Today's agenda

↳ Intro

↳ Types of graph

↳ Storage

↳ BFS (level order) +1

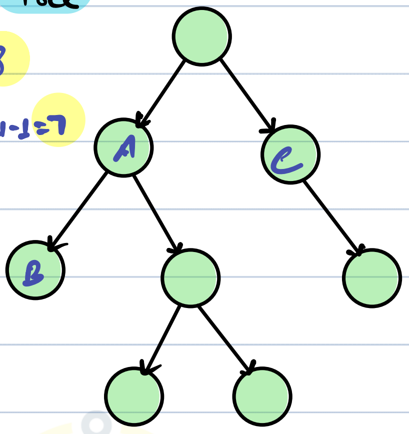
Intro

↳ Graph: Connection of nodes & edges

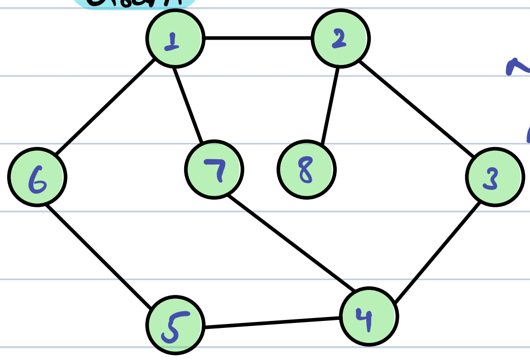
1. Tree

N: 8

E: N-1=7



Graph



N: 8

E: 9

2, 6 & 7 are nbrs of 1.

→ main diff betⁿ trees & graphs

1. Nodes in graph can have more than 1 Parent/nbrs.

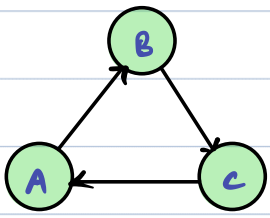
2. Graph has no hierarchy or root node.

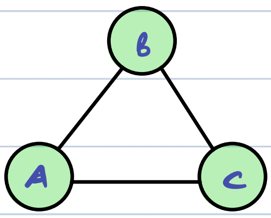
3. Graph can have cycles.

4. Any directional movement is allowed in graph.

* Classification of graphs

Case I: Based on types of edges.

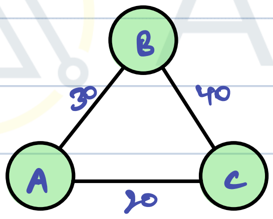


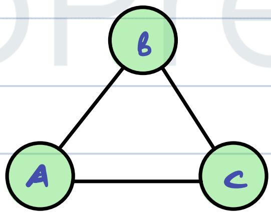


↳ directed graph
↳ insta follower

↳ undirected graph / Bidirectional
↳ facebook friend

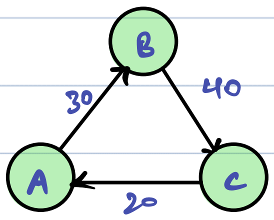
Case II: Based on weight of edge





↳ weighted graph

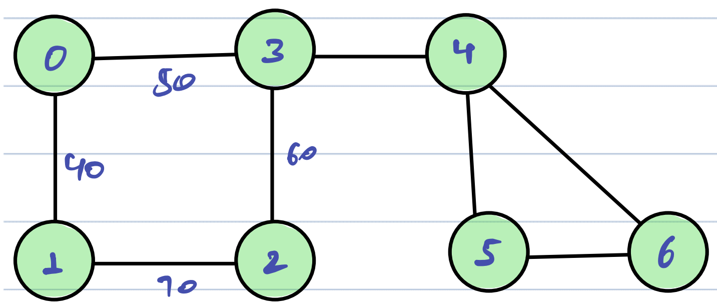
↳ unweighted graph



↳ directed weighted graph

Storing a graph

↗ undirected graph



input

$N=7$ → no. of nodes $M=8$ → no. of edges

3	4	20
1	2	
2	3	
4	6	
0	1	
4	5	
5	6	
0	3	

① Adjacency matrix representation

Graph

nbrs ⟷

Node ↕

$$\begin{array}{c|ccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
0 & 0 & \cancel{0}1 & 0 & \cancel{0}1 & 0 & 0 & 0 \\
1 & 1\cancel{0} & 0 & \cancel{0}1 & 0 & 0 & 0 & 0 \\
2 & 0 & 1\cancel{0} & 0 & \cancel{0}1 & 0 & 0 & 0 \\
3 & 1\cancel{0} & 0 & \cancel{0}1 & 0 & \cancel{0}\cancel{1}\,20 & 0 & 0 \\
4 & 0 & 0 & 0 & \cancel{1}\cancel{0}\,20 & 0 & \cancel{0}1 & \cancel{0}1 \\
5 & 0 & 0 & 0 & 0 & \cancel{0}1 & 0 & 1\cancel{0} \\
6 & 0 & 0 & 0 & 0 & 1\cancel{0} & \cancel{0}1 & 0
\end{array}$$

