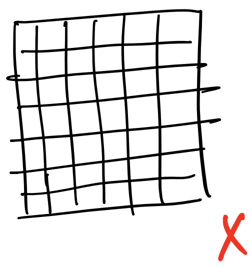
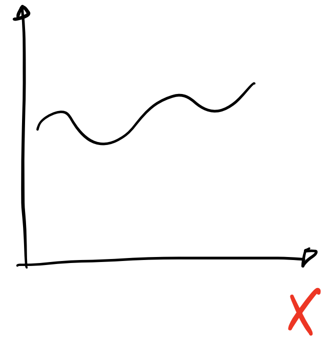
Graphs

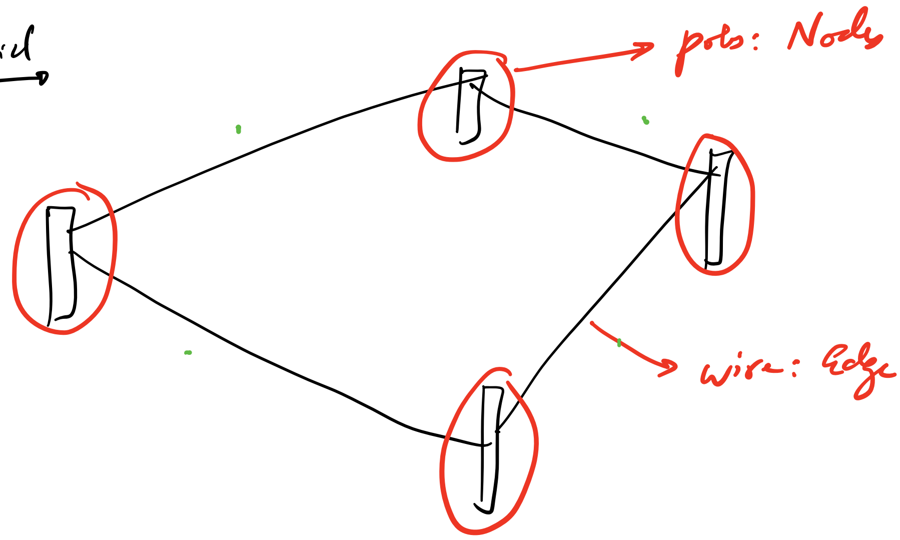




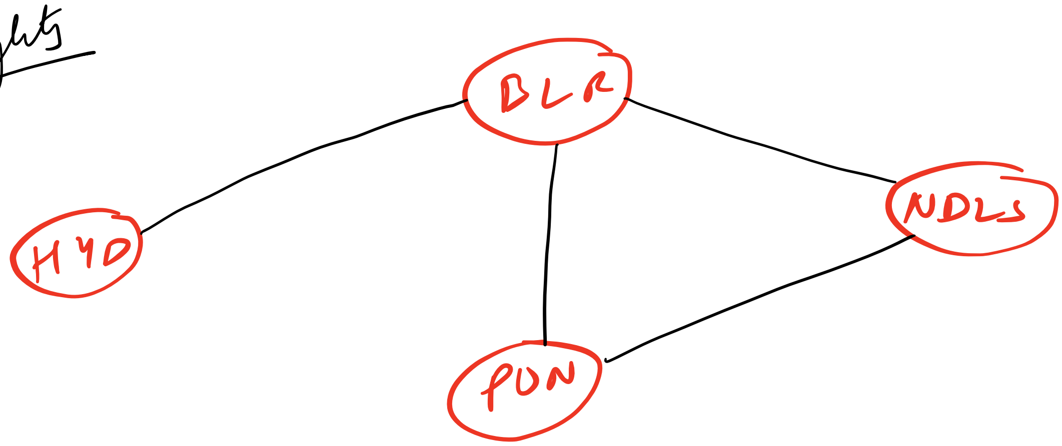
A graph has 2 entities :-

1) Node → Element
2) Edge → Connection b/w Nodes!

