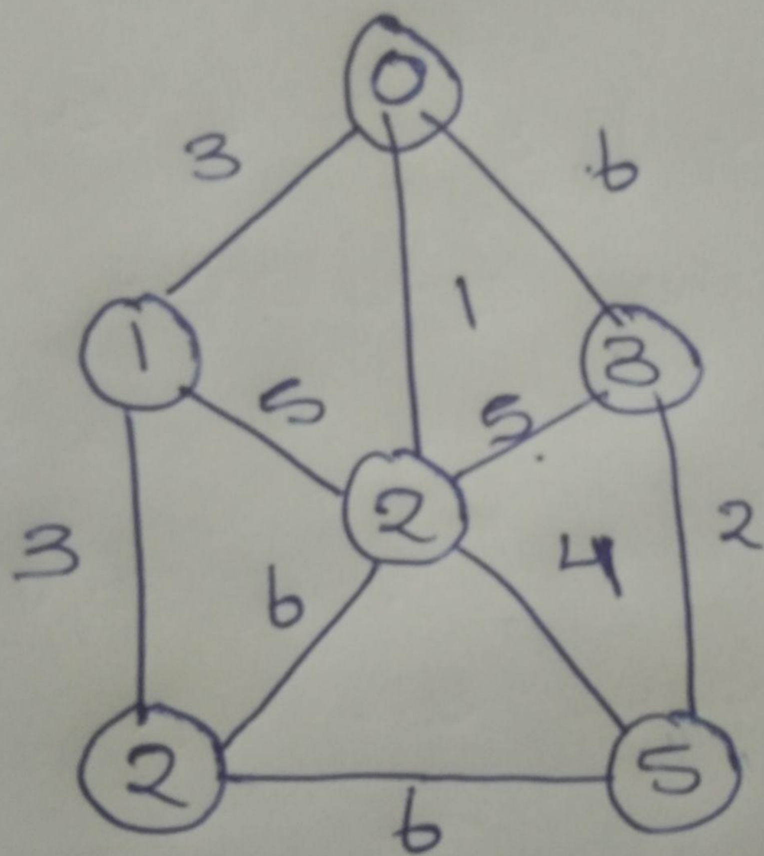
MAYURI.M
ROLLNO:MCA 222
REG NO:TKM20MCA-2022

DATA STRUCTURE LAB

GITHUB LINK:https://github.com/mayurim20/DSLab.git

1.DEVELOP A PROGRAM TO GENERATE MINIMUM SPANNING TREE USING KRUSKAL'S ALGORITHM OF THE GIVEN GRAPH AND COMPUTE THE TOTAL COST


0
3
6
1
1
3
5
5
2
3
2
6
4
2
5
6

ALGORITHM:

KRUSKAL(G):
A = ∅
For each vertex v ∈ G.
V: MAKE-SET(v)
For each edge (u, v) ∈ G.
E ordered by increasing order by weight(u, v):
 if FIND-SET(u) ≠ FIND-SET(v):
A = A ∪ {(u, v)}
UNION(u, v)
return A

* KRUSKAL (G):

1) $A = \phi$
2) for each vertex $v \in G$.
3) V: MAKE-SET (V)
4) for each edge $(u,v) \in G$
5) E ordered by increasing order by weight (u,v):
6) IF FIND-SET (U) $\neq$ FIND-SET (V):
7) $A = A \cup \{(u,v)\}$
8) UNION (u,v)
9) ~~return (u,v)~~
   return A.

