Depth First search (DFS)

G is graph

↳ V : set of vertices (nodes) $= \{v_1 v_2 \dots v_n\}$ $|V| = n$ ← # of nodes

↳ E : set of edges $\subseteq V \times V$

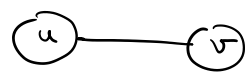
$(u,v) \in E \Rightarrow$ there is an edge from u to v

$|E| = m$ ← # of edges

Undirected

vertex order in (u,v) does not matter

$(u,v) \equiv (v,u)$



u & v are adjacent

Directed

vertex order matters

$(u,v) \Rightarrow$ u is source and v is the target/desination

$u \rightarrow v$

$(v,u) \Rightarrow$ v is source & u is the target

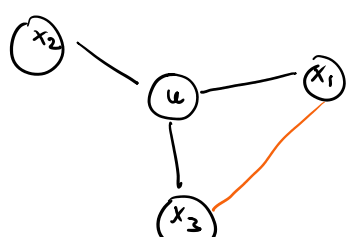
$v \rightarrow u$

Neighbors

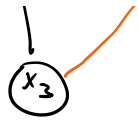
$G = (V, E)$, undirected (Unweighted)

for any $u \in V$

$N(u) = \{ v \mid (u,v) \text{ is an edge in } E \}$

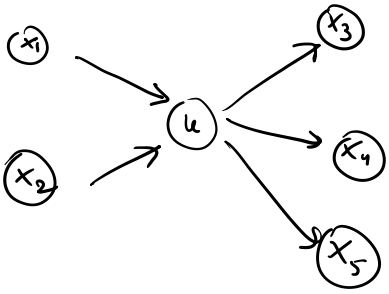


$N(u) = \{x_1, x_2, x_3\}$



Degree $d(u)$: $|N(u)|$ # of neighbors

Directed graphs



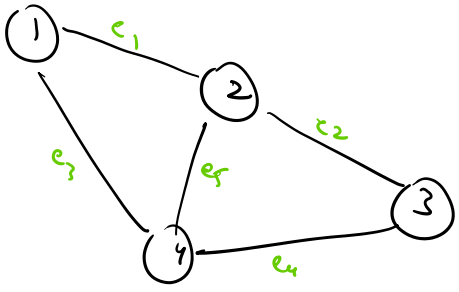
In Neighbors : $N^{in}(u)$

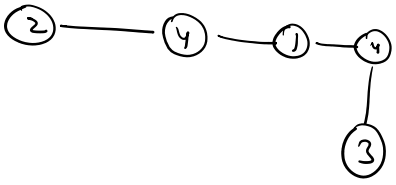
In degree (u) : $\left|\underbrace{\{v \mid (v,u) \in E\}}_{set}\right|$ Cardinality

out Neighbors : $\{v \mid (u,v) \in E\} : N^{out}(u)$

out degree $(u) = |N^{out}(u)|$

Walk: an alternating sequence of vertices & edges





$(2, 4, 1, 4, 3)$

Implicit

$2, e_5, 4, e_3, \ldots$

explicit

Trail: a walk with no repeated edge

$(2, 4, 1, 4, 3)$ is a walk, but not a trail

Path: a walk with no repeated vertex

$(2, 4, 1, 4, 3)$ is also not a path

(2, 4, 1, 4, 3) is also not a path

Graph Traversal

- Depth First search (DFS) ←
- Breadth First search (BFS)

DFS: Connectivity, cycles, ...

Undirected graphs

dfs(G, u):

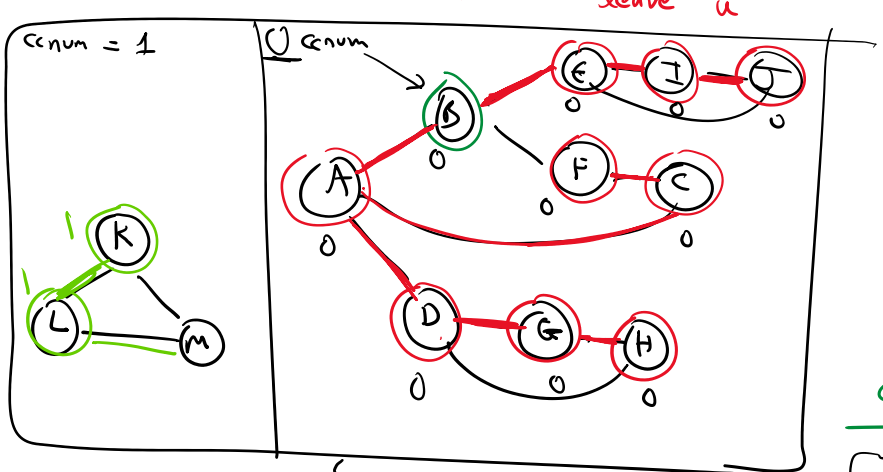
→ visited(u) = True

previsit(u) ← enter u

```
for v ∈ N(u) :
    If not visited(v)
        dfs(G, v)
```

