**Transitive closure of a Graph: -**

**Medium** Accuracy: **29.1%** Submissions: **33K+** Points: **4**

Given a directed graph, determine whether a vertex **j** is reachable from another vertex **i** for all vertex pairs **(i, j)** in the given **graph**. Here, vertex **j** is reachable from another vertex **i**means that there is a path from vertex **i**to **j.** The reachability matrix is called the **transitive closure of a graph**. The directed graph is represented by an **adjacency matrix**where there are **N** vertices.

**Example 1:**

**Input:** N = 4

graph = {{1, 1, 0, 1},

  {0, 1, 1, 0},

  {0, 0, 1, 1},

  {0, 0, 0, 1}}

**Output:** {{1, 1, 1, 1},

  {0, 1, 1, 1},

  {0, 0, 1, 1},

  {0, 0, 0, 1}}

**Explanation:**The output list shows the reachable indexes.

**Example 2:**

**Input:** N = 3

graph = {{1, 0, 0},

  {0, 1, 0},

  {0, 0, 1}}

**Output:** {{1, 0, 0},

  {0, 1, 0},

  {0, 0, 1}}

**Explanation:**The output list shows the reachable indexes.

**Your Task:**  
You do not need to read input or print anything. Your task is to complete the function **transitiveClosure()** which takes an integer **N** and a 2D array **graph**(adjacency matrix of the graph)as **input parameters**and returns the **transitive closure**of the graph in the form of **2D array**.

**Expected Time Complexity:** O(N3)  
**Expected Auxiliary Space:** O(N2)

**Constraints:**  
1 ≤ N ≤ 100

**Code: -**

//{ Driver Code Starts

// Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

// Back-end complete function Template for C++

class Solution{

public:

vector<vector<int>> transitiveClosure(int N, vector<vector<int>> graph)

{

// code here

for(int k = 0; k<N; ++k){

for(int i = 0; i<N; ++i){

for(int j = 0; j<N; ++j){

if(i == j)

graph[i][j] = 1;

graph[i][j] = graph[i][j] or (graph[i][k] and graph[k][j]);

}

}

}

return graph;

}

};

//{ Driver Code Starts.

int main(){

int t;

cin>>t;

while(t--){

int N;

cin>>N;

vector<vector<int>> graph(N, vector<int>(N, -1));

for(int i = 0;i < N;i++)

for(int j=0;j<N;j++)

cin>>graph[i][j];

Solution ob;

vector<vector<int>> ans = ob.transitiveClosure(N, graph);

for(int i = 0;i < N;i++)

{for(int j = 0;j < N;j++)

cout<<ans[i][j]<<" ";

cout<<"\n";}

}

return 0;

}

// } Driver Code Ends

**T.C: - O(N3)**

**S.C: - O(N2)**