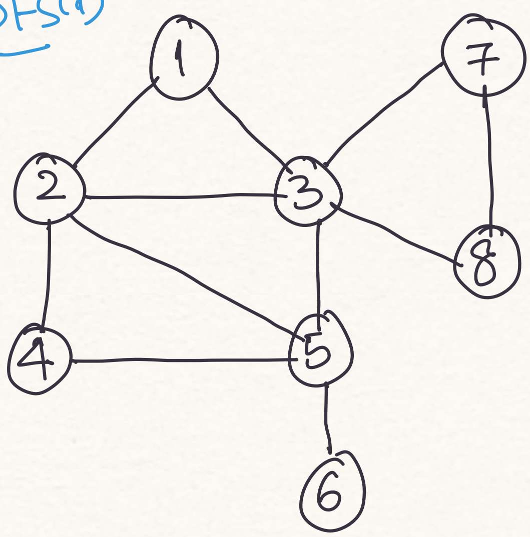
Depth First Search.

```
Stack S ← 1

DFS(s):
    S.push(s)
    While S is not empty:
        u ← S.pop()

        If Discovered[u] == False:
            Discovered[u] ← True
            For each edge (u,v) incident on u:
                S.push(v).
```

DFS(1)



R= {1}

1							

2	3						

R= {1, 3}

2							

2	5	7	8				

R= {1, 3, 8}

2	5	7					

2	5	7	7				

R= {1, 3, 8, 7}

2	5	7					

2	5	7					

2	5						

R= {1, 3, 8, 7}

2	5						

R= {1, 3, 8, 7, 5, 6}

2	4	6					

2	4						

2	4						

R= {1, 3, 8, 7, 5, 6}

2	4						

R= {1, 3, 8, 7, 5, 6, 4}

2							

2							

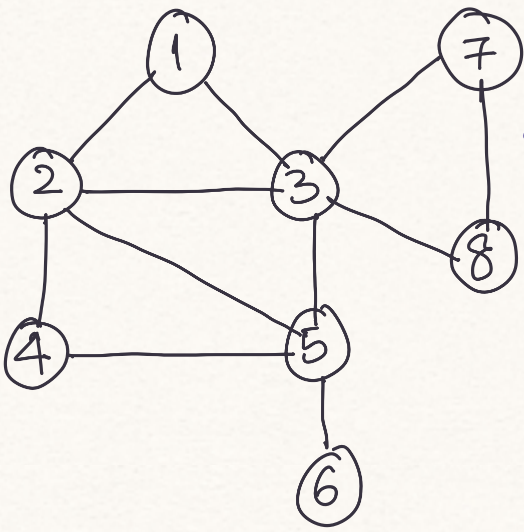
R= {1, 3, 8, 7, 5, 6, 4, 2}

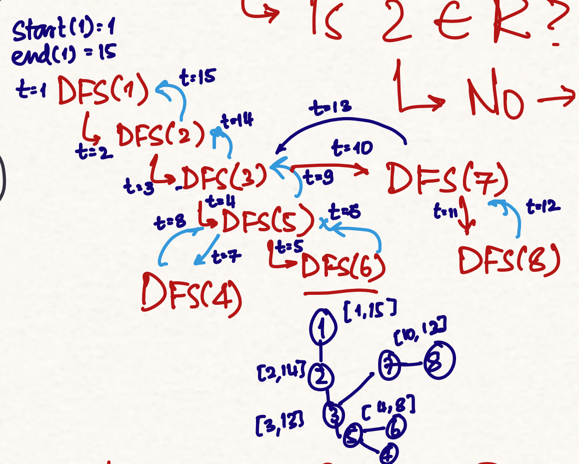
Recursive algorithm:

$T \leftarrow \{\ \}$

For each $u \in V$:
  Discovered[u] = False

DFS(u):
  Discovered[u] = True
  For each edge (u,v) incident on u:
    If Discovered[v] == False:
      $T \leftarrow T \cup \{(u,v)\}$
      DFS(v)





$R = \{1\}$ ①

↳ Is $2 \in R$?

↳ No → DFS(2)

$R = \{1,2\}$

↳ Is $3 \in R$?

No

DFS(3) → $R = \{1,2,3\}$

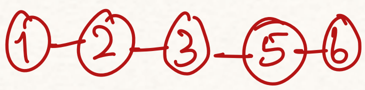
Is $5 \in R$? No

DFS(5) $R = \{1,2,3,5\}$

Is $6 \in R$?

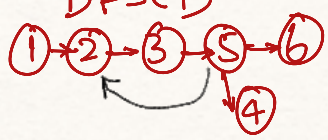
DFS(6)

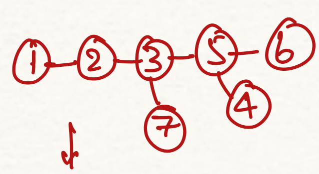
$R = \{1,2,3,5,6\}$



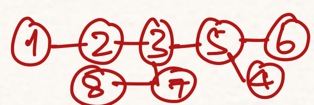
Is $4 \in R$? No → $R = \{1,2,3,4,5,6\}$ → $R = \{1,2,3,4,5,6,7\}$

DFS(4)





$R = \{1,2,3,4,5,6,7,8\}$



Obs: We can associate timestamps start(u) and end(u) for each vertex $u \in V$ through the DFS.

→ If u is a descendant of v in T

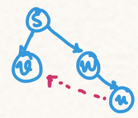
start(v) < start(u) < end(u) < end(v).

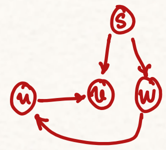
→ If u and v are "unrelated" in T then

[start(u), end(u)] and [start(v), end(v)] are disjoint.

v is looked at from u.

Edges in a DFS tree: $(\overrightarrow{u,v})$





Cross edge example.

- Tree/Forward: start(u) < start(v) < end(v) < end(u)
- Back edge: start(v) < start(u) < end(u) < end(v)

  v is an ancestor and not a parent
- Cross edge: start(v) < end(v) < start(u) < end(u)

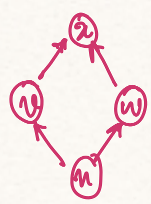
  directed graphs only

Question: Can we find if there are cycles in a given graph?

(u,v) is a back edge that creates a cycle.

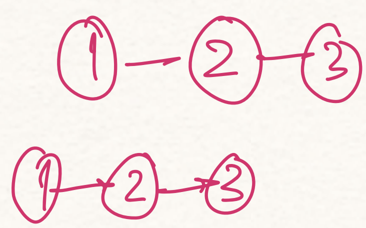
If u is not discovered through v then u gets discovered through some other neighbour.

→ Path $v \rightsquigarrow v' \rightsquigarrow u$ + $u \rightsquigarrow v$ } creates a cycle.



DAG:

Directed Acyclic Graphs.



$u \to v$

(V, E)

$\hookrightarrow (\overrightarrow{u,v}) \Rightarrow u \leq v.$

Sorting of nodes wrt a given "order" of comparision is called

Topological sort: // Assuming the graph is a DAG.

Initialize array InDegree[v] for all $v \in V$.

While there is a vertex that is not pushed into a DS:

    $U \leftarrow$ set of vertices with in-degree 0.

    For all $v \in N(u)$:

        InDegree[v] = InDegree[v] $- |N(v) \cap U|$.

    DS.append(U).