7x7

0 → disconnected

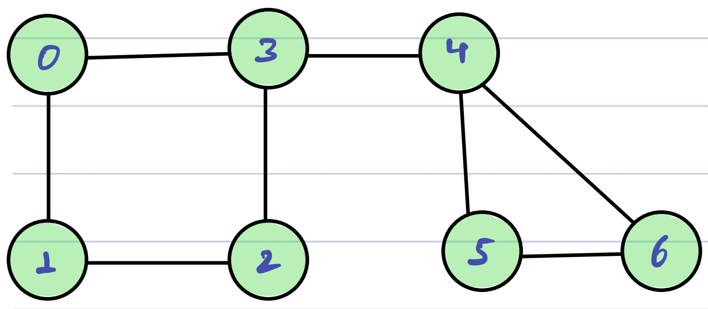
1 → connected

issues:

↳ 1. Space wastage

2. Say edge weight +ve / 0 / -ve, difficult to represent.

② adjacency list representation



i=0

→ no. of nodes

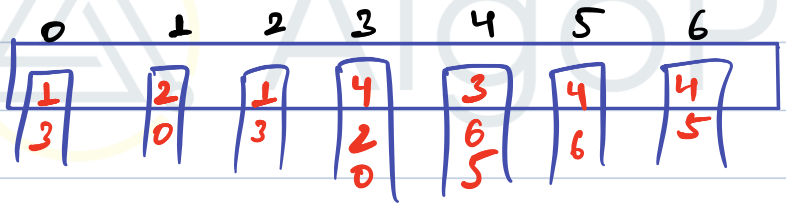
→ no. of edges

N=7 m=8

	0	1
0	3	4
1	1	2
2	2	3
3	4	6
4	0	1
	4	5
	5	6
	0	3

List < List < Integer > > graph = new ArrayList<>();

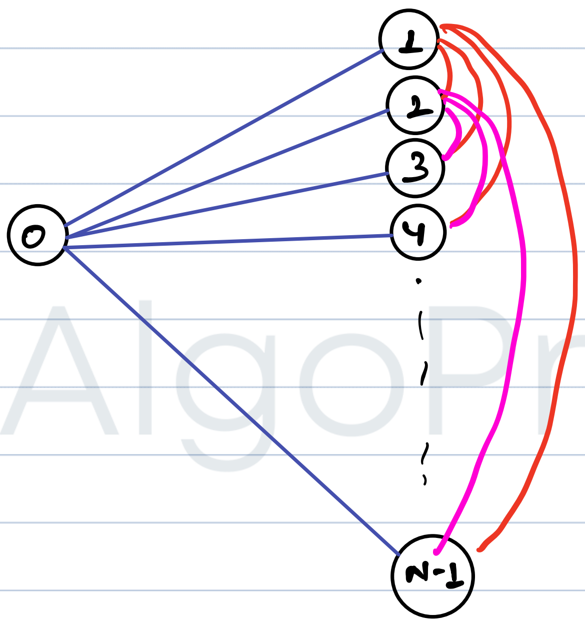
graph



```
main ( )  {
        Scanner scn = new Scanner(System.in);
           int n = scn.nextInt();
           int m = scn.nextInt();
            int [] [] edges = new int[m][2]

              for (int i=0; i<m; i++ ) {
                   edges[i][0] = scn.nextInt();
                   edges[i][1] = scn.nextInt();
              }

              Construction (n, m, edges);

        }

Construction (int n, int m, int[][]edges){

     List < List < Integer > > graph = new ArrayList<>();
          for (int i=0; i<n; i++) {
               graph.add( new ArrayList<>());
          }
          for (int i=0; i<m; i++ ) {
                  int u = edges[i][0];   → 3
                  int v = edges[i][1];   → 4
                 graph.get (u).add (v);
}        }       graph.get (v).add (u);
```

Break till 9:45 PM

Q) Find max number of edges possible if N nodes are present in graph.



$(N-1) + (N-2) + (N-3) + \dots\dots\dots + 1$

$\Downarrow$ $\frac{N*(N-1)}{2}$

↳ Graph-3 → fri $\xrightarrow{\text{Graph-2}}$ ✓ → 8:00AM

$\left\{\begin{array}{l} \text{mon} \xrightarrow{\text{miscell.}} ✓ \\ \text{wed} \to \text{off} \\ \text{fri} \to \text{off} \end{array}\right.$

Mon → levelup