recursive exploration

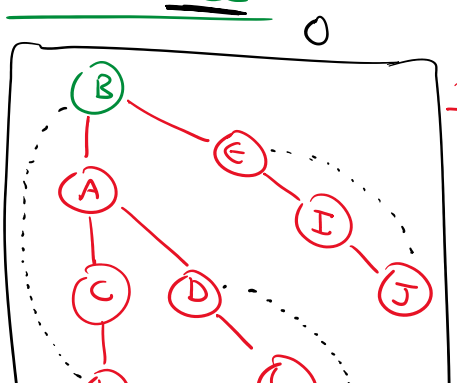
postvisit(u) ← leave u



dfs(B)

N(x): will be looked at in alphabetic order

dfs-tree

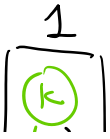


red edge: tree edges (forward)

3 non-tree edges

dfs also give us connected components

Components: largest set of vertice, C, such ...



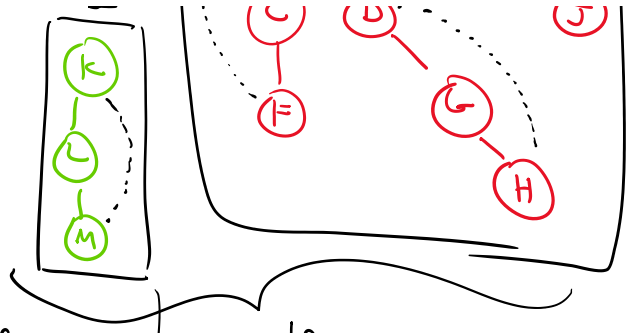
Components

(maximal)

↑

Containment relationship

largest set of vertice, C, such that $\forall\ u, v \in C$ $\exists$ a path between u & v



5 non-tree edges dotted (backward)

dfs forest

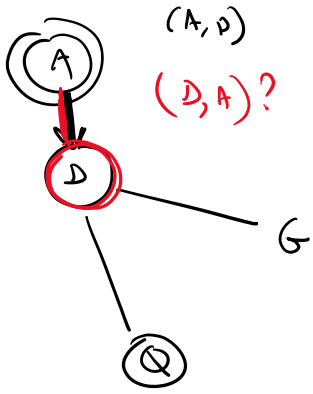
```
cc(G):
    ccnum = 0
    for u ∈ V:
        If not visited(u):
            dfs(G, u, ccnum)
        ccnum = ccnum+1
```

$O(|V|)$

$O(V)$

```
dfs(G, u, ccnum):
    visited(u) = True
    Previsit
    cnum(u) = ccnum
    for v ∈ N(u)
        dfs(G, v, ccnum)
```

$\sum_u N(u)$



(A, D)

(D, A)?

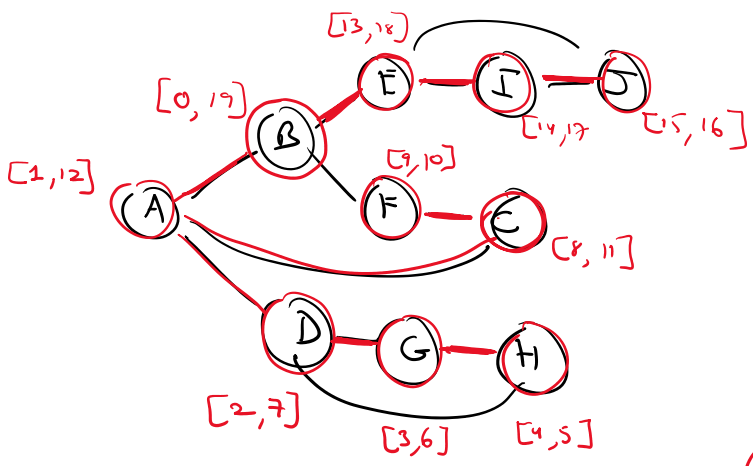
$O(V) + O(E)$

$\boxed{O(V+E)}$

linear

$2E$

Previsit & postvisit numbers



→ previsit(u)

