**Week 7: Write a java program that implements Prim’s algorithm to generate minimum cost spanning tree**

import java.io.\*;

class prim

{

public static void main(String args[])throws IOException

{

int G[][];

int i,j,n;

DataInputStream in=new DataInputStream(System.in);

System.out.println("Enter number of nodes in graph");

n=Integer.parseInt(in.readLine());

G=new int[n+1][n+1];

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

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

{

if((i!=j)&&(i<j))

{

System.out.print(i+" and "+j+": ");

G[j][i]=G[i][j] = Integer.parseInt(in.readLine());

}

if(G[i][j]==0)

G[i][j]=9999;

}

prim ob=new prim();

ob.primfun(G,n);

}

void primfun(int g[][],int n)

{

int i,j=0,k=0,l=0;

int min=9999;

int near[]=new int[n+1];

int mincost=0,nearver;

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

{

near[i]=0;

}

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

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

if(g[i][j]<min)

{

min=g[i][j];

k=i;l=j;

}

System.out.println("the min cost edge is"+k+"and"+l+"is with cost"+min);

mincost=mincost+g[k][l];

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

if(g[i][l]<g[i][k])

near[i]=l;

else

near[i]=k;

near[k]=near[l]=0;

for(i=2;i<=n-1;i++)

{

int min1=9999;

for(nearver=1;nearver<=n;nearver++)

{

if(g[nearver][near[nearver]]<min1&&near[nearver]!=0)

{

min1=g[nearver][near[nearver]];

j=nearver;

}

}

System.out.println("the min cost edge is"+j+" "+near[j]+"with cost"+min1);

mincost=mincost+g[j][near[j]];

near[j]=0;

for(k=1;k<=n;k++)

if(near[k]!=0&&g[k][near[k]]>g[k][j])

near[k]=j;

}

System.out.println("the total minimum cost spanning tree is"+mincost);

}

}