DAY - 02

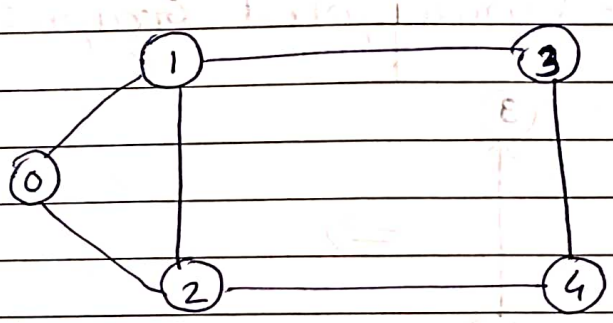
GoodLuck | Page No.
Date

* GRAPH REPRESENTATION

i] Adjacency Matrix
ii] Adjacency List

- To represent a graph using Adjacency matrix or adjacency list, we are given with the
  - vertices
  - edges

consider the below graph and the vertices and edges of the graph are given as:

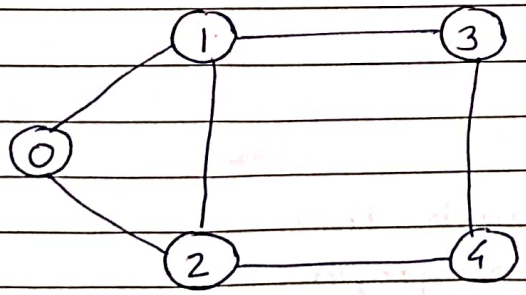


- vertices = {0, 1, 2, 3, 4}
- edges = {{0,1}, {1,2}, {0,2}, {1,3}, {2,4}, {3,4}}

1] ADJACENCY MATRIX :-

- representing the vertices and edges in a matrix

a] Undirected - Unweighted Graph :-



⇒

vertices →	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

no edge for 4→0

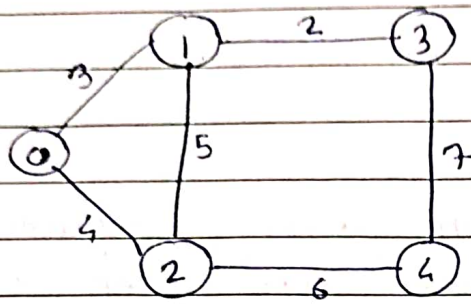
edge for 4→2

in undirected graph:

mark u-v = 1, v-u = 1 } for an edge between u and v

GoodLuck | Page No.
Date / /

b] Undirected - Weighted Graph :-



⇒

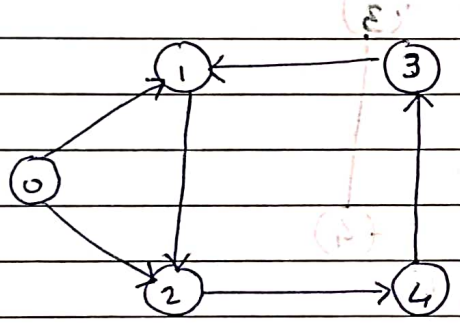
vertices → / ↓	0	1	2	3	4
0	0	3	4	0	0
1	3	0	5	2	0
2	4	5	0	0	6
3	0	2	0	0	7
4	0	0	6	7	0

weight for u-v

for undirected graph, mark (u,v) & (v,u),

~~mark~~ $u \to v$
$v \to u$ is in matrix for an edge between u, v

c] Directed ~~Gr~~ - Unweighted Graph :-



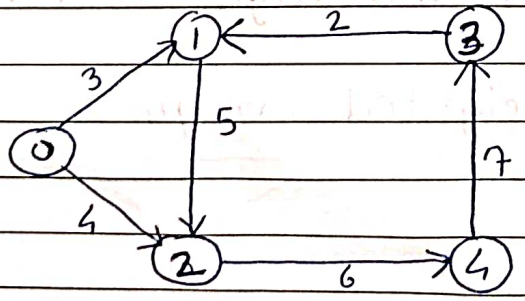
⇒

vertices → / ↓	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	0
2	0	0	0	0	1
3	0	1	0	0	0
4	0	0	0	1	0

edge for u → v

mark only (u,v) cell such that edge $u \to v$ is in graph

d] Directed - Weighted Graph :-



	0	1	2	3	4
0	0	3	4	0	0
1	0	0	5	0	0
2	0	0	0	0	6
3	0	2	0	0	0
4	0	0	0	7	0

weight for u→v

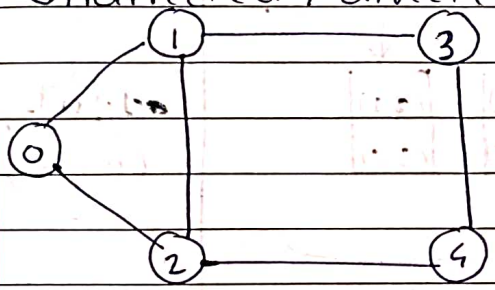
mark only (u,v) cell such that, edge $u \to v$ is in graph

