DATA STRUCTURE

LAB

BY,

NIHAL K

TKM20MCA-2026

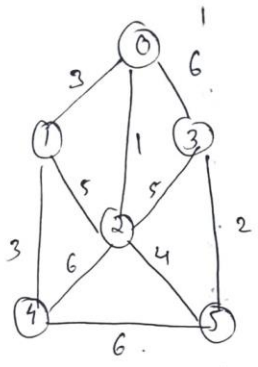
Question 1:

Nihal.k

TKM20MCA - 2026

DS Lab.

1) Develop a Program to Generate a minimum Spanning tree using kruskal algorithm for the given graph. and Compute the total Cost.



Algorithm

1) for each vertex $v \in V[G]$
2) do make findeset(v).
3) Sort the edges of E in non decreasing order by cost.
4) For each edge $(u,v) \in E$, taken in non decreasing order by cost.
   do if findeset(u) $\neq$ findeset(v)
   then $A \leftarrow A \cup \{(u,v)\}$
   join(u,v)
5) return A.

~~Ans~~

Finding cost of each edge :

		Cost
1	$\Rightarrow (0,2)$	1
2	$\Rightarrow (3,5)$	2
3	$\Rightarrow (0,1)$	3
4	$\Rightarrow (1,4)$	3
5	$\Rightarrow (2,5)$	4
6	$\Rightarrow (1,2)$	5
7	$\Rightarrow (2,3)$	5
8	$\Rightarrow (0,3)$	6
9	$\Rightarrow (2,4)$	6
10	$\Rightarrow (4,5)$	6.

Minimum Cost = $1+2+3+3+4$

$= 13$

Code:

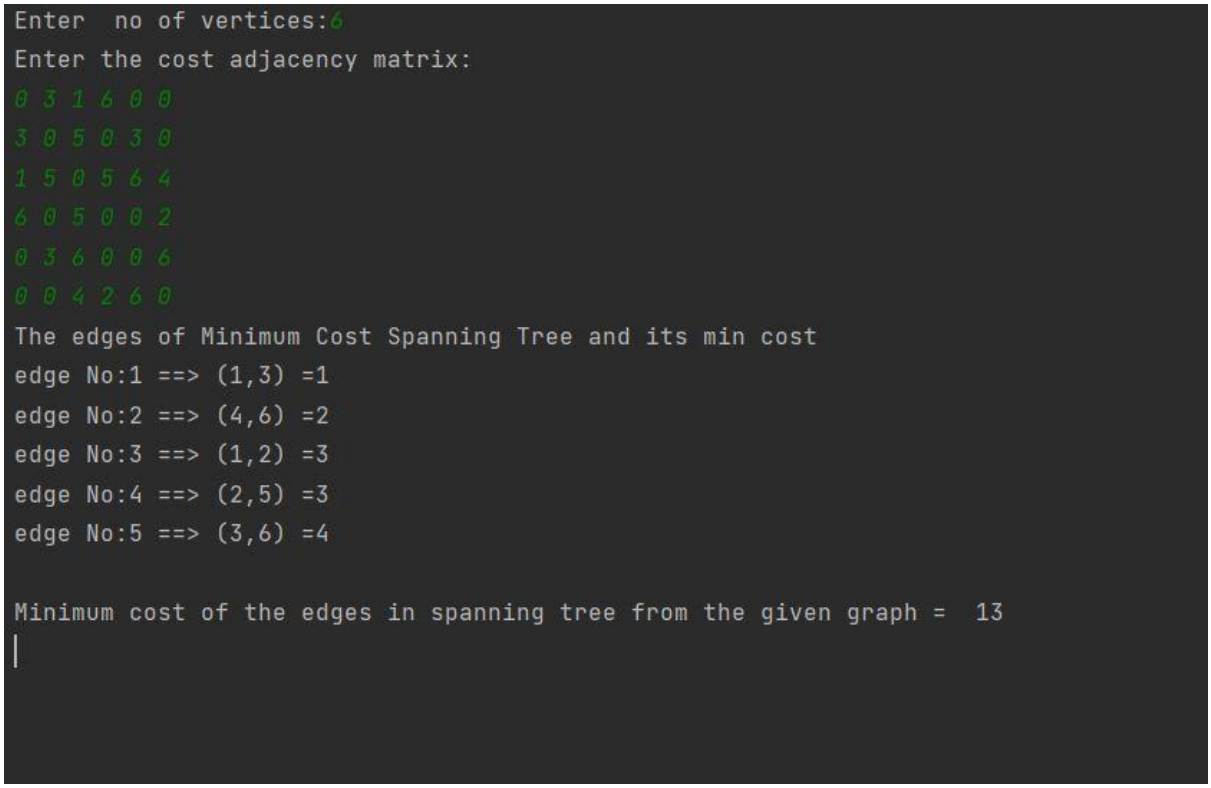
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int findeset(int);
int join(int,int);
void main()
{

