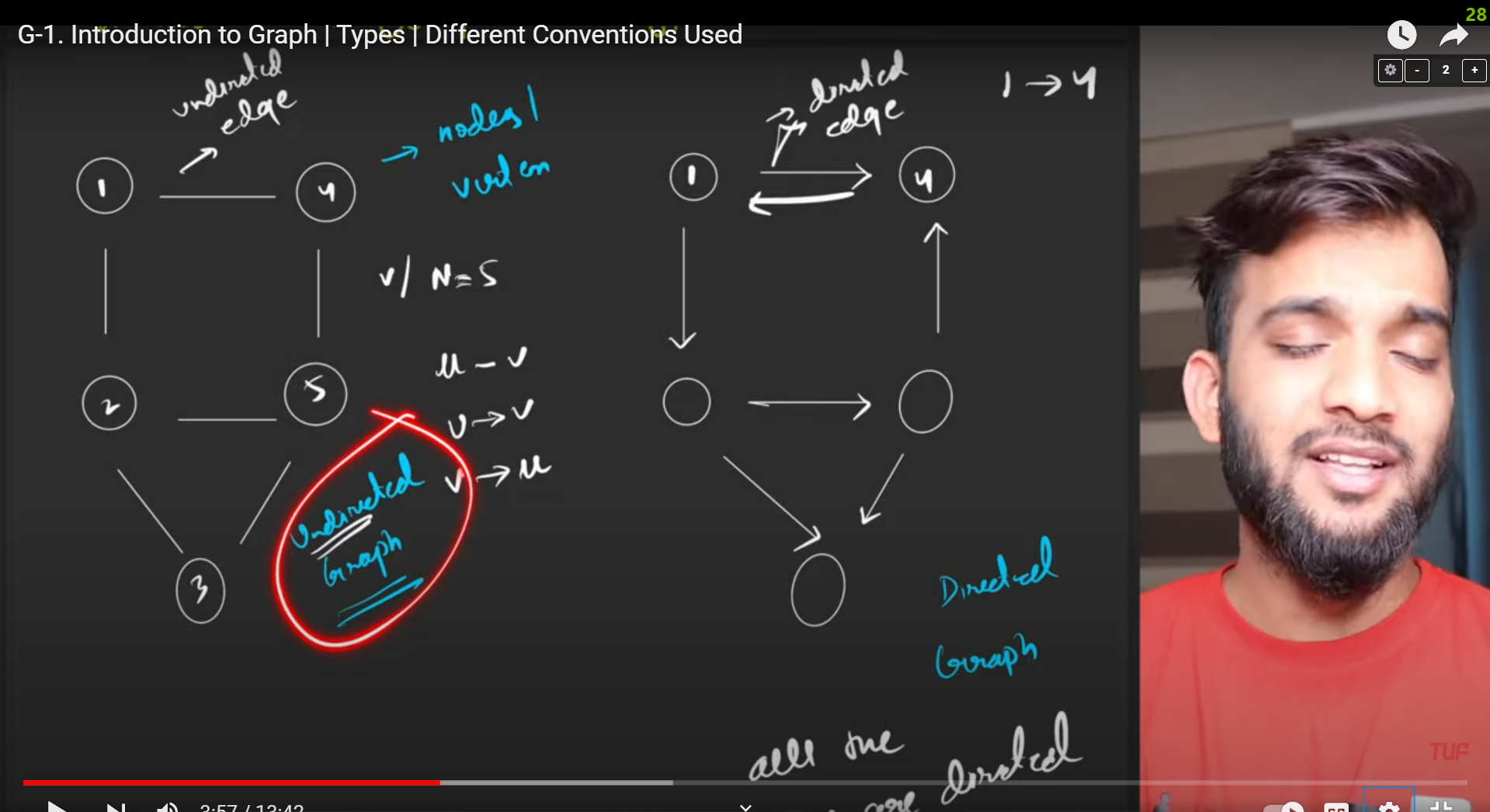
**Striver Graphs**



Cycle in a graph means that if you start from a node and can reach back the same node.

Binary tree can also be called as a graph.