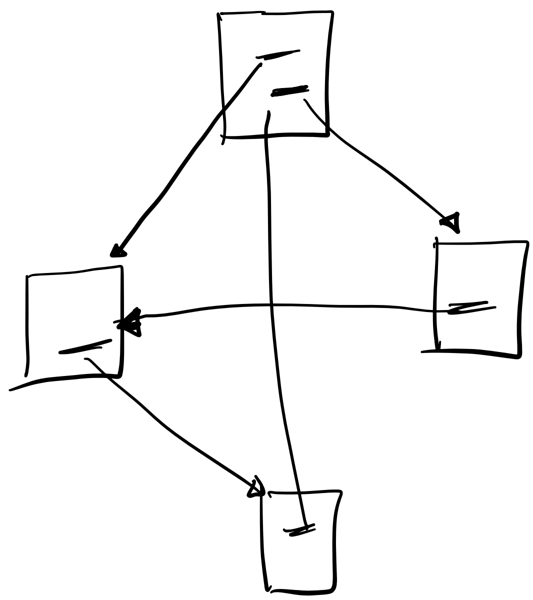
Electrical grid



Flights



Webpages

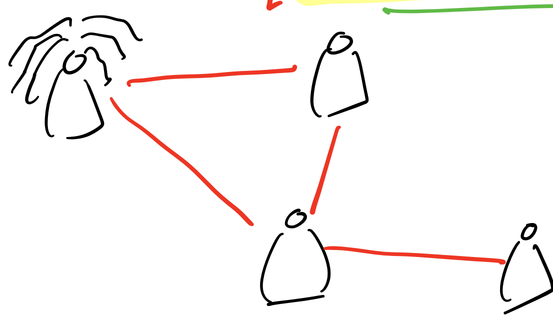


① Social Media

[UNDIRECTED GRAPH]

No direction in Edges

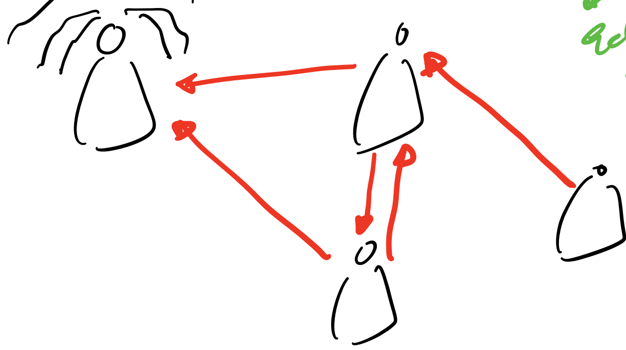
fb



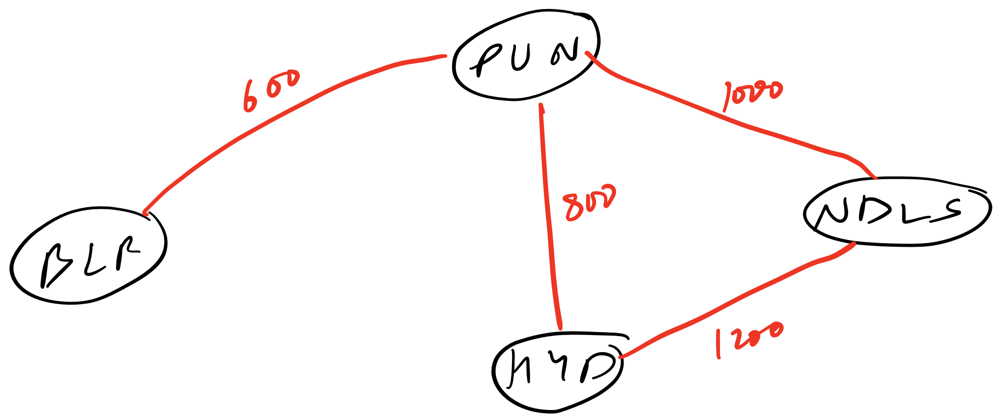
insta

[DIRECTED GRAPH]

edges have direction.