   printf("Enter  no of vertices:");
   scanf("%d",&n);
   printf("Enter the cost adjacency matrix:\n");
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
      {
         scanf("%d",&cost[i][j]);
         if(cost[i][j]==0)
            cost[i][j]=999;
      }
   }
   printf("The edges of Minimum Cost Spanning Tree and its min cost\n");
   while(ne < n)
   {
      for(i=1,min=999;i<=n;i++)
      {
         for(j=1;j <= n;j++)
```

```
        {
            if(cost[i][j] < min)
            {
                min=cost[i][j];
                a=u=i;
                b=v=j;
            }
        }
    }
    u=findeset(u);
    v=findeset(v);
    if(join(u,v))
    {
        printf("edge No:%d ==> (%d,%d) =%d\n",ne++,a,b,min);
        mincost +=min;
    }
    cost[a][b]=cost[b][a]=999;
  }
  printf("\nMinimum cost of the edges in spanning tree from the given graph =  %d\n",mincost);
  getch();
}
int findeset(int i)
{
  while(parent[i])
    i=parent[i];
  return i;
}
int join(int i,int j)
{
  if(i!=j)
  {
```

```
        parent[j]=i;
        return 1;
    }
    return 0;
}
```

Output :

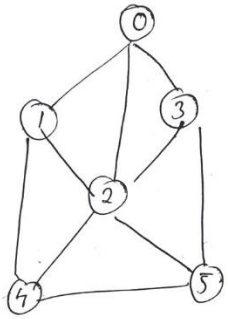


Question 2:

2) Develop a program to implement DFS and BFS

2) Develop a program to implement DFS and BFS.

Consider graph.



adjacency Matrix

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

Algorithm for BFS

1) Start
2) Add any one of the graphs vertices at the back of a queue
3) Take the front item of the queue and add it to the visited list.
4) Create a list of that vertex's adjacent nodes Add the ones which aren't visited list to the back of the queue.
5) Keep repeating Step 3 and 4 until the queue is empty

Algorithm for DFS

1) Start
2) Add any one of the graphs vertices and add to top of the Stack.
3) Take the top item of the Stack and add it to the visited list
4) Creat a list of vertex's adjacent nodes. add the ones which aren't in the visited list to the top of the Stack.
5) Keep repeating 3 and 4 till Stack is empty.

DFS

Code :

```
#include<stdio.h>
int n,g[100][100],v[100],s;
void dfs(int s)
{
   int i;
   v[s]=1;
   printf("Visited Node = %d\n",s);
   for(i=0;i<n;i++){
      if(v[i]==0&&g[s][i]==1)
      {
         dfs(i);
      }
   }
}
int main()
{
   int i,j;
   printf("Enter the no of vertices\n");
   scanf("%d",&n);
   printf("Enter the adjacency Matrix : \n");
   for(i=0;i<n;i++){
      for(j=0;j<n;j++){
         scanf("%d",&g[i][j]);
      }
   }
```