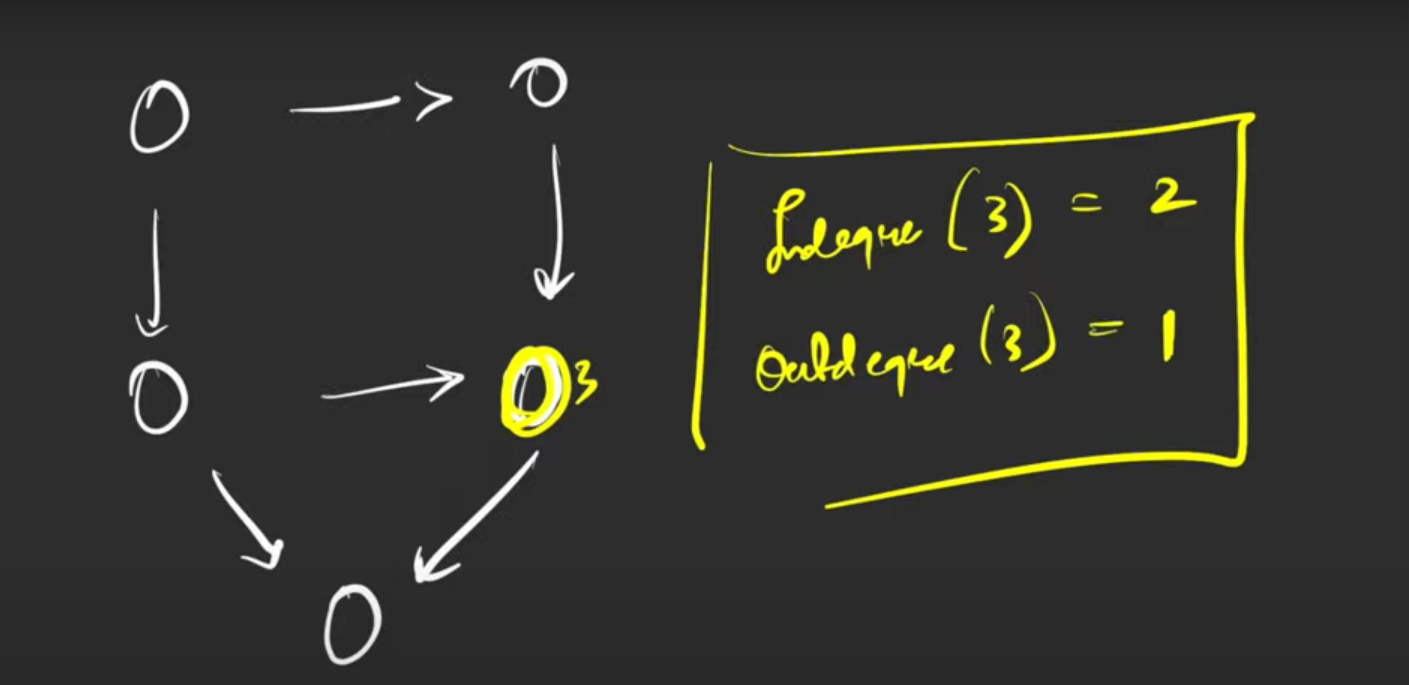
A node can not appear twice in a path.

Degree of a graph: For an undirected graph the number of edges attached to it is called as its degree.