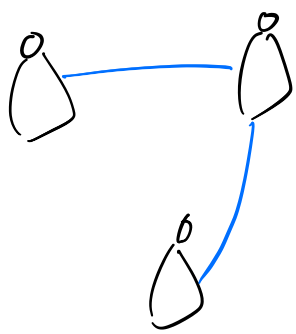


1. Weighted graphs → Weights are associated with edges!

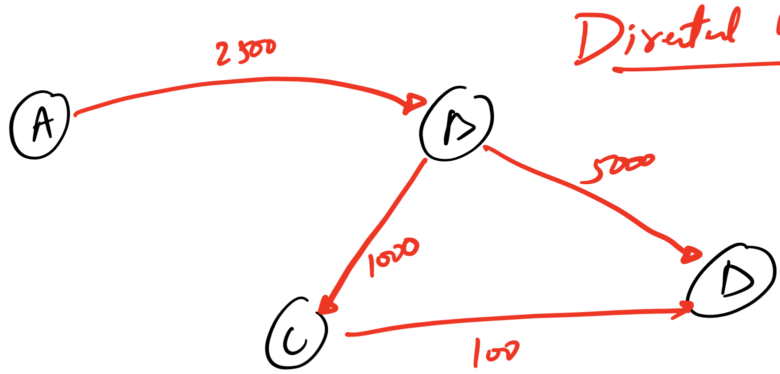


1. Unweighted graph

fb



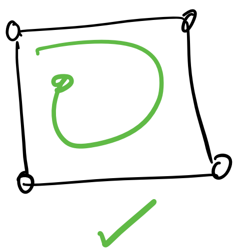
Directed Weighted graph

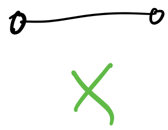


Cyclic graph

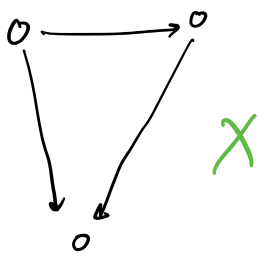
→ If we have a simple path from a node to itself.

