Basic Graph Algorithms - DFS

Init Discovered

$R \leftarrow \{\}$

DFS(s):

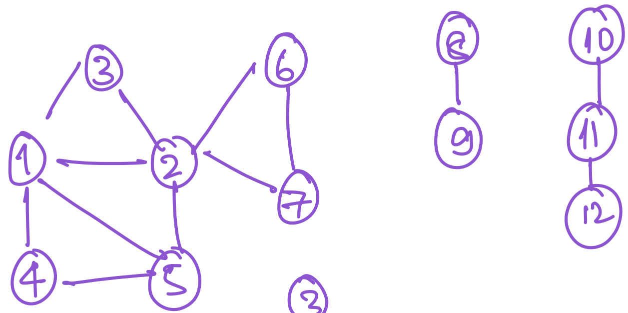
  Discovered[s] $\leftarrow$ True

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

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

    if Discovered[u] == False:

      DFS(u).

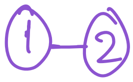


DFS(1)

$R=\{1\}$

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

DFS(2)

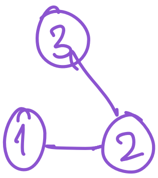


$R=\{1,2\}$

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

DFS(3)

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

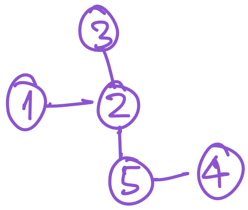


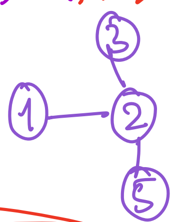
$N(3)=\{1,2\}$

DFS(5)

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

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





DFS(4)

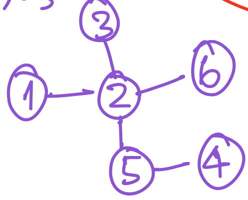
$R=\{1,2,3,5,4\}$

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

DFS(6)

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

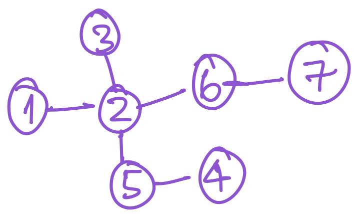
$N(6)=\{\cancel{2},\cancel{7}\}$

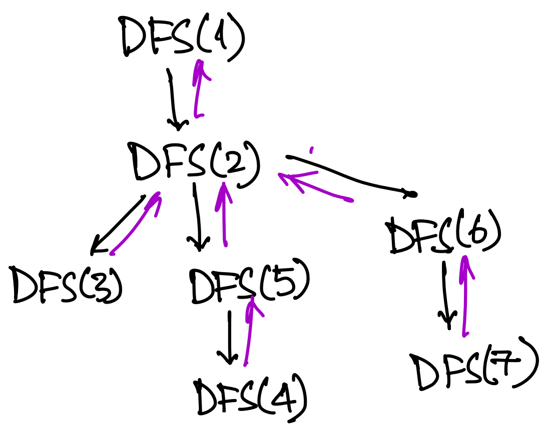


DFS(7)

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

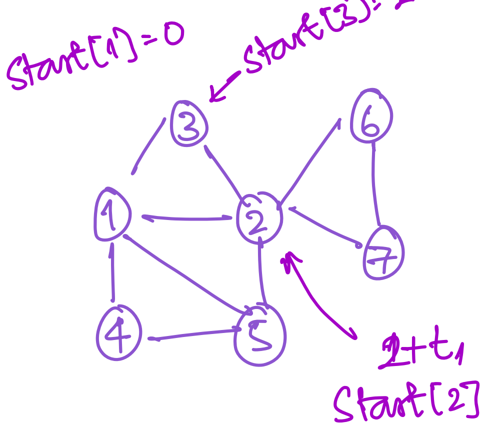
$N(7)=\{\cancel{2},\cancel{6}\}$

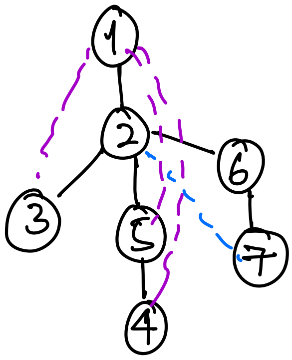




Remark: Let us call the edges from E that appear in DFS tree as tree edges.

$D=t_1+2t_2$



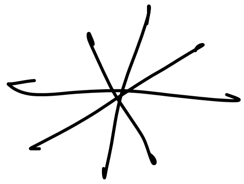


Non-tree edges
↳ Back edges.

<u>Obs:</u> No cross edges are possible.
(undirected)

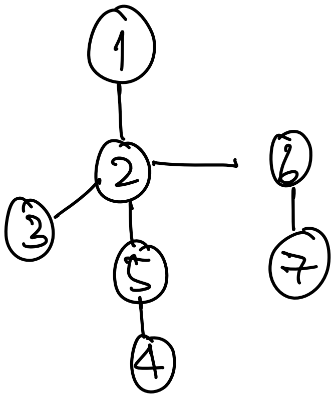
<u>Qn:</u> What if the BFS tree and DFS tree are identical.
What can we say about the graph

This happens if and only if the graph is a tree.



DFS(s):

    Discovered[s] ← True
    R ← {s} ∪ R   start[s] ← current time.   } t
    For each edge (s,u) incident on s:
        if Discovered[u] == False: 1
            DFS(u).
    End[s] ← current time.



<u>Obs:</u>

- If u is a descendent of v in DFS tree,
  $\text{start}(v) < \text{start}(u) < \text{end}(u) < \text{end}(v)$.
- If u and v are unrelated (on diff branches)
  then $(\text{start}(u), \text{end}(u))$ & $(\text{start}(v), \text{end}(v))$ are disjoint.

This cannot happen for any pair of vertices (u,v): $\{ start(u) < start(v) < end(u) < end(v)$

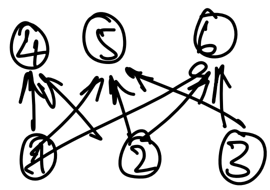
Between start and end times of a node u, lie the start-end intervals of all nodes reachable from u without going through parent(u).

Graph: (Directed) Want "Topological sort". (Acyclic)

Ordering of vertices:

- For any edge in $(u,v) \in E$, there is an ordering $u \leq v$
- Ordering is transitive.

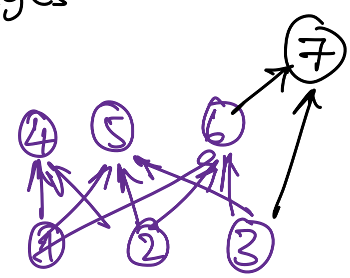
$u \leq v$; $v \leq w \Rightarrow u \leq w$.



Want the sorting of vertices as per $\leq$.

- Start with nodes with no-incoming edges.

Topological sort (G):



- Initialize InDegree[v] $\forall v \in V(G)$.

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

$U \leftarrow$ set of vertices w/ indegree 0.

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

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

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

Indegree(v) = Indegree$(v) - |N_{in}(v) \cap U|$

Indeg(4) $= 2-2=0$

DS.append(U).

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

4, 5, 6, 7
0 0 0 1.

Indeg(7) $= 2-1=1$.

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