**Also :**

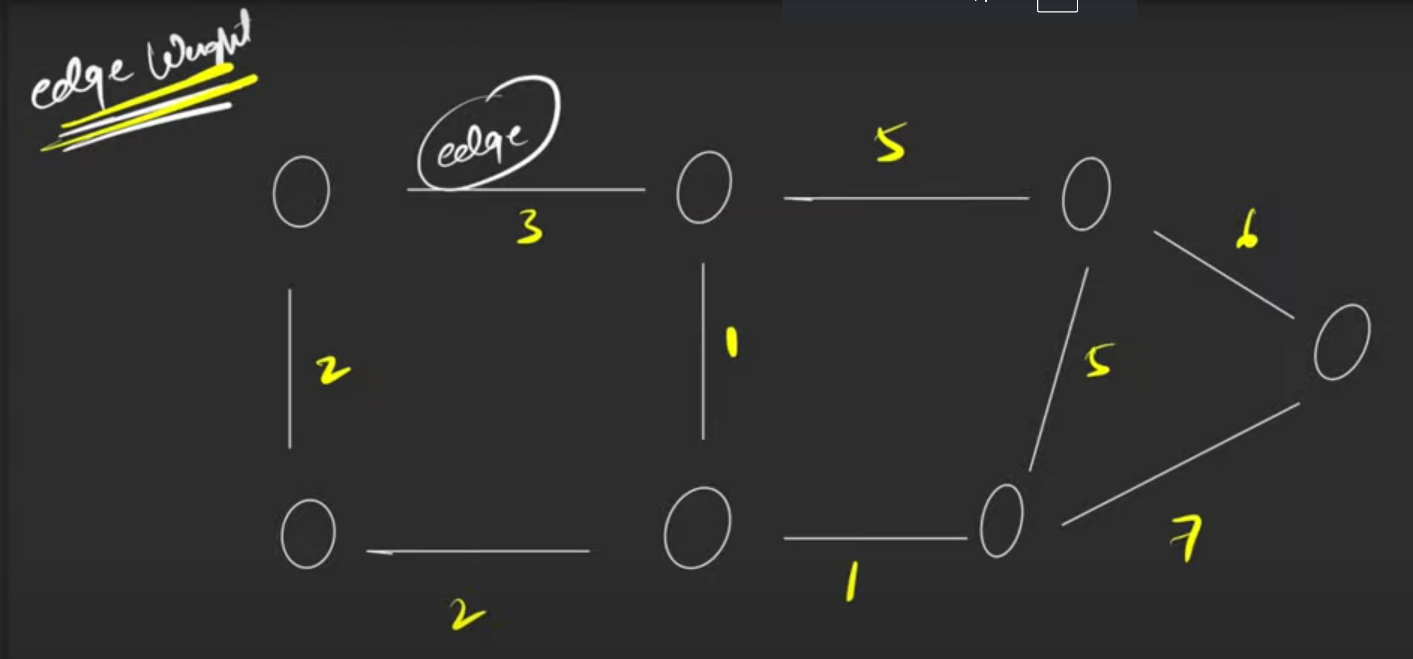
**Total degree of a graph = 2 X edges of the graph**

(because every edge is associated to 2 nodes).