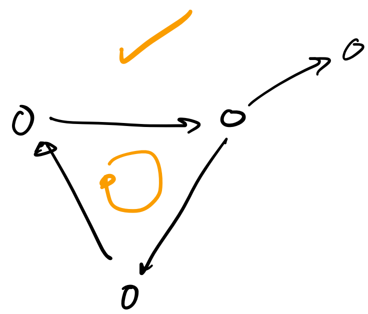
DISTINCT EDGES

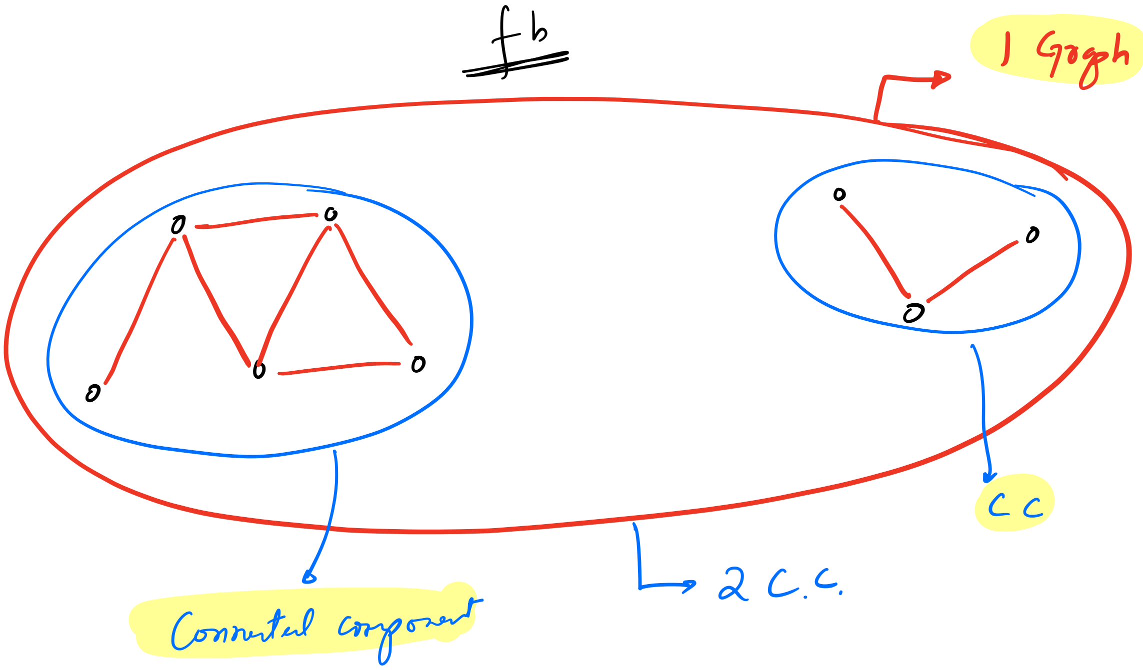




DAG → Directed Acyclic Graph!

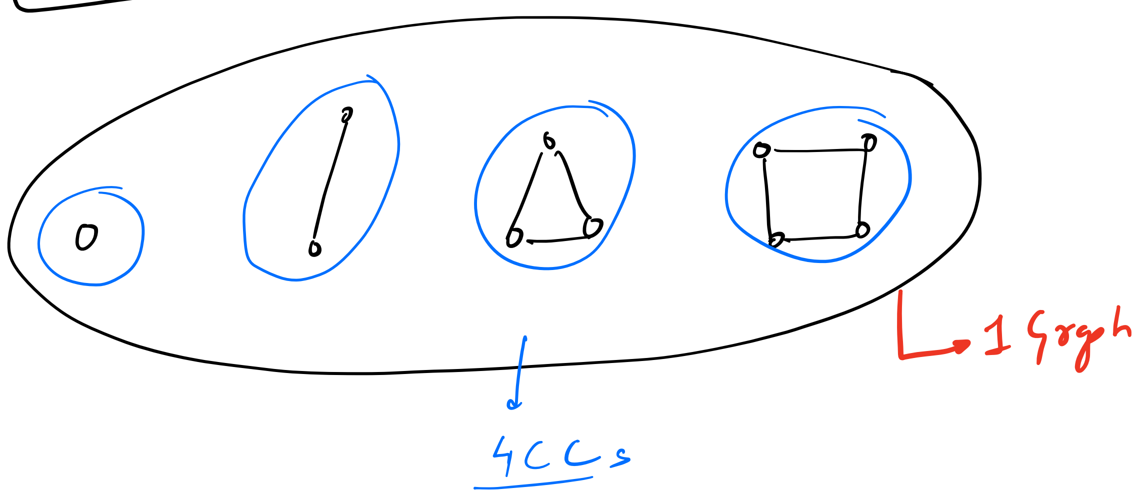






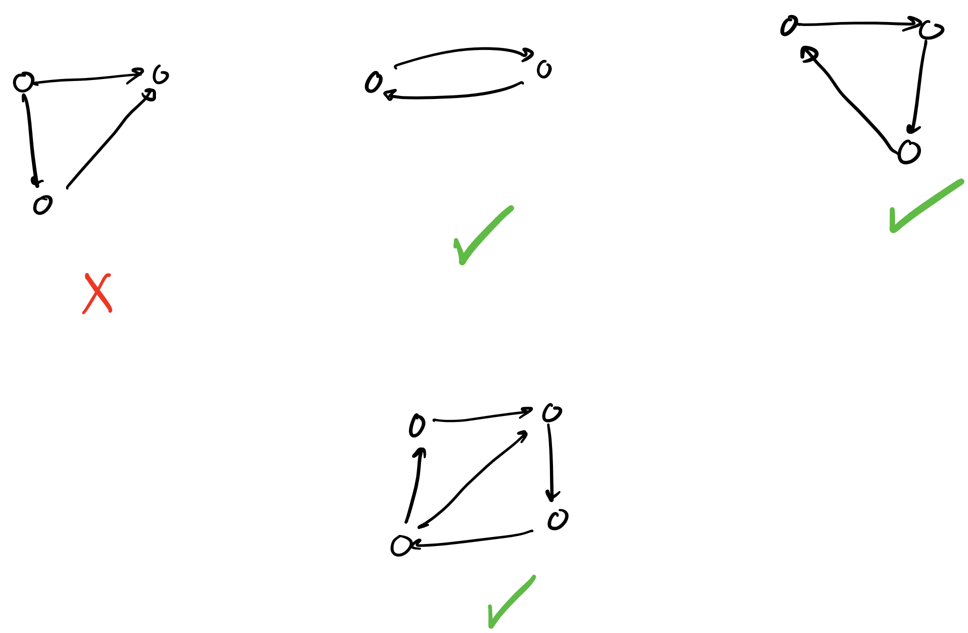
CC : [Undirected graph]

