# Assignment - 2

Harsh Dangi
1918351
09

```
#include < climit >
#include < list >
# include <vector>
# include < iostream>
using namespace std;

class graph {

        int n;
        int ~~**~~ edge;
        int count;

        public:
                graph (int n) {
                        count =0;
                        then ->n =n;
                        edge = new int *[n];
                        for(int i =0; i<n ; i++){

                                edge [i ] = new int[n];
        for(int j=0, j<n, j++){
                edge[i][j]=0;
        }
    }

}
void addedge(int a, int b){
            edge[a][b]=1;
            count++;
}

bool    has_edge(int a, int b){
        if(edge[a][b]==1)
                return true;

        else return false;
}

int count_edge(){
            int count_edge = count;
            return count_edge;
}
int findMatrix(int *distance, bool(*visited, int n){

            int minvertex=-1;
            for(int i=0; i<n; i++){
                if(!visited[i] && (minvertex == -1) ||
                                distance[i] < distance[minve
                    minvertex = i;


}
std::vector<int> bfs(int n) {
        vector<int> distance;
        bool * visited = new bool[n];

        for(int i=0; i<n; i++){
                distance[i]=INT_MAX;
                visited[i]=false;
}       }
    distance[0]=0;

     for(int i=0; i<n; i++){
        int minvertex = findminvertex( distance,
                                    visited, n);

        visited[minvertex] = true;

        for(int j=0; j<n; j++){
            if(edge[minvertex][j]!=0 &&
                !visited[j]){
                int dist = distance[minvertex] +
                        edges[minvertex][j];

                    if(dist < distance[j]){
                            distance[j]=dist;
                    }
                }
            }
        }
    return distance;
}
```

```
bool dfs(int n){
        bool cycle = false;
        for(int i=0; i<n-2; i++){
                for(int j=i+1; j<v-1; j++){
                        for(int k=j+1; k<v; k++){
                                if(edge[i][j] && edge[j][k] &&
                                        edge[k][i])
                                        cycle = true;

                        }

                }

        }
        return cycle;
}

bool haspath(bool *visited, int s, int e){

        if(s==e)
                return true;

        visited[s]= true;
        for(int i=0; i<n; i++){
                if(edge[s][i]==1){
                        if(visited[i])
                                continue;


                bool temp= haspath(edge, n, visited, i, e);
                if(temp)
                        return temp;

        }
    }visited[s] = false;
    return false;
```

```
bool is_connected (int a, int b){
    bool * visited = new bool[n];
    for(int i=0; i<n; i++){
            visited[i]=0;

    for(int i=0;i<n;i++) {
        if(!visited[i]&&edge[a][i]==1)
                bool ans = haspath(visited, i,b);

            if(ans)
                return true;
    }
    delete[] visited;
    return false;
}
};
```