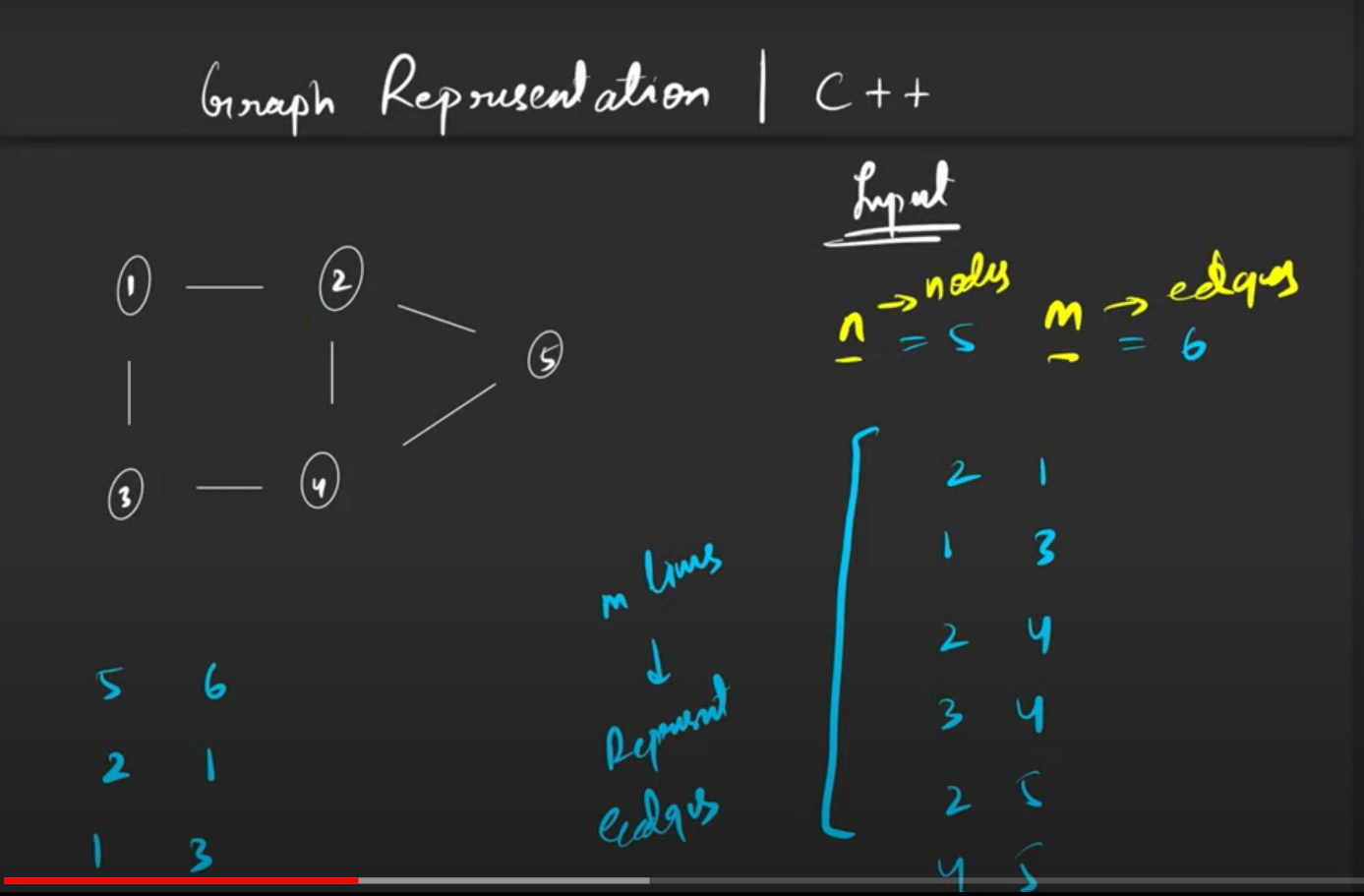


Indegree : the number of incoming edges to a node.

Outdegree : the number of edges outgoing from a node.



If weights are not assigned then we assume unit weight.