→ You have a path b/w any 2 nodes in that CC



Strongly Connected Component [Directed G]

→ Def is same!



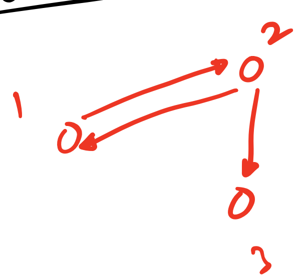
⊙ How to rep graph in CODE ?

V: Vertex : Nodes

G → { V, E }

1) Adjacency Matrix

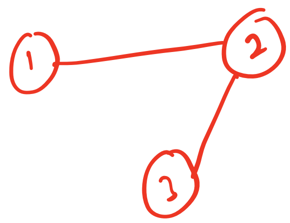
A) DIRECTED



$$
\begin{array}{c|ccc}
 & 1 & 2 & 3 \\
\hline
1 & 0 & 1 & 0 \\
2 & 1 & 0 & 1 \\
3 & 0 & 0 & 0
\end{array}
$$

if (A[i] [j] == 1) ⇒ $i \to j$

3) UNDIRECTED



$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array}$$

$A_{ij} == A_{ji}$

$\downarrow$

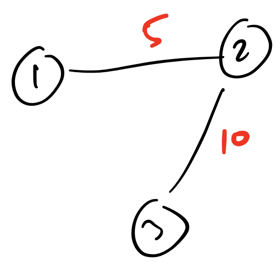
Symmetric

if (A[i][j] == 1)

$\Rightarrow$ A[j][i] = 1

$\Rightarrow$ (i) — (j)

4) WEIGHTED

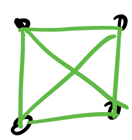


$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 5 & 0 \\ 2 & 5 & 0 & 10 \\ 3 & 0 & 10 & 0 \end{array}$$

$$\boxed{SC = O(V^2)}$$

MAX # edges

Undirected Graph (V)



$^{V}C_2$

Directed graph (V)