```
    for(i=0;i<n;i++)
    {
       v[i]=0;
    }
    printf("Enter the starting vertex");
    scanf("%d",&s);
    dfs(s);
}
```

Output :

```
Enter the no of vertices
6
Enter the adjacency Matrix :
0 1 1 1 0 0
1 0 1 0 1 0
1 1 0 1 1 1
1 0  1 0 0 1
0 1 1 0 0 1
0 0 1 1 1 0
Enter the starting vertex0
Visited Node = 0
Visited Node = 1
Visited Node = 2
Visited Node = 3
Visited Node = 5
Visited Node = 4

Process finished with exit code 0
```

BFS

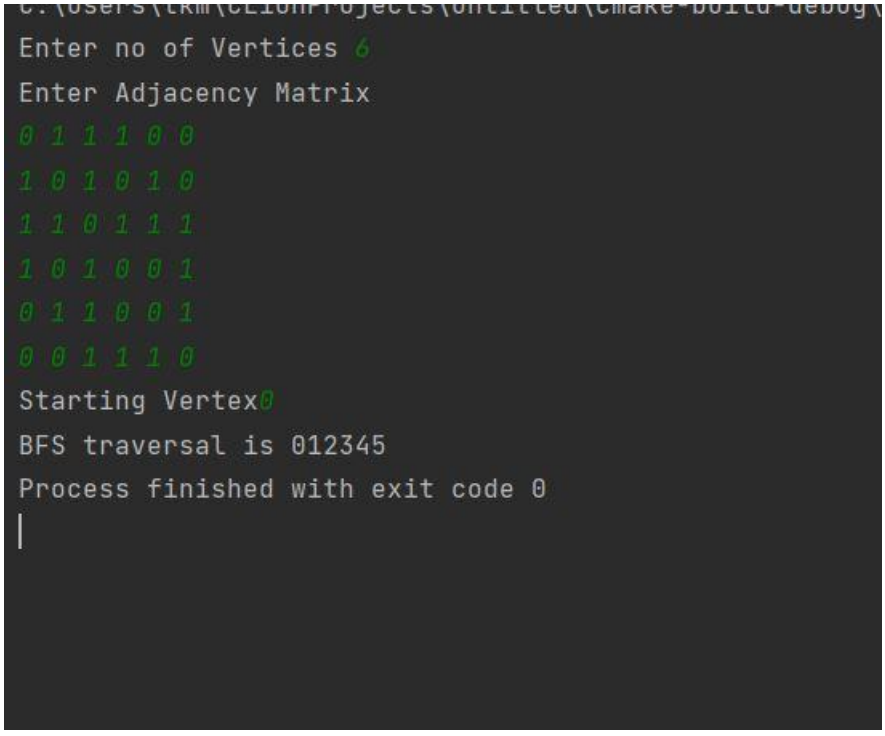
Code :

```
#include<stdio.h>
int n,g[20][20],v[20],a[20],f=-1,r=-1,s;
void bfs(int s)
{
   int i=0;
   for(i=0;i<n;i++)
   {
      if(v[i]==0 && g[s][i]==1){
         r=r+1;
         a[r]=i;
         v[i]=1;
         printf("%d",i);
      }
   }
   f=f+1;
   if(f<=r){
      bfs(a[f]);
   }
}
int main(){
   int i,j;
   printf("Enter no of Vertices");
   scanf("%d",&n);
   printf("Enter Adjacency Matrix");
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
```

```
        scanf("%d",&g[i][j]);
    for(i=0;i<n;i++)
       v[i]=0;
    printf("Starting Vertex");
    scanf("%d",&s);
    f=r=0;
    a[r]=s;
    v[s]=1;
    printf("BFS traversal is %d",s);
    bfs(s);
    if(r!=n-1)
       printf("Not possible");

}
```

Output :



GIT LINK

https://github.com/nihalnhk/DSLABEXAM