**LAB 13**

**Question 01:**

import java.util.\*;

public class Main{

static class line{

int s,loc,w;

public line(int s,int loc,int w){

this.s=s;

this.loc=loc;

this.w=w;

}

}

static class u\_graph{

int n,a[][];

public u\_graph(int n){

this.n=n;

a=new int[n][n];

}

public void insert\_line(int s,int loc){

a[s][loc]=1;

a[loc][s]=1;

}

public void display\_M(){

for(int i=1;i<n;i++){

for (int j=1;j<n;j++){

System.out.print(a[i][j]+" "); }

System.out.println(); }

}

}

public static void generate\_graph(ArrayList<line>[] g){

for (int i=1;i<g.length;i++){

g[i]=new ArrayList<>(); }

g[1].add(new line(1,4,1));

g[2].add(new line(2,4,1));

g[2].add(new line(2,5,1));

g[3].add(new line(3,5,1));

g[4].add(new line(4,1,1));

g[4].add(new line(4,2,1));

g[4].add(new line(4,5,1));

g[5].add(new line(5,2,1));

g[5].add(new line(5,4,1));

g[5].add(new line(5,3,1));

}

public static void display\_L(ArrayList<line>[] g){

for(int i=1;i<g.length;i++){

System.out.print("Node = "+i+",");

for(line e : g[i]){

System.out.print(e.loc+","); }

System.out.println();

}

}

public static void DFS(ArrayList<line>[] g,int c, boolean[] sur){

System.out.print(c+",");

sur[c]=true;

for(int i=0;i<g[c].size();i++){

line e=g[c].get(i);

if(!sur[e.loc]){

DFS(g,e.loc,sur); } }

}

public static void BFS(ArrayList<line>[] g){

Queue<Integer> x=new LinkedList<>();

boolean[] sur=new boolean[g.length];

x.add(1);

while(!x.isEmpty()){

int c=x.remove();

if(!sur[c]){

System.out.print(c+",");

sur[c]=true;

for(int i=0;i<g[c].size();i++){

line e=g[c].get(i);

x.add(e.loc); } }

}

}

public static void main(String[] args){

int N=6;

u\_graph u=new u\_graph(N);

u.insert\_line(1,4);

u.insert\_line(2,4);

u.insert\_line(2,5);

u.insert\_line(4,5);

u.insert\_line(3,5);

ArrayList<line>[] g=new ArrayList[N];

generate\_graph(g);

System.out.println("The adjacency list is : ");

display\_L(g);

System.out.println();

System.out.println("The adjacency matrix is : ");

u.display\_M();

System.out.println();

System.out.println("The bfs of the graph is : ");

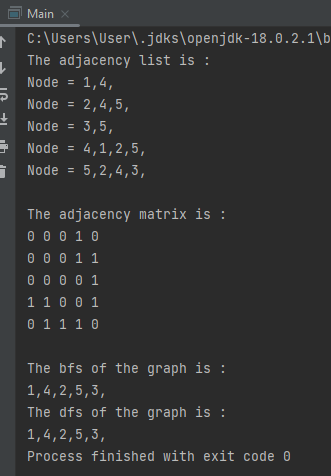
BFS(g);

System.out.println();

System.out.println("The dfs of the graph is : ");

DFS(g,1,new boolean[N]);

} }



**Question 02:**

**By Prims Algorithm :**

class prims\_algo{

static final int N=6;

public int low(int loc[],boolean r[]){

int m=Integer.MAX\_VALUE,mi\_in=-1;

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

if(r[i]==false&&loc[i]<m){

m=loc[i];

mi\_in=i; } }

return mi\_in;

}

public void primsalgo(int g[][],int s){

int loc[]=new int[N];

int p[]=new int[N];

boolean r[]=new boolean[N];

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

loc[i]=Integer.MAX\_VALUE;

r[i]=false; }

loc[s]=0;

p[s]=-1;

for(int c=0;c<N-1;c++){

int x=low(loc,r);

r[x]=true;

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

if(g[x][i]!=0&&r[i]==false&&g[x][i]<loc[i]){

p[i]=x;

loc[i]=g[x][i]; } } }

display(p,g);