Increment a counter for both events

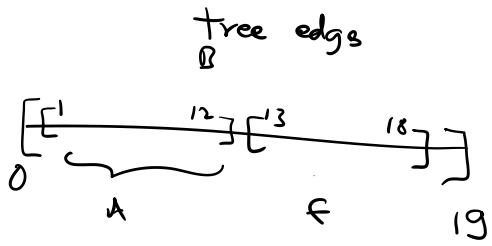
→ postvisit(u)

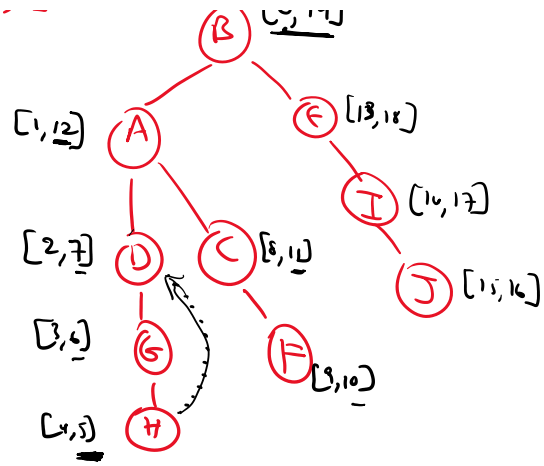
[pre, post]

think of this as an interval

interval

tree edges





dfs order : all tree edges

$\boxed{\text{In the dfs tree if } u \text{ is before } v\text{, then } post(u) > post(v)}$

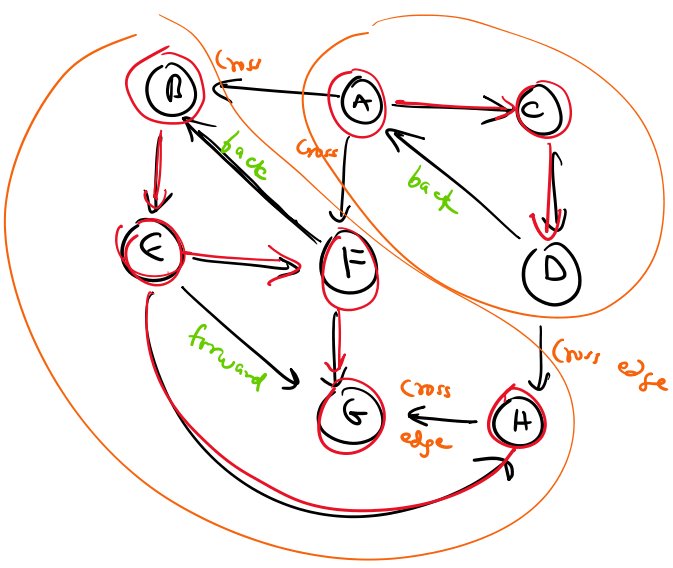
Directed graphs

↳ directed acyclic graphs (DAGs)

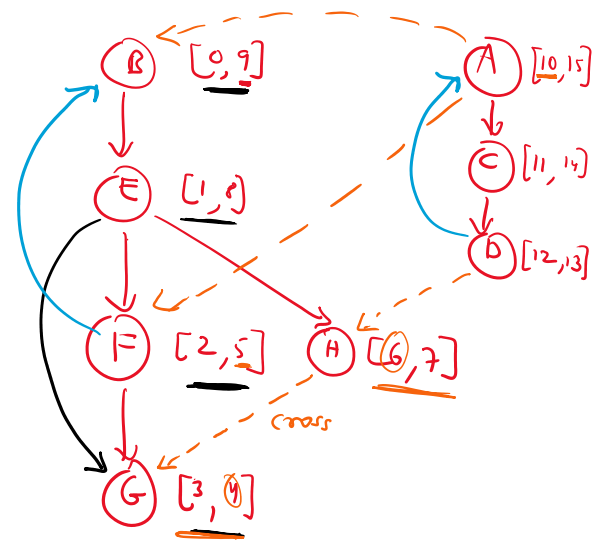
↳ Strongly connected components

① Pre/post numbers

② classify the edge



DFS Forest



red edges: tree edges (dfs walk)

forward edge:
edges from a node to a descendant (E, G)

back edge:
edge from a node to its ancestor (F, B)