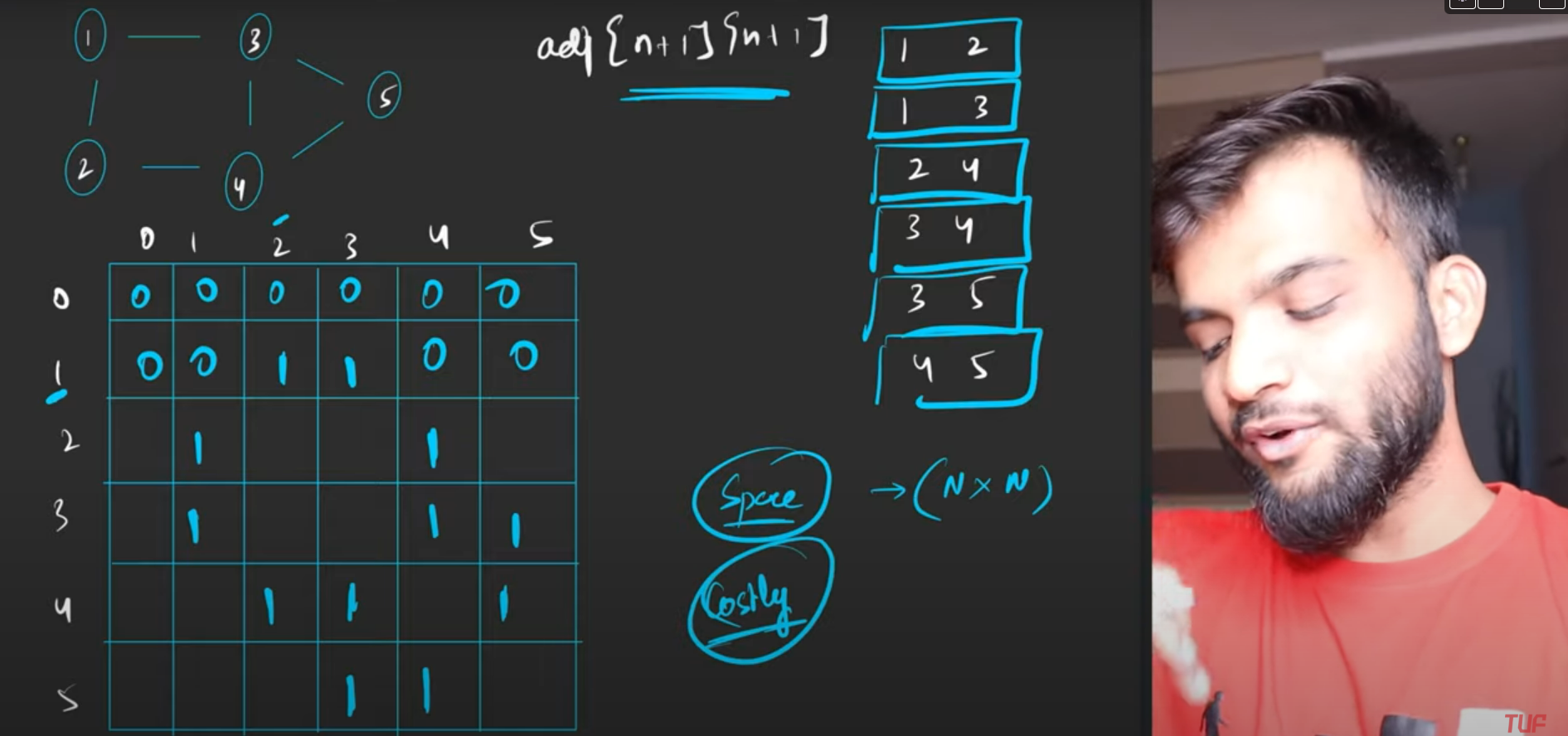


Graph is represented as they will give n(nodes) and m(edges) and then m lines depicting the pair between which an edge is there and they can be in any order.

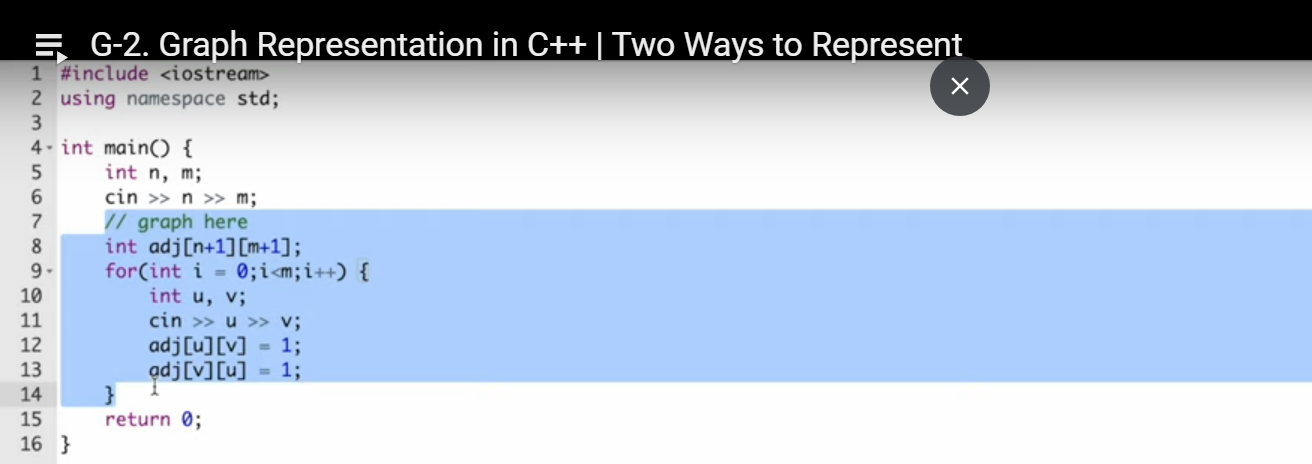
We store graph in 2 ways:

1. Matrix Way **(Adjacency Matrix)**
2. List Way

1. Matrix Way



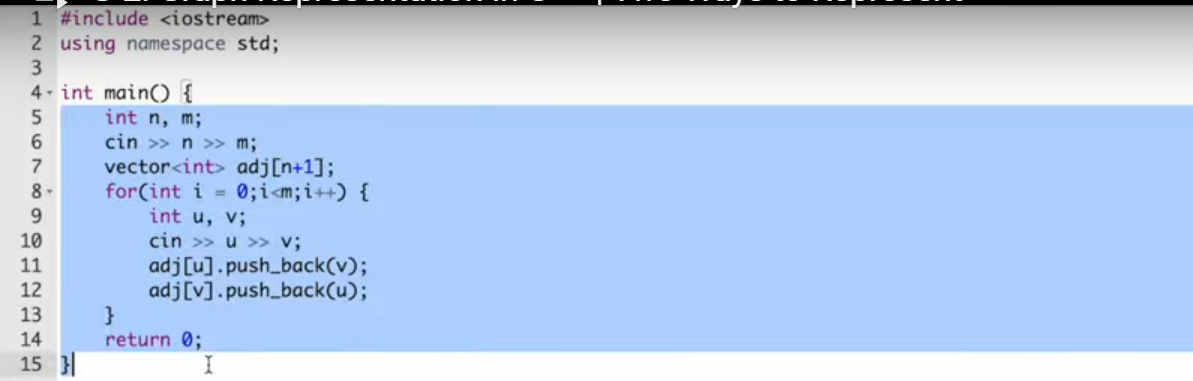
We make a matrix and it we mark the intersection of the pair as 1 and all rest are 0.