PROGRAM CODE:

```
#include<stdio.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 find(int);
   int uni(int,int);
   void main()
   {
printf("\n\t Implementation of Kruskal's algorithm\n");
       printf("\nEnter the no. of vertices:");
       scanf("%d",&n);
       printf("\nEnter 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 are\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=find(u);
          v=find(v);
          if(uni(u,v))
          {
             printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
             mincost +=min;
```

```
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("\n\tMinimum cost = %d\n",mincost);

  }
int find(int i)
  {
    while(parent[i])
    i=parent[i];
    return i;
  }
  int uni(int i,int j)
  {
    if(i!=j)
    {
      parent[j]=i;
      return 1;
    }
    return 0;
  }
```

OUTPUT:

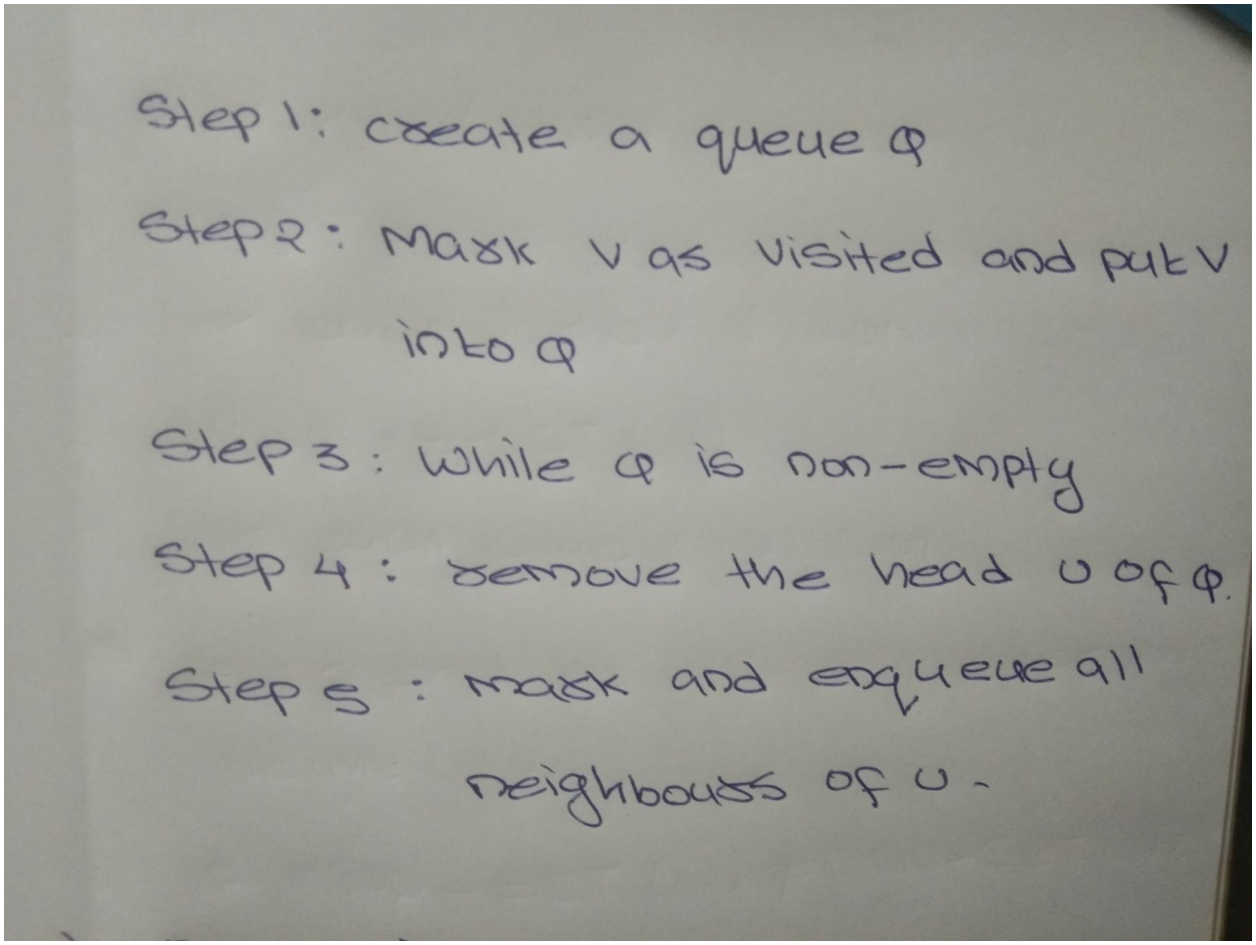
```
mayurim@mayurim-VirtualBox:~/Desktop/Dslab/Dslab/lab$ gcc -o kru kru.c
mayurim@mayurim-VirtualBox:~/Desktop/Dslab/Dslab/lab$ ./kru

        Implementation of Kruskal's algorithm

Enter the no. of vertices:6

Enter the cost adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,3) =1
2 edge (4,6) =2
3 edge (1,2) =3
4 edge (2,5) =3
5 edge (3,6) =4

        Minimum cost = 13
```

