DAA Lab (Week-06)

Notes By - Deepak Uniyal

Consider the following input format in the form of adjacency matrix for graph based questions (directed/undirected/weighted/unweighted graph).

Input Format: Consider example of below given graph in Figure (a).
A boolean matrix AdjM of size V X V is defined to represent edges of the graph. Each edge of graph is represented by two vertices (start vertex u, end vertex v). That means, an edge from u to v is represented by making AdjM[u,v] and AdjM[v,u] = 1. If there is no edge between u and v then it is represented by making AdjM[u,v] = 0. Adjacency matrix representation of below given graph is shown in Figure (b). Hence edges are taken in the form of adjacency matrix from input. In case of weighted graph, an edge from u to v having weight w is represented by making AdjM[u,v] and AdjM[v,u] = w.

Adj. list, Adj. matrix

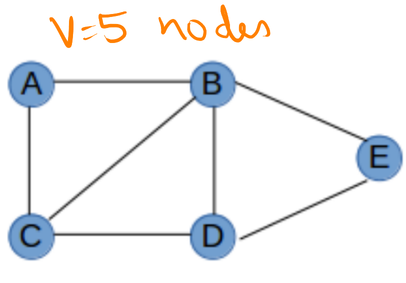
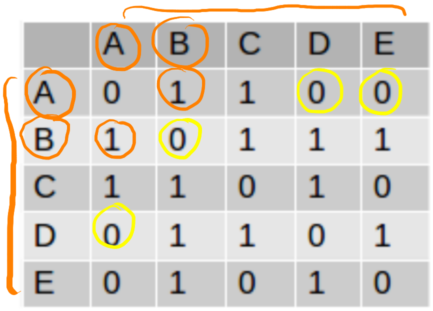
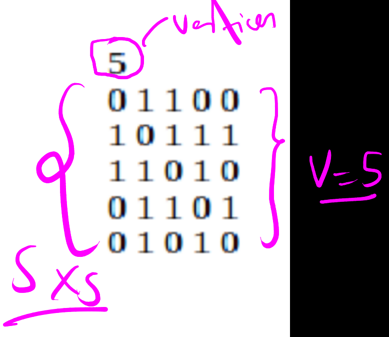


Figure (a)



	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	1	1
C	1	1	0	1	0
D	0	1	1	0	1
E	0	1	0	1	0

Figure (b)



vertices

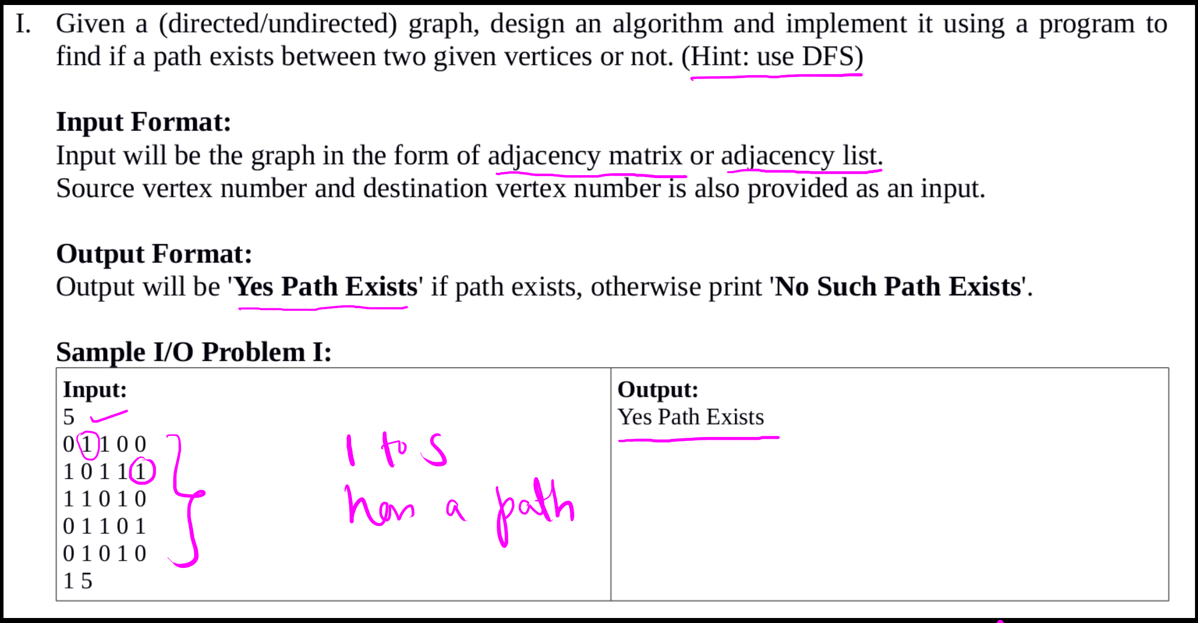
```
5
0 1 1 0 0
1 0 1 1 1
1 1 0 1 0
0 1 1 0 1
0 1 0 1 0
```

