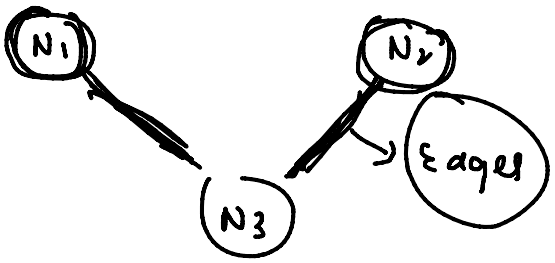
Class - 92

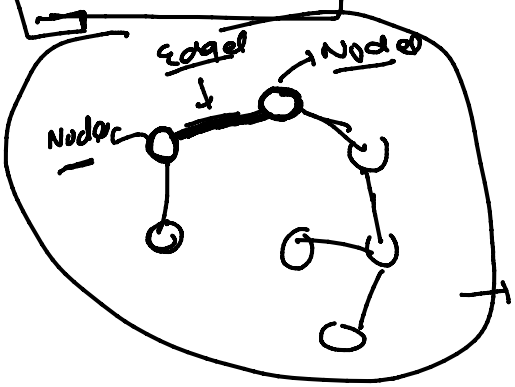
Graphs

Tree → Heirachical Data structure



Edges → Connection / relationship b/w Nodes

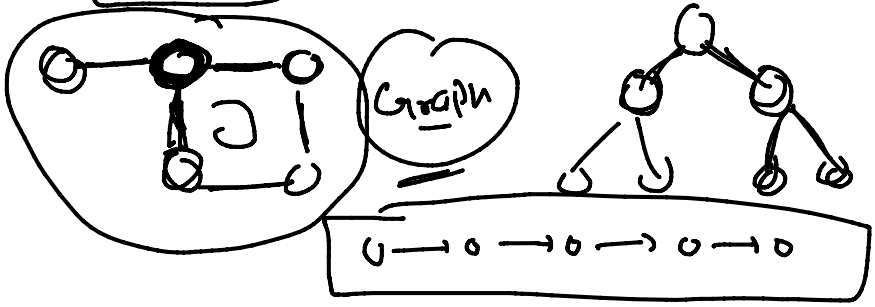
Multiple Nodes → connected through edge



Edges → Relationship present b/w Nodes

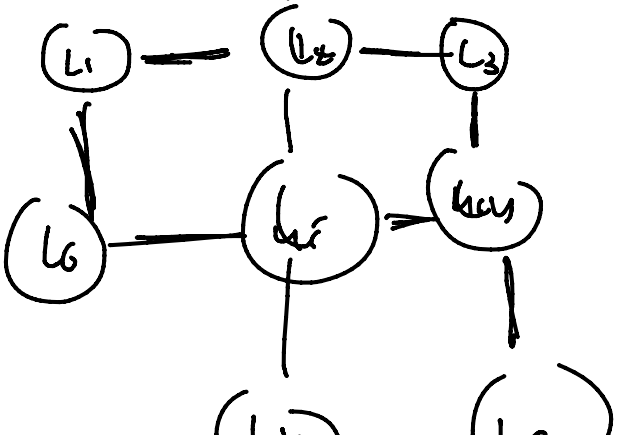
Trees → subset of Graph

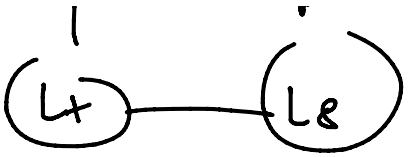
Graphs may contain cycles



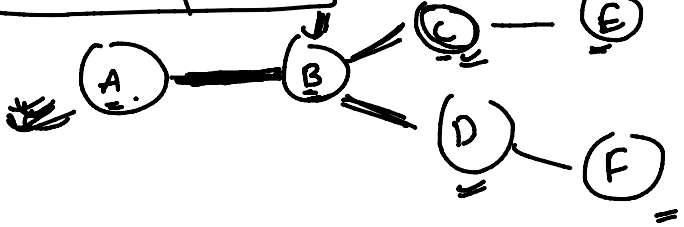
Graph Data Structures Real life

① Google maps :-





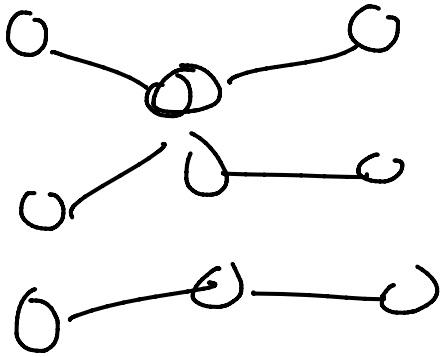
(2) Social Networking site :- Mutual friend :