2.DEVELOP A PROGRAM TO IMPLEMENT DFS AND BFS

a)BFS

ALGORITHM

1. Create a queue Q
2. Mark v as visited and put v into Q
3. while Q is non-empty
4. remove the head u of Q
5. mark and enqueue all (unvisited) neighbours of u

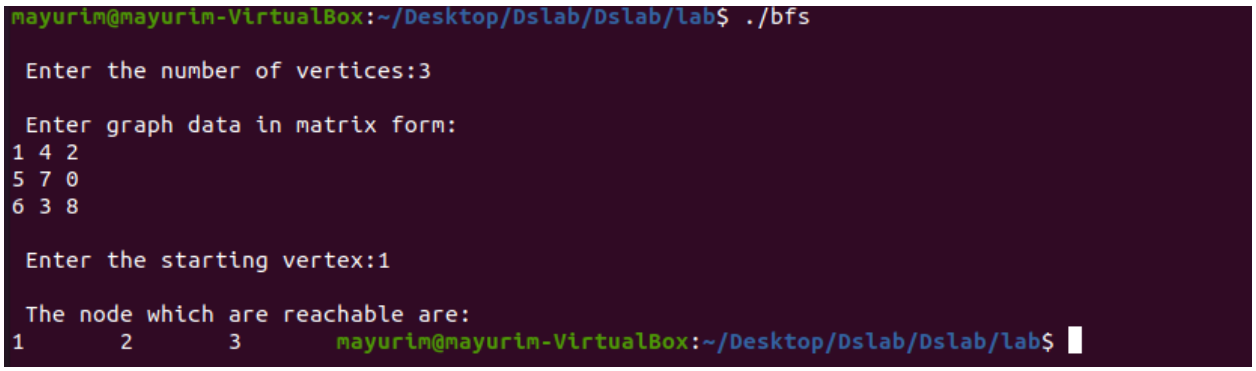


PROGRAM CODE

```
#include<stdio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
```

```
void bfs(int v)
 {
 for(i = 1; i <= n; i++)
 if(a[v][i] && !visited[i])
 q[++r] = i;
 if(f <= r) {
 visited[q[f]] = 1;
 bfs(q[f++]);
 }
}
int main()
{
 int v;
 printf("\n Enter the number of vertices:");
 scanf("%d", &n);
 for(i=1; i <= n; i++)
  {
 q[i] = 0;
 visited[i] = 0;
 }
 printf("\n Enter graph data in matrix form:\n");
 for(i=1; i<=n; i++)
  {
 for(j=1;j<=n;j++)
  {
 scanf("%d", &a[i][j]);
 }
 }
 printf("\n Enter the starting vertex:");
 scanf("%d", &v);
 bfs(v);
 printf("\n The node which are reachable are:\n");
 for(i=1; i <= n; i++)
  {
 if(visited[i])
 printf("%d\t", i);
 else
  {
 printf("\n Bfs is not possible. Not all nodes are reachable");
break;
 }
 }
}
```

