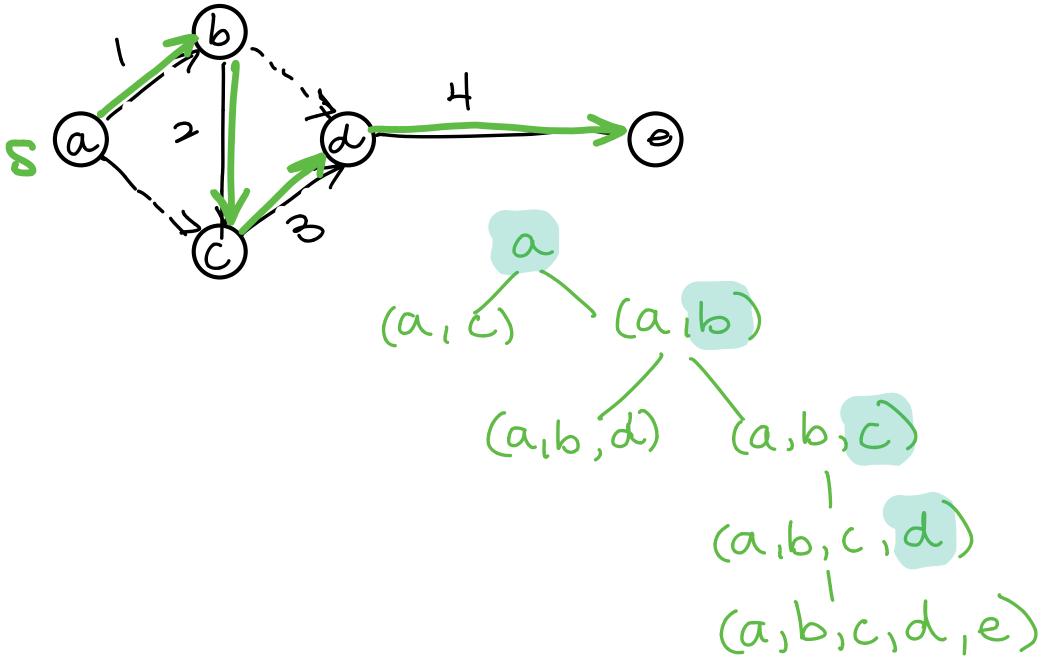
A)

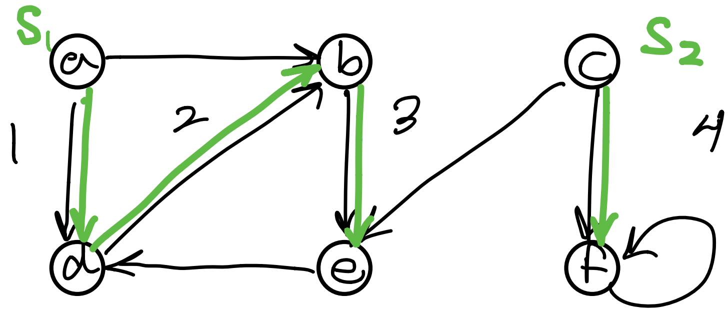


B)



Sol.





Disconnected/ not strongly connected

(No tree, only forest)

Tree edges : (a,d), (d,b), (b,e), (c,f)

See Handout #6 for a version of DFS alg.

Thm. the above implementation runs in $O(V+E)$ time if the graph is given by its adjacency representation

Edge Classification

We can define 4 edge types in terms of the DFS tree (forest) produced by a DFS on a directed graph G

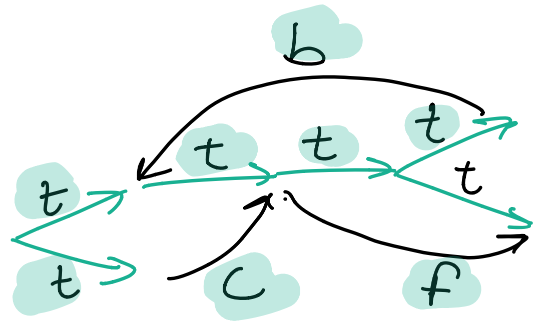
1. Tree edges are edges in the DFS tree (forest)

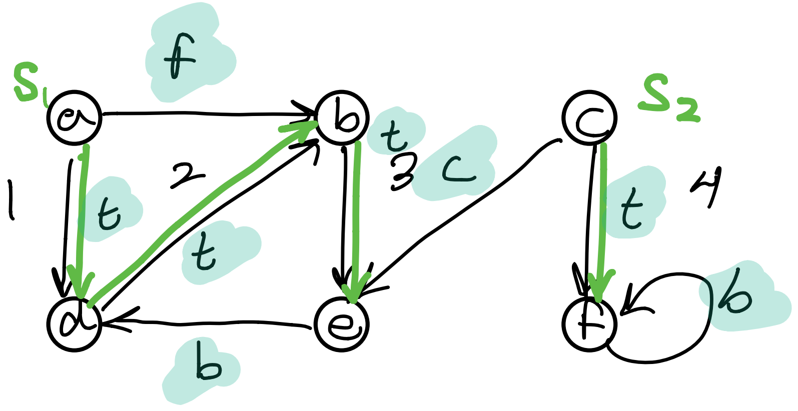
2. Back edges are those nontree edges (u,v) connecting a vertex u to an ancestor v in a DFS tree.

   * we consider self-loops to be back edges.

3. Forward edges - nontree edges (u, v) connecting a vertex u to a descendent v in a DFS tree

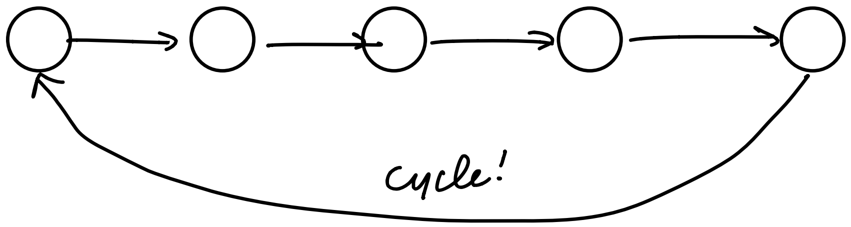
4. Cross edges - all other edges (u and v have no ancestor-descendent relationship in DFS tree)





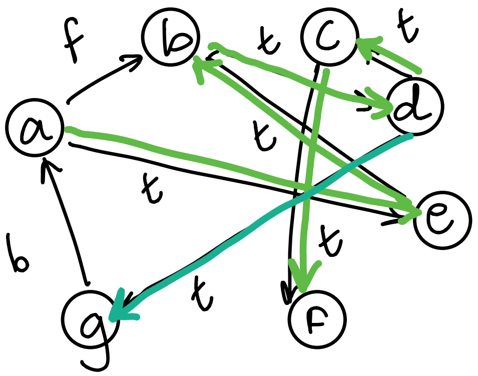
Application : Cycle Detection

FACT : Graph G has a cycle IFF $\iff$ DFS has a back edge.



Back edge to
a tree ancestor

Ex. For the following graph, draw a DFS tree using alphabetical ordering (starting at a), classify the edges as tree, forward, back or cross and determine if the graph has any cycles (w/ justification)



Homework:

https://u.osu.edu/alzalg.1/files/2019/11/hw15.pdf

Handout:

https://u.osu.edu/alzalg.1/files/2019/11/The-depth-first-search-algorithm.pdf