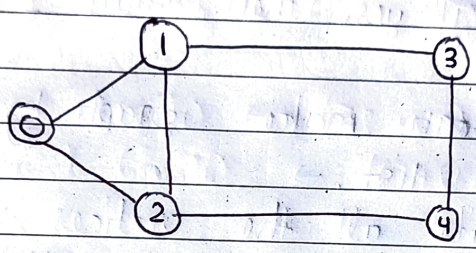
Date _____ Page _____

Day - 198

Graph - 2

* **Graph Representation:**



- There are two ways to store or represent graphs —
  - Adjacency matrix.
  - Adjacency List.

- When we get any graph, we will get many information about it —

  vertex : {0, 1, 2, 3, 4} → 5

  edge : {(0,1), (0,2), (1,2), (1,3), (2,4), (3,4)}

  adjacency matrix:

	0	1	2	3	4
0	0	1	1	0	0
1	1	0	1	1	0
2	1	1	0	0	1
3	0	1	0	0	1
4	0	0	1	1	0

Date _____ Page _____

⇒ For creating matrix, write all the vertex as ~~we~~ the write in the above matrix.

⇒ Then, check if there is an edge b/w that vertices if yes, then write 1 otherwise 0.

⇒ Also, we can make graph by using adjacency list.

⇒ Just write all the vertices then do the connection b/w the vertices i.e. if 1 then do connection otherwise not.

⇒ Suppose, if our graph is weighted undirected graph then instead of writing one (1), we will write the weight of that edge.

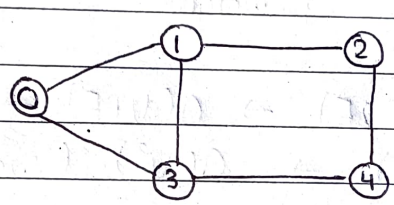
⇒ So, any no. either than 0 will show that there is a connection b/w that vertices with weight equals to that number.

⇒ If our graph is directed then only those edges will consider that are going from one vertex to another.

Date _____ Page _____

Time Complexity → $O(V^2)$
S.C → $O(V^2)$

* Adjacency List:



⇉ First create a array of all the vertices.
⇉ Then write all the vertices that we can go from that vertex.

0	→ 1 → 3
1	→ 0 → 2 → 3
2	→ 1 → 4
3	→ 0 → 1 → 4
4	→ 2 → 3

⇉ For storing , we will use vector.
⇉ vector<int> adjList[5];
⇉ So, here we have a adjList named vector or array of size 5 that pointing towards the vector of type int.

⇉ adjList[0].push_back(1);
adjList[1].push_back(0);

Date _____ Page _____

⇉ So, if we have a weighted graph then we ~~can~~ have to store weight too.

⇉ So, for that, we can use pair.

pair<int, int>
↙ vertex ↘ weight

Time : $O(V+2E) \rightarrow O(V+E) \rightarrow O(V^2)$ ↗ worst case

Space : $O(V+E) \rightarrow O(V^2)$ (worst case).

⇉ So, now, which is best to use, let is find out this by doing comparision.

	Adj. Matrix	Adj. List
Add Edge :	$O(1)$	$O(1)$
Remove Edge :	$O(1)$	$O(V)$
Edge exists :	$O(1)$	$O(V)$
Space Complexity :	$O(V^2)$	$O(V+E)$

⇉ We prefer Adj. Matrix if we have dense graph (no. of edges are very high) and Adj. List if we have sparse graph (no. of edges are very less).

⇉ We mostly time use Adj. List.

Date _____ Page _____

* Tree Vs Graph:

⇉ If we have n-node in tree then there will be (n-1) edges. But this is not the case in graph.

⇉ There is no loop in trees but in graph.

⇉ ⓪ ① → This can be a graph
② but not tree.

⇉ There is only one parent of every node in tree but not in graph.

In-degree → Coming edges.
Out-degree → Outgoing edges.