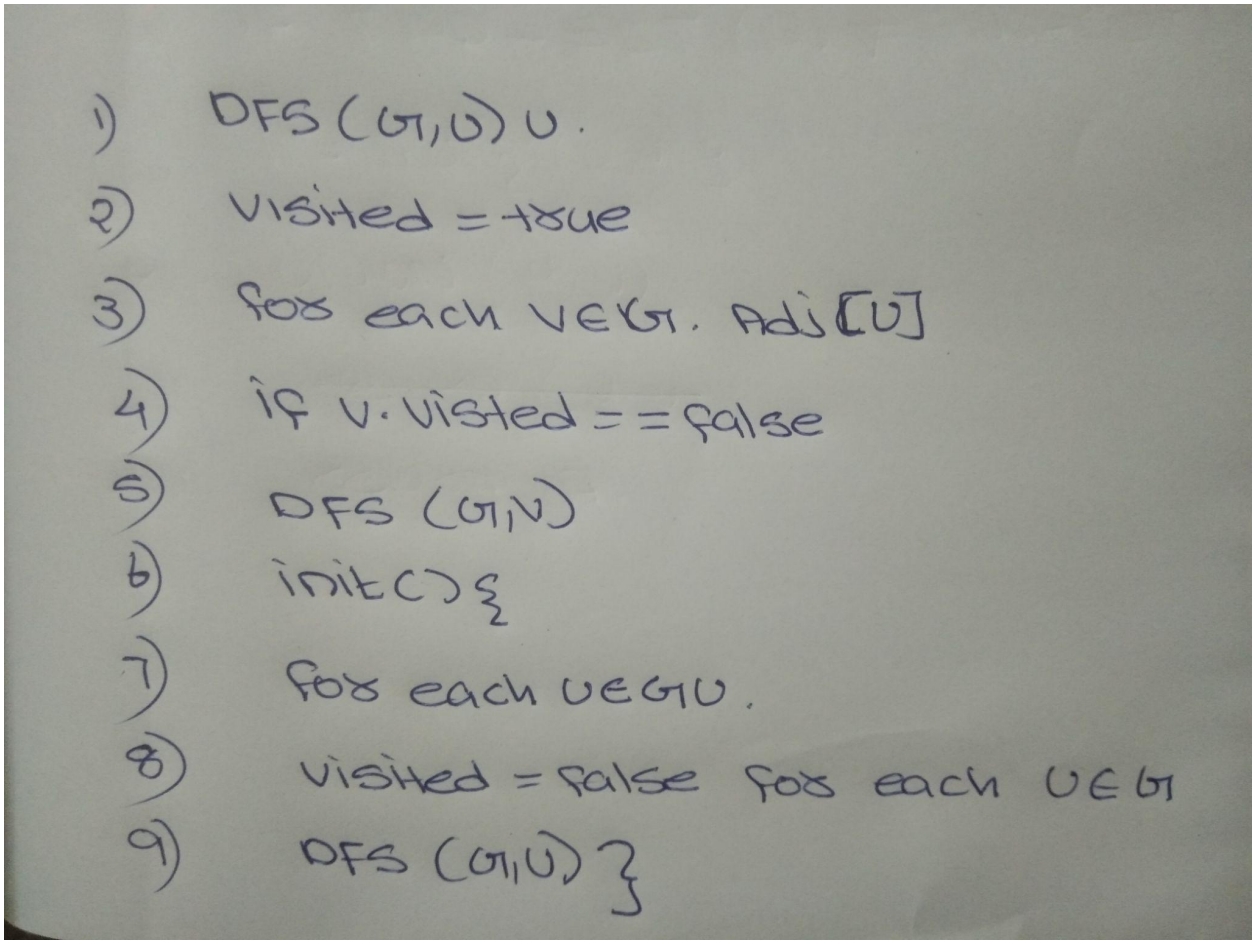
OUTPUT:



(b)DFS:

ALGORITHM

DFS(G, u) u.
visited = true
for each v ∈ G.Adj[u]
if v.visited == false
DFS(G,v)
init()
{
For each u ∈ G u.
visited = false For each u ∈ G
DFS(G, u) }