2. List Way:



So we create a vector array of size n+1 and we store each vertex’s neighbours in that list. The space complexity in this case is O(2XE) as we store each edge twice rather than O(N2) in case of matrix method.



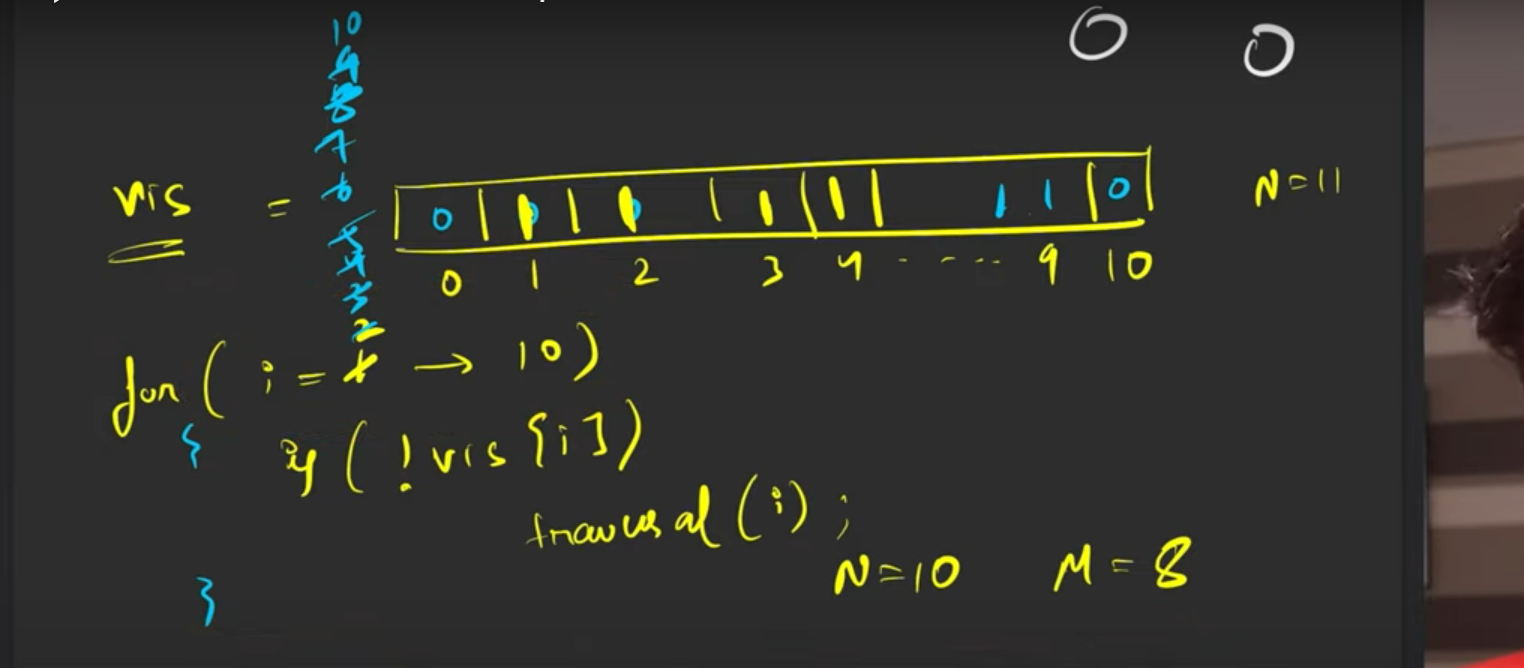
If you want to store weights as well then in case of matrix method wherever there is an edge instead of putting a 1 over there put the weight of that edge.

In the case of List method instead of just storing the neighbours vertex store its weight as well and store it in the form of pair.

So now we will make vector<pair<int, int>> for storing edges in graphs.



Connected components in graph: the above picture is 1 single graph with 4 components.



If a graph has multiple components, then we use the visited array approach to keep track of unvisited vertices and then we call traversal algorithm on those vertices.