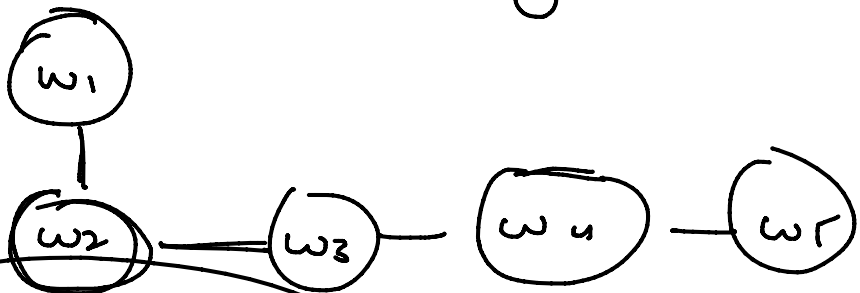
Mutual friend

(3) Recommendation engine

Graph



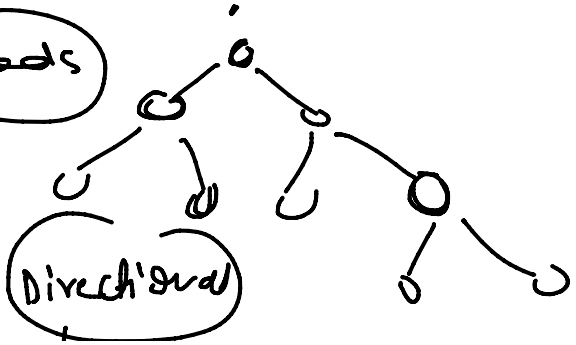
(4) web search



wikipedia

Graph

Bit torrent -> seeds



Directional

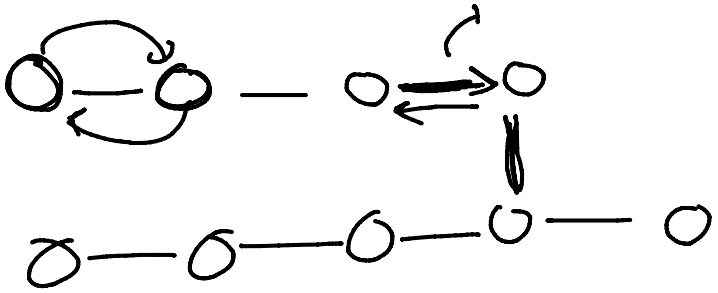
L1 — L2

Types of Graphs

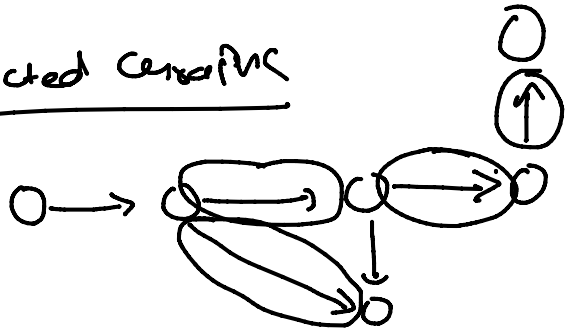
Bidirectional Graph

Undirected Graph

edges



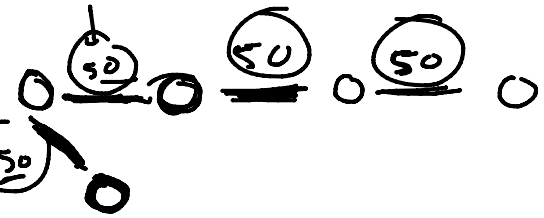
Directed Graphs



Weighted Graph

weight

friendship score



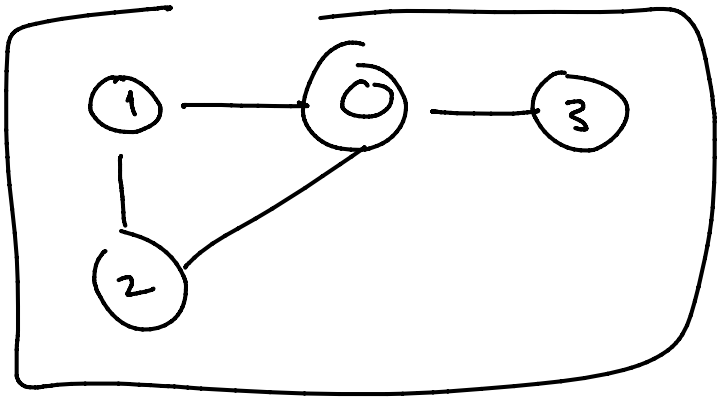
edges → Directional
→ weighted

Unweighted Graph

Directional → weighted
→ unweighted

Undirected → weighted
→ unweighted

⇒ Cyclic graph

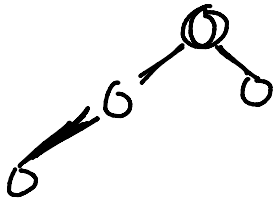


Graph vs Tree

Tree ⇓ root Node

(n) Nodes

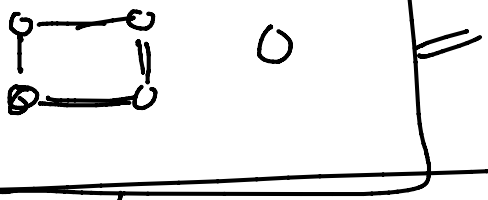
Graph with (N) nodes and (n-1) edges

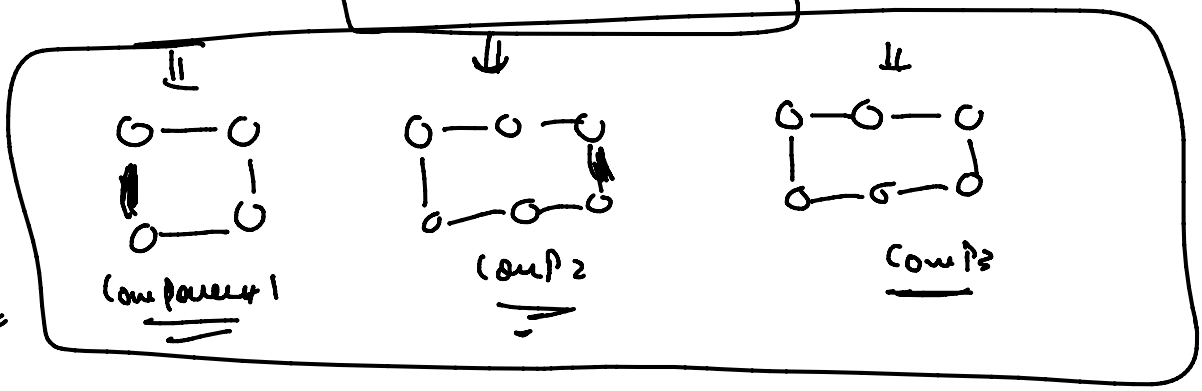


(n-1)