Prove:
a directed graph contains a cycle iff ∃ a back edge!

contains a cycle
iff $\exists$ a back edge!

edge from a node to its ancestor (F, B)

cross-edge:

edge from a node u to another node v such that v has already been exited before u is seen

$pre(u) > post(v)$ (H, G)

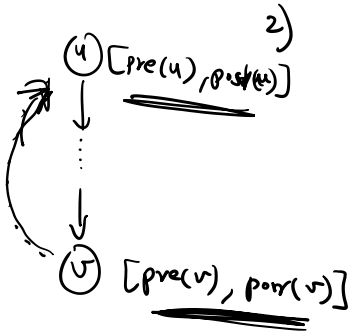
Detecting cycle:

Is acyclic (G): directed graph

1) do pre-post numbering in $\underbrace{O(V+E)}_{\text{linear}}$ time

2) during pre-post numbering, output False if we encounter a back edge

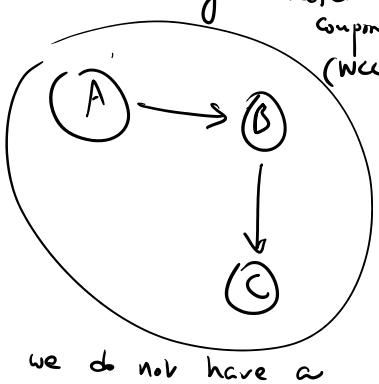
$\exists (v, u) \in E$ // $u \in N(v)$ in dfs



$[pre(v), post(v)] \subseteq [pre(u), post(u)]$

In a directed graph

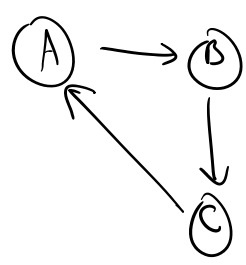
weakly connected component (WCC)



we do not have a path between all pairs of vertices

strongly connected component (SCC)

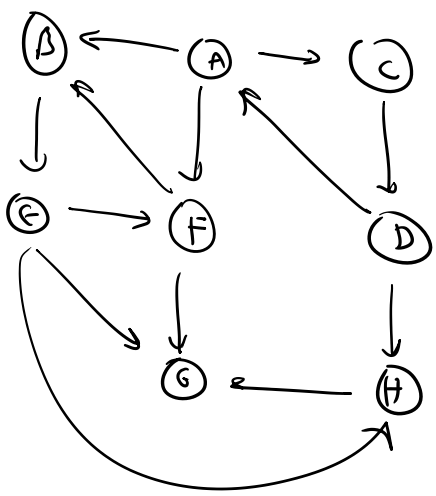
$\exists$ a path between all pairs of vertices in the component

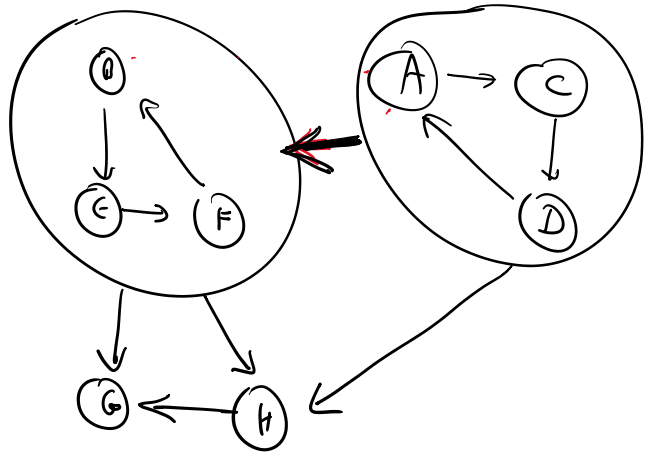


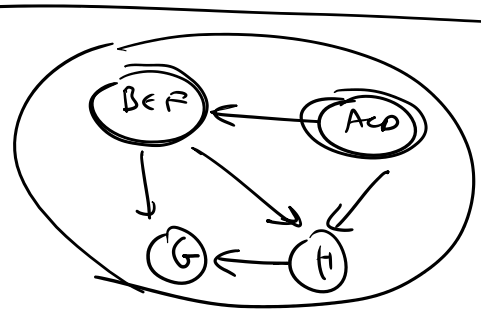
SCC

path between all
pairs of vertices

Any directed graph is a DAG over its SCCs.







DAG