V=5

5x5

Figure (c)

Input format for this graph is shown in Figure (c).
First input line will obtain number of vertices V present in graph.
After first line, V input lines are obtained. For each line i in V, it contains V space separated boolean integers representing whether an edge is present between i and all V.



I. Given a (directed/undirected) graph, design an algorithm and implement it using a program to find if a path exists between two given vertices or not. (Hint: use DFS)

Input Format:
Input will be the graph in the form of adjacency matrix or adjacency list.
Source vertex number and destination vertex number is also provided as an input.

Output Format:
Output will be '**Yes Path Exists**' if path exists, otherwise print '**No Such Path Exists**'.

Sample I/O Problem I:

Input: 5 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 1 5	**Output:** Yes Path Exists

1 to 5 has a path

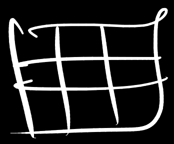
{ Directed / Undirected } { weighted / unweighted }

Time - $O(V^3)$

Space - $O(V^2)$

* floyd warshall algo. — all pairs

5 nodes

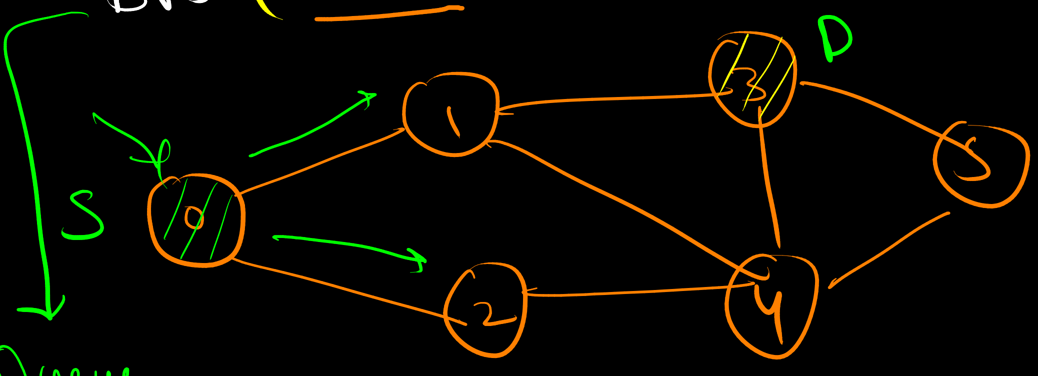


* if there is a direct edge

* if a path exists between i & j

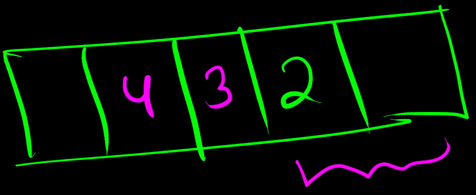
BFS (Breadth first search)

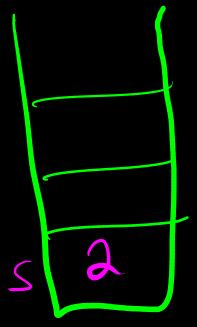
DFS (Depth first search)



Stack

Queue





isPathBFS (G , S, $\underline{D}$)

1	2	3	4	5
1	1~~0~~	0	0	0

```
Queue - Q
Q.enqu (s) , mark S as visited
while (Q is not empty)
        v= Q.deque()
        for all neighbours u of v :
            if u==D :
                return true
            if u is not visited:
                Q.enque(u)
                mark u as visited
return false
```

isPathDFS (G , S, D)

1	2	3	4	5
1~~0~~	0	0	0	0

```
Stack st
st.push (S) , mark s as visited
while( st is not empty)
        v = st.top()
        st.pop()
        for all neighbours u of v:
            if u==D:
                return true
            if u is not visited:
                st.push(u)
                mark u as visited
return false
```