Single connected

Components of graph





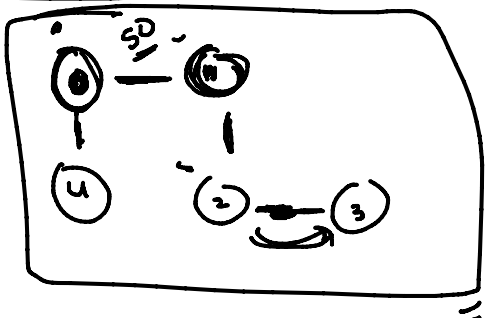
Representation of Graphs

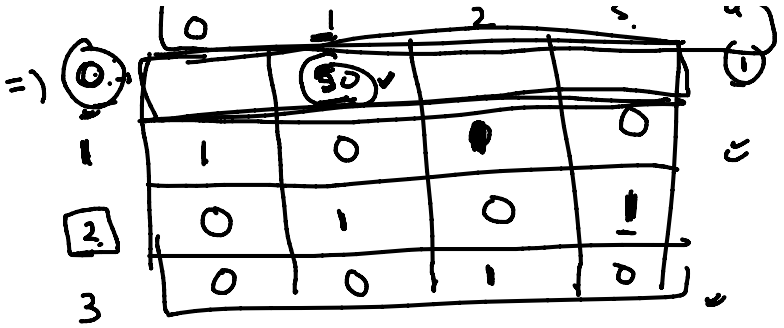
store, access

0 — 1

→ Adjacency matrix :-

Adjacency Matrix :-





Check (u) — (v) O(1)

(u) (v)

(2) (3)