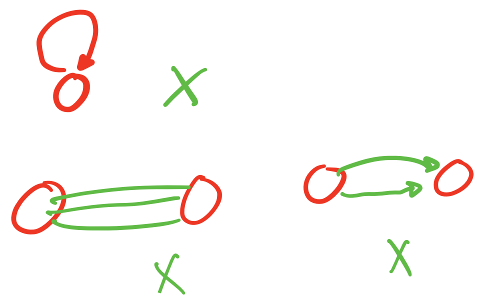
$2 \cdot {}^{V}C_2$

$\sim V^2$

Simple graph

→ No self loops

→ No multiple edges



Pros of Adj. Matrix

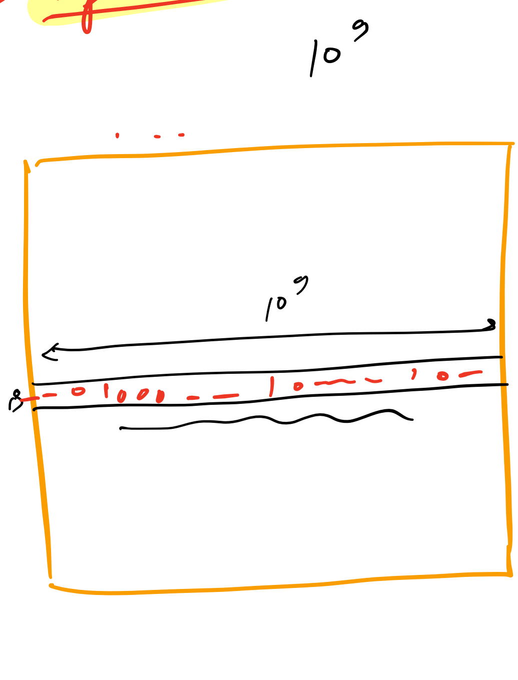
1) Check if edge b/w i & j → $O(1)$
2) add → $O(1)$
3) remove → $O(1)$