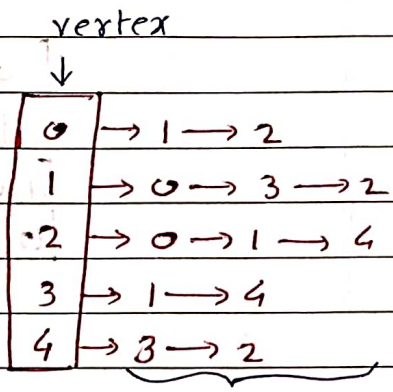
GoodLuck | Page No.
Date

- Time Complexity → $O(V^2)$
  - to create adj. matrix from given vertex and edges: For each cell in V×V matrix
- Space Complexity → $O(V^2)$
  - adj[v][v] matrix space

2] ADJACENCY LIST :-

a] Undirected / directed graph :-

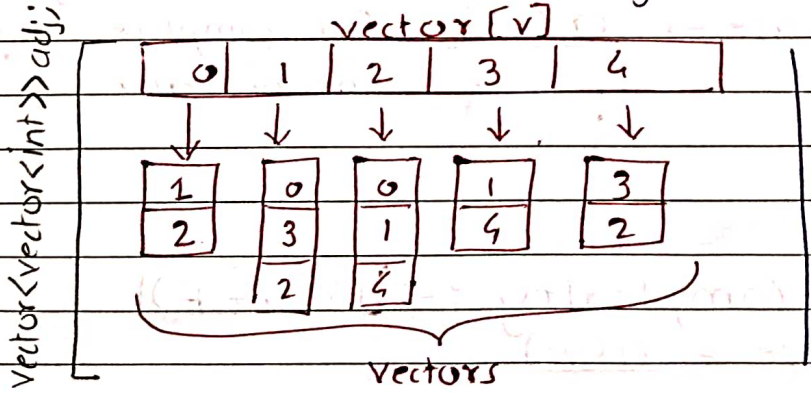




- In adjacency list, we represent the adjacent vertices of a vertex in list form
- each vertex stores it's adjacent vertex list

vertices → use array / vector

list can be → use array/vector or linked list

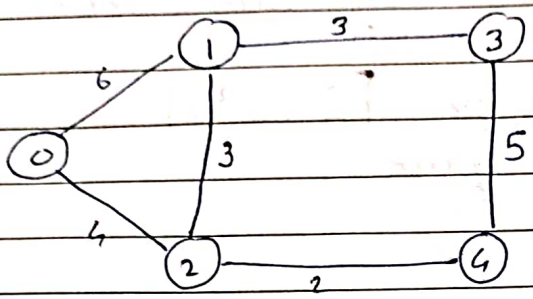


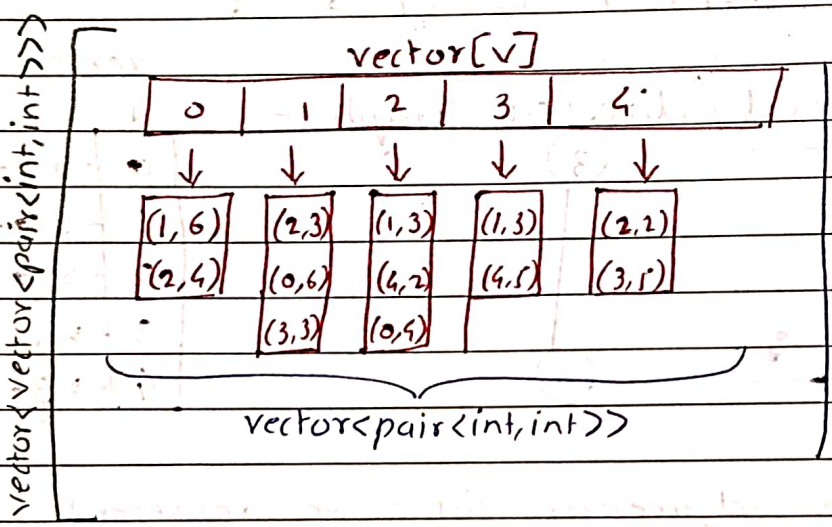
to store graph in adj. list =>

```
adj[u].push_back(v);
```

GoodLuck | Page No.
Date

b) Undirected / directed weighted graph :-





- Time complexity :- $O(V+E)$
  - For each vertex push-back its all edges
  - worst case → $O(V^2)$ · for complete graphs,
    no. of vertex = V, edges = $\frac{V*(V-1)}{2}$ approx. V^2

- Space complexity :- $O(V+E)$
  worst case = $O(V^2)$

GoodLuck | Page No.
Date

* ADJACENCY MATRIX V/S LIST :-

	ADJACENCY MATRIX	ADJACENCY LIST
i]	Adding edge takes : O(1) (vector[u][v] = 1)	Adding edge takes : O(1) (vector push_back : O(1) → avg, O(n) → worst)
ii]	Removing edge :: O(1) (vector[u][v] = 0)	Removing edge : O(v) (v is the vertices to be shifted to remove vertex)
iii]	Edge exists : O(1) (vector[u][v] == 1 ?)	Edge exists : O(v) (check all adj nodes of u to find v)
iv]	Space complexity : $O(V^2)$	Space complexity : O(V+E) worst case : $O(V^2)$ → comp. graph
v]	Preferred for dense graph: (no. of edges is high, almost equal to complete graph)	Preferred for Sparse graph: (no. of edges is low)