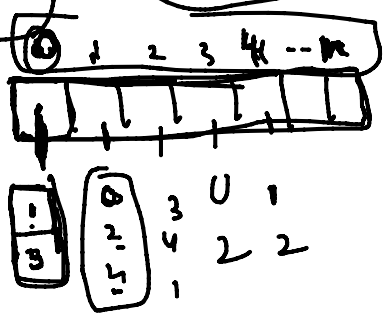
Memory → $O(n^2)$

Every Node →

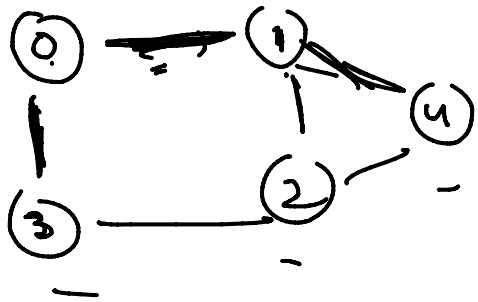
Graph travel → T.C ⇒ $O(n^2)$

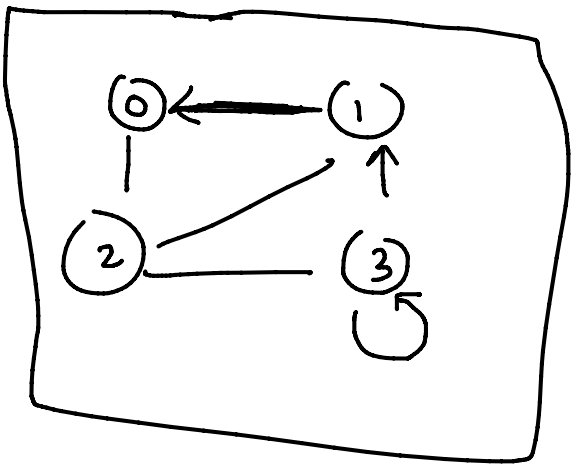
(2) Adjacency list

N Nodes

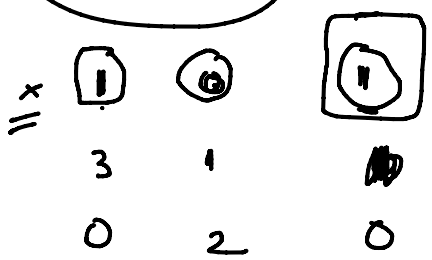


Vector vector ki list



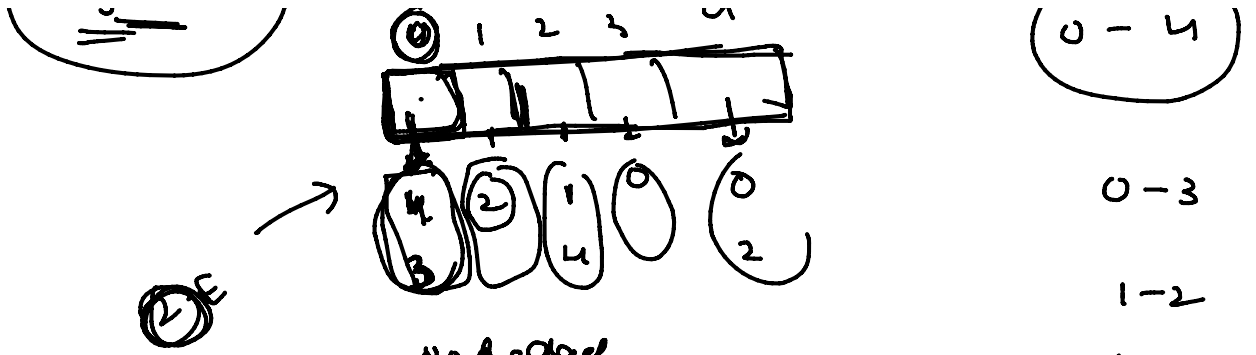


→ adj Matrix



Adj list

0 - 4

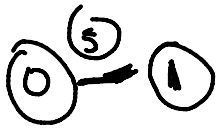


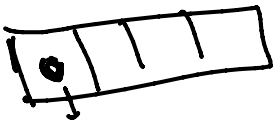
No of edges

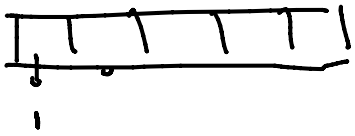
S.C $\Rightarrow$ O(V+E)

T.C = O(V+E)

No of Nodes







Graph Traversal → visit every Node and edge of graph exactly once in a well defined order

Backtracking

0 1 2 3

3 2 1 0

0 2 3 1

Tree

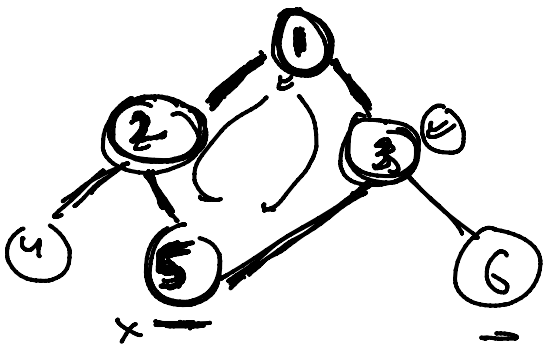
DFS, BFS

Graph