Cons

X Space usage is high

↳ Sparse Matrix

fb ~ 10^9

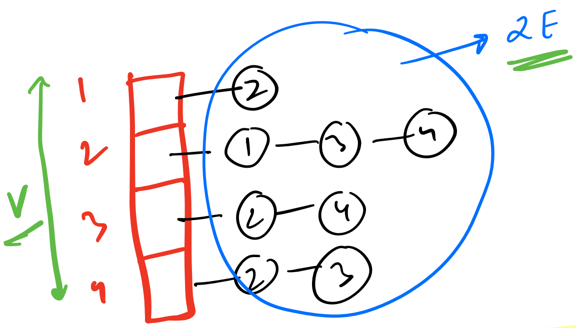


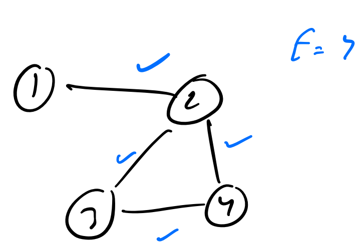
10^9

$(10^9 \times 10^9)$ →

2) Adjacency List

$G \to \{V, E\}$



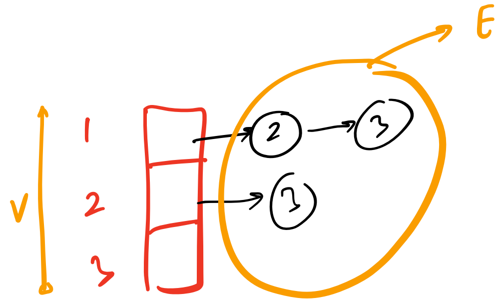


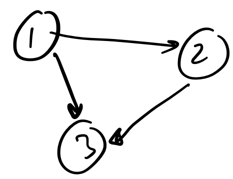
x — y

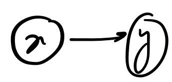
adj[x].add(y)
adj[y].add(x) × 2

$$\boxed{SC = O(V + 2E)}$$

Array < ArrayList > adj

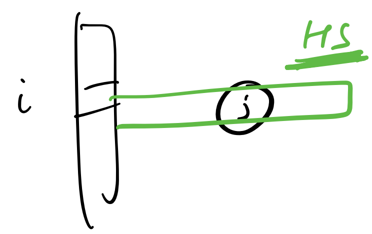






adj[x].add(y) ✓

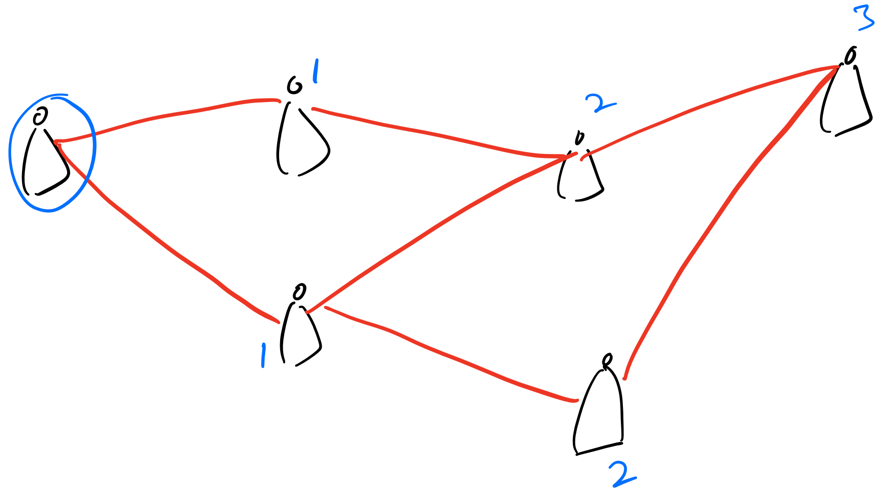
$$\boxed{SC = O(V + E)}$$



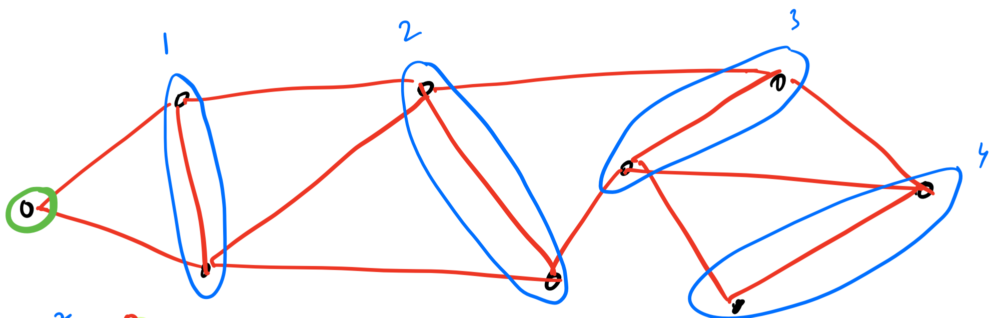
pros
→ Space efficient ✓

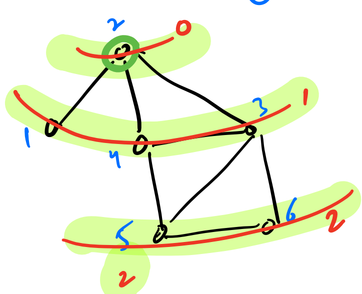
Cons
→ check if edge is present b/w i & j quickly
i & j — ?
→ add — ?
→ rem

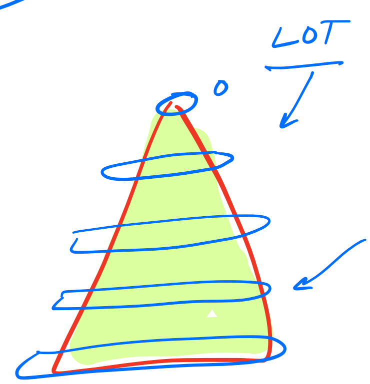
(8) LinkedIn

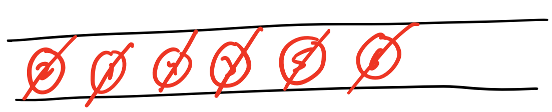


Q Given an UNDIRECTED graph.
find the shortest distance to all nodes from
a given node!









vis [] or vis HS

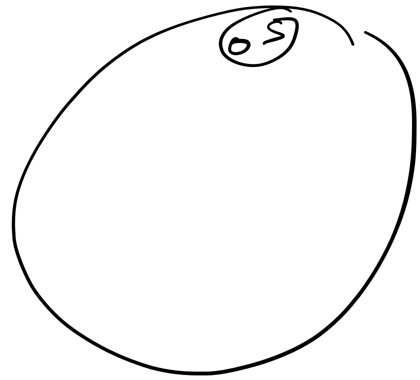
BFS [Breadth First Search]

vis[], dis[]

// Adj list[]

```
void bfs ( int source ) {
    queue<int> q;
    q.enqueue ( source );
    vis[source] = true;
    dis[source] = 0;

    while( ! q.isEmpty() ){
        int p = q.front();
        q.dequeue();
        f( u : adj[p] ){
            if( vis[u] == false ){
                q.enqueue( u );
                vis[u] = true;
                dis[u] = dis[p] + 1;
            }
        }
    }
}
```



