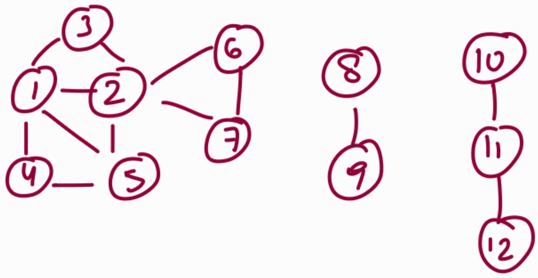
- DFS



$R \leftarrow \{\ \}$

DFS(s):

Discovered [s] $\leftarrow$ True

$R \leftarrow \{s\} \cup R$

For each edge (s, u) incident on s:

if Discovered [u] != True:

DFS(u).

Eg: DFS(1):

$R = \{1\}$

$N(1) = \{2, 3, 4, 5\}$.

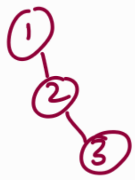
DFS(2):

$R = \{1, 2\}$

$N(2) = \{1, 3, 5, 6, 7\}$.

DFS(3):

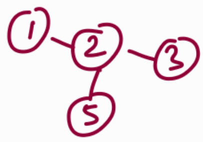
$R = \{1, 2, 3\}$



$N(3) = \{1, 2\}$.

DFS(5):

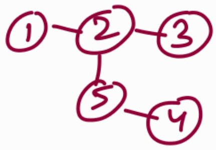
$R = \{1, 2, 3, 5\}$



$N(5) = \{1, 2, 4\}$

DFS(4):

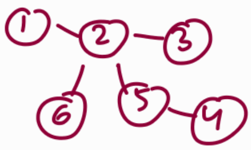
$R = \{1, 2, 3, 5, 4\}$



$N(4) = \{1, 5\}$

DFS(6):

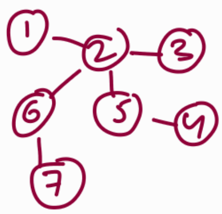
$R = \{1, 2, 3, 5, 4, 6\}$.



$N(6) = \{2, 7\}$

DFS(7):

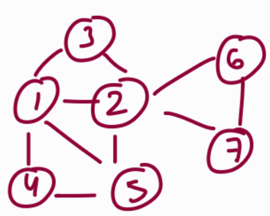
$R = \{1, 2, 3, 5, 4, 6, 7\}$.



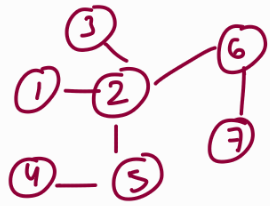
$N(7) = \{2, 6$

$\rightarrow$ Tree edges: Edges $\in$ that appear in DFS tree.

Back edges: Non-Tree edges.



DFS Tree :



→ Only edges to ancestors present. No cross edges.

Useful to detect the presence of cycles in the graph

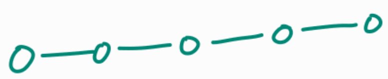
→ There are no cross edges.

Proof: We return only when all the neighbours of the of a given node are discovered. And if it is already discovered, then we don't call DFS on it again. ∴ no cross edge possible.

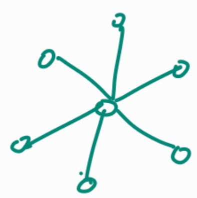
Application: Package mangers in linux sys. have dependency trees

Q: When the BFS & DFS trees of a graph are the same. What can you say about the graph?

1) Path graph.



2) Star graph.



} Undirected, no cycles. So trees.

The above situation arises if & only if the graph is a TREE.

• Modification of DFS to account for the start & end time.

$R \leftarrow \{\}$

DFS(s):

  Start time [s] ← Current time

  Discovered [s] ← True

  $R \leftarrow \{s\} \cup R$

For each edge (s, u) incident on s:

if Discovered $[u]$!= True :

DFS(u).

End time $[s] \leftarrow$ Current time.

- Observation : • If u is a descendant of v in DFS tree then

  start $(v) <$ start $(u) <$ end $(u) <$ end (v)

  • If u and v are unrelated (on diff. branches)

  then (start(u), end(u)) & (start(v), end(v))

  are disjoint.

  • Not possible : start$(u) <$ start$(v) <$ end$(u) <$ end(v)
  for any pair
  of vertices (u, v)

→ Between start and end times of a node u, lie the start & end intervals of all nodes reachable from u without going through parent(u).

- TOPOLOGICAL SORT (DAG)

  • Ordering of vertices :

  Dir^n gives the ordering

  → For any edge in $(u, v) \in E$, there is an ordering $u \leq v$

  → Ordering is transitive. $u \leq v$; $v \leq w \Rightarrow u \leq w$

  ↳ Defining order b/w vertices where there are no edges b/w them.

→ Want the ordering of vertices as per $\leq$.

Topological sort (G):

- Initialise $\text{InDegree}[v]\ \forall v \in V(G)$

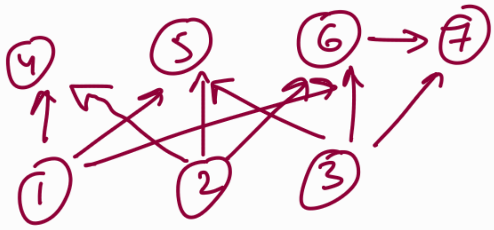
While $\exists$ a vertex that is not pushed into a DS:

  $U \leftarrow$ Set of vertices with indegree 0.

  For all $v \in N_{out}(U)$:

  $$\text{Indegree}(v) = \text{Indegree}(v) - \underbrace{|N_{in}(v) \cap U|}_{\text{Incoming edges to } v \text{ from } U.}$$

  DS.append(U)



$U = \{1, 2, 3\}$.

$N_{out}(U) = \{4, 5, 6, 7\}$.

$\text{Indeg}(4) = 2 - 2$

$\vdots$

$\text{Indeg}(7) = 2 - 1 = 1$

$\Rightarrow \text{Indeg}[4] = 0$, $\text{Indeg}[7] = 1$

$\text{Indeg}[5] = 0$

$\text{Indeg}[6] = 0$

$\{4, 5, 6, 7\} \rightarrow$ Not pushed to DS.

Work it out with examples

$U = \{4, 5, 6\}$.

$$\text{Indeg}(7) = 1 - \underbrace{|\{6\} \cap \{4, 5, 6\}|}_{1} = 0.$$

→ Topological sort in a DFS tree is given by decreasing order of "end times".

DFS on directed tree??

$$
\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \\
4 \arrow[r] & 6 \arrow[ur] &
\end{tikzcd}
$$

First DFS(1), then DFS(4).

Think:

Q. When is a BFS tree unique?
When is a DFS tree unique?