DFS BFS

BFS

BFS



1 2 3 4 5 6

1 2 3 4 5 6

queue → 4 5 6

+ visited array

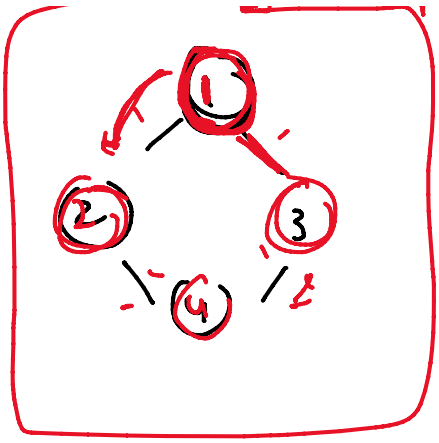
1 2 3 4 5 6

```
vector<int>vis(n,0);

queue<int> q;

q.push(0);

vis[0]=1;

while(!q.empty())

{ auto tp= q.front(); cout<<tp<<" ";

   q.pop();

   for( auto j : adj[tp])

   { if(!vis[j])

      { q.push(j);

        vis[j]=1;
      }
   }
```

1 2 4 6 5 3

vis[j] = 1;

DFS



1	2	3	4
1	1	1	1

1 2 4 3

T.C = O(V+E) = O(V+E)

```
void dfs(int node)
{
    vis[node] = 1;
    for(auto j : adj[node])
    {
        if(!vis[j])
        {
            dfs(j);
        }
    }
}
```

adj list

```
cout << node << " ";
for(i=0; i<v; i++)
{
    if(adjMatrix[node][i])
    {
        if(!vis[i])
        {
        }
    }
}
```

V * V = V^2