}

public void display(int p[],int g[][]){

int z=0;

for(int i=1;i<N;i++){

z=z+g[i][p[i]]; }

System.out.println("MST = "+z);

}

}

public class Main{

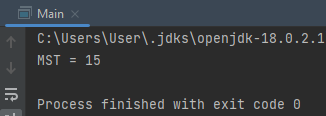
public static void main(String[] args){

int g[][]=new int[][] { {0,6,7,0,2,3},{6,0,5,0,0,0},{7,5,0,0,5,0},{0,0,0,0,4,1},{2,0,5,4,0,2},{3,0,0,1,2,0}};

prims\_algo p= new prims\_algo();

p.primsalgo(g,0);

} }



**By Kruskal Algorithm :**

public class Main{

int N=5;

int k[]=new int[N];

static int d= Integer.MAX\_VALUE;

public int searcing\_i(int p){

while (k[p]!=p)

p=k[p];

return p;

}

public void join(int p,int q){

int x=searcing\_i(p);

int y=searcing\_i(q);

k[x]=y;

}

public void kruskal\_algo(int g[][]){

int mi\_d=0;

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

k[i]=i;

int in=0;

while(in<N-1){

int m=d,x=-1,y=-1;

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

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

if(searcing\_i(i)!=searcing\_i(j)&&g[i][j]<m){

m=g[i][j];

x=i;

y=j; } } }

join(x,y);

System.out.println("At point "+in+" the minimum distance is = "+m);

in++;

mi\_d=mi\_d+m; }

System.out.println("MST = "+mi\_d);

}

public static void main(String[] args){

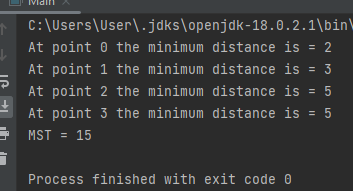
int g[][]={

{d,6,7,d,2,3},{6,d,5,8,5,d},{7,5,d,d,7,d},{d,d,d,d,3,1},{2,d,d,3,d,2},{3,d,d,1,2,d} };

Main f=new Main();

f.kruskal\_algo(g);

} }



**Question 03:**

import java.util.\*;

public class Main{

private static class line{

int s,loc,w;

public line(int s,int loc,int w){

this.s=s;

this.loc=loc;

this.w=w; }

}

public static void dijkstra(ArrayList<line>[] g,int s){

int x[]=new int[g.length];

for(int i=0;i<g.length;i++){

if(i!=s){

x[i]=Integer.MAX\_VALUE; } }

boolean r[]=new boolean[g.length];

PriorityQueue<couple> z=new PriorityQueue<>();

z.add(new couple(0,0));

while(!z.isEmpty()){

couple c=z.remove();

if(!r[c.f]){

r[c.f]=true;

for(int i=0;i<g[c.f].size();i++){

line t=g[c.f].get(i);

int u=t.s;

int v=t.loc;

int wt=t.w;

if((x[u]+wt)<x[v]){

x[v]=x[u]+wt;

z.add(new couple(v,x[v])); } } }

}

for(int i=0;i<x.length;i++){

System.out.println("Distance "+(char)(i + 65)+" is "+x[i]+","); }

}

private static class couple implements Comparable<couple>{

int f,p;

public couple(int f,int p){

this.f=f;

this.p=p;

}

public int compareTo(couple k) {

return this.p-k.p;

}

}

public static void main(String[] args) {

ArrayList<line>[] g=new ArrayList[6];

for(int i=0;i<g.length; i++)g[i]=new ArrayList<>();

g[0].add(new line(0, 1, 4));

g[0].add(new line(0, 3, 8));

g[1].add(new line(1, 0, 4));

g[1].add(new line(1, 2, 9));

g[2].add(new line(2, 1, 9));

g[2].add(new line(2, 3, 6));

g[2].add(new line(2, 4, 14));

g[3].add(new line(3, 0, 8));

g[3].add(new line(3, 2, 6));

g[3].add(new line(3, 5, 4));

g[4].add(new line(4, 2, 14));

g[4].add(new line(4, 5, 3));

g[5].add(new line(5, 3, 4));

g[5].add(new line(5, 4, 3));

dijkstra(g,0);

} }