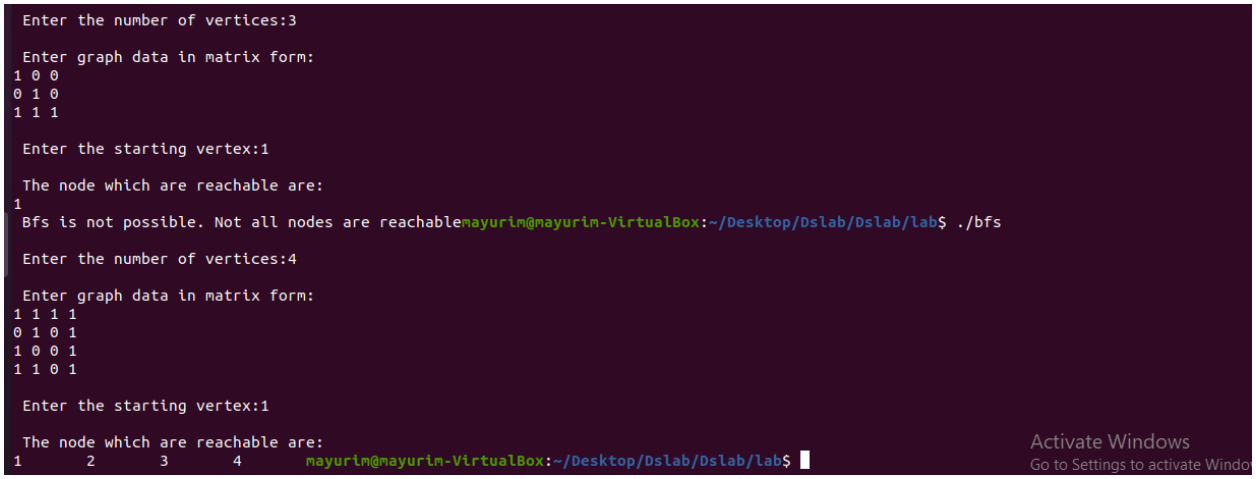


PROGRAM CODE:

```
#include<stdio.h>
int a[20][20],reach[20],n;
int dfs(int v)
{
   int i;
   reach[v]=1;
   for (i=1;i<=n;i++)
     if(a[v][i] && !reach[i])
      {

 printf("\n %d->%d",v,i);
       dfs(i);
   }
}
```

```
int main()
{
    int i,j,count=0;
    printf("\n Enter number of vertices:");
    scanf("%d",&n);
    for (i=1;i<=n;i++)
     {
        reach[i]=0;
        for (j=1;j<=n;j++)
           a[i][j]=0;
    }
    printf("\n Enter the adjacency matrix:\n");
    for (i=1;i<=n;i++)
      for (j=1;j<=n;j++)
       scanf("%d",&a[i][j]);
    dfs(1);
    printf("\n");
    for (i=1;i<=n;i++)
     {
        if(reach[i])
           count++;
    }
    if(count==n)
      printf("\n Graph is connected"); else
      printf("\n Graph is not connected");
    return 0;
    }
```

OUTPUT:


Enter the number of vertices:3
Enter graph data in matrix form:
1 0 0
0 1 0
1 1 1
Enter the starting vertex:1
The node which are reachable are:
1
Bfs is not possible. Not all nodes are reachablemayurim@mayurim-VirtualBox:~/Desktop/Dslab/Dslab/lab$ ./bfs
Enter the number of vertices:4
Enter graph data in matrix form:
1 1 1 1
0 1 0 1
1 0 0 1
1 1 0 1
Enter the starting vertex:1
The node which are reachable are:
1 2 3 4 mayurim@mayurim-VirtualBox:~/Desktop/Dslab/Dslab/lab$
Activate Windows
Go to Settings to activate Windo