$G = \{V, E\}$

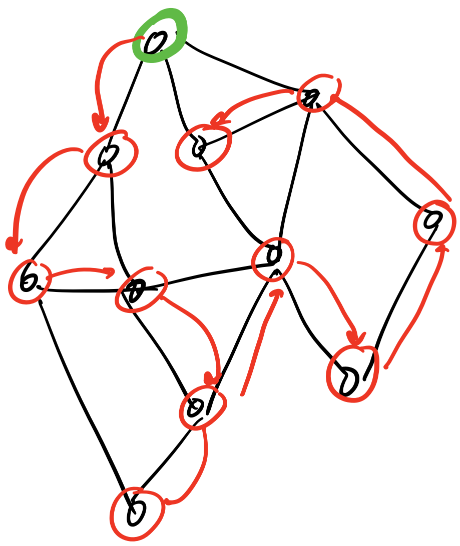
$TC = O(V + 2E)$

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

$\boxed{SC = O(V)}$

Adj list is NOT to be counted.

DFS [Depth First Search]



Use

don't care about the order of traversal

→ Easy to code!

// vis[]

```
void dfs( int v) {
    vis[v] = true;

    f( u : adj[v]) {
        if ( vis[u] == false) {
            dfs( u);
        }
    }
}
```

$TC = V + 2E$

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

vis[] Rec

$SC = V + V$

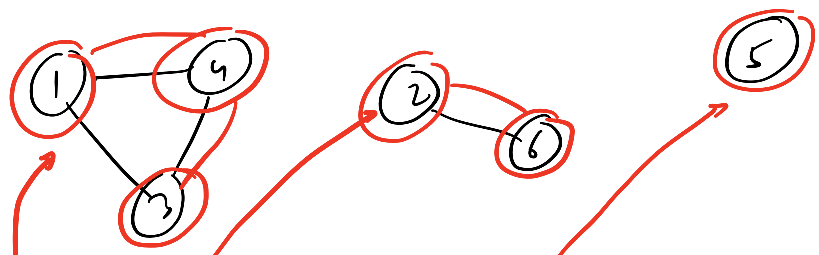
$\boxed{SC = O(V)}$

Q Given a undirected graph with nodes numbered from 1 — V

Find the # of CCs!

u — v → Adj list → 3 CC's

V = 6



dfs() (1) (2) 3 4 (5) 6

CC ✓ ✓ ✗ ✗ ✓ ✗ → 3 CCs

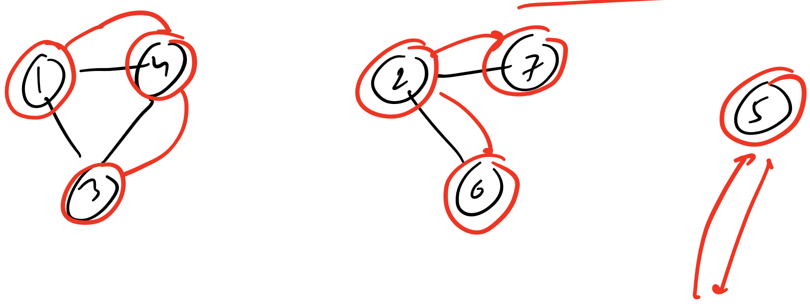
```
cc = 0;
f(i = 1 → V) {
    if(!vis[i]) {
        dfs(i);
        cc++;
    }
}
ret cc;
```

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

$\boxed{SC = O(V)}$

Q Return a list of no. of nodes in each CC!

ANS: < 3, 3, 1 > → ANY permutation!



dfs	(1)	(2)	3	4	5	6	7
	✓	✓	✗	✗	✓	✗	✗
	3	3			1		

```
f(i=1 → v){
    if(vis[i]==false){
        cnt=0;
        dfs(i);
        print(cnt);
    }
}
```

```
void dfs(int v){
    vis[v]=true;
    cnt++;
    f(u : adj[v]){
        if(vis[u]==false){
            dfs(u);
        }
    }
}
```

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

$\boxed{SC = O(V)}$