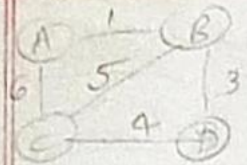


A good day

	A	B	C	D
A	0	1		~~5~~
B		0		~~1~~
C			0	
D				0

→ instead ∞

→ cost matrix

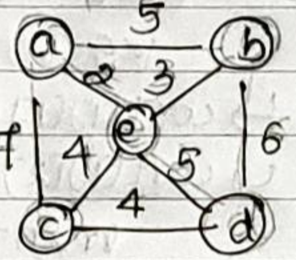
P8) Find the minimum cost spanning tree of a given undirected graph using Kruskal's algorithm Use union-Find algorithm in your program.

```
import Java.util.*;
public class kruskal {
static int min_cost=0;
  public static void main (String[] args) {
   int cost[][]= new int[10][10];
   int i, j, mincost =0;
   Scanner in= new Scanner (System.in);
   system.out.println("--Kruskal Algorithm--");
   System.out.println("Enter the number of nodes:");
   int n=in.nextInt();
   System.out.println("Enter the cost matrix:");
   for(i=1; i<=n; i++)
   {
     for(j=1; j<=n; j++)
     {
        cost[i][j]=in.nextInt();
     }
   }
   System.out.println(" cost matrix:");
   for(i=1; i<=n; i++) {
        for(j=1; j<=n; j++){
        } System.out.print( cost[i][j]+"\t");
System.out.println();
}
mincost = kruskals(n, cost);
System.out.println("The minimum
   spanning tree cost : "+mincost);
System.out.println("* * *");
}}

static int kruskals(int n, int cost[][])
{
   int ne=1, a=0, u=0, b=0, v=0, min;
   int parent[]=new int[10];
   while(ne<n){
      min=999;
   for (int i=1;i<=n;i++){
      for (j=1;j<=n;j++){
         if((cost[i][j] <min) && (cost[i][j]!=0)){
         min=cost[i][j];
         a=u=i;
         b=v=j;
 }}}
 while(parent[u]>0)
  u=parent[u];  //find(u)
 while(parent[v]>0)
 v=parent[v];  //find(v)
```

(1,5) 2 (4,5) 5
(2,3) 3
(3,5) 4

```
if (u!=v){
  System.out.println((ne++)+"> minimum edge is:("
  +a+","+b+") and its cost is:"+min);
min_cost+=min;
parent[v]=u; //union kind of
              → for undirected graph
}
cost[a][b]=cost[b][a]=999; //already visited so skip [illegible]
}
return min_cost;
}
}
```



	a	b	c	d	e
a	0	5	7	999	2
b	5	0	999	6	3
c	7	999	0	4	4
d	999	6	4	0	5
e	2	3	4	5	0

Output:

```
-- Kruskals Algorithm --
Enter the number of nodes
5
Enter the cost matrix
0    5    7    999  2
5    0    999  6    3
7    999  0    4    4
999  6    4    0    5
2    3    4    5    0
The cost matrix is
0    5    7    999  2
5    0    999  6    3
```

```
7     999   0   4   4
999   6     4   0   5
2     3     4   4   0
```

1> minimum edge is : (1,5) & its cost is : 2
2> minimum edge is : (2,5) & its cost is : 3
3> minimum edge is : (3,4) & its cost is : 4
4> minimum edge is : (3,5) & its cost is : 4
The minimum spanning tree cost is = 13

Tracing

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

ne=1 n=5

1<5 min=999

↳ cost[1][1]<999 & cost[1][1]!=0 → fail → j++

↳ cost[1][2]<999 & cost[1][2]!=0 → true

min=cost[1][2] =5   a=u=1 & b=v=2

↳ cost[1][3] ≮ 5

↳ cost[1][4] ≮ 5

↳ cost[1][5] =2<5 && 2!=0 ⇒ true.

min=2   a=u=1 & b=v=5

↳ cost[2][1] :.. upto cost [5][5] //note that in
// matrix none of cost is less than 2 thus
// values of min & a,u,b,v remains unchanged.

↳ parent[u]>0 ⇒ parent[1] ≯ 0
parent[v]>0 ⇒ parent[5] ≯ 0

↳ $u \neq v \Rightarrow (1 \neq 5) \rightarrow$ true

pf: **1 > minedge : (a,b) // (1,5) ∴ cost = min (2)**

* min-cost = 0+2 = 2   ne++ ; ne = 2

parent [v] = parent [5] = u = 1

cost [a][b] = cost [b][a] = 999

i.e cost [1][5] = cost[5][1] = 999

0	5	7	999	$\boxed{999}$
5	0	999	6	3
7	999	0	4	4
999	6	4	0	5
$\boxed{999}$	3	4	5	0

parent

0	0	0	0	1
1	2	3	4	5

ne = 2

2<5   min = 999

↳ cost[1] [1] != 0 → fail

↳ cost[1] [2] < 999 & cost[1] [2] != 0 → true

min = 5   a = u = 1   b = v = 2

↳ cost[1] [3] - - - cost [2] [4] condition fails

↳ cost[2] [5] ⇒ 3 < 5 & 3 != 0

min = 3   a = u = 2   b = v = 5

↳ cost [3] [1] upto cost[5][5] condition fails.

↳ parent[u] > 0   parent[2] ≯ 0

↳ parent[v] > 0   parent[5] > 0 → true

v = parent[v] = 1 $\boxed{v = 1}$

↳ u ≠ v ⇒ 2 ≠ 1 → True

pf: **(ne) 2 > minedge (2,5)   cost : 3   ne = 3**

parent [v] = u ⇒ parent[1] = 2

* $\boxed{\text{min cost} = 5}$ parent

2	0	0	0	1
1	2	3	4	5

cost [2][5] = cost[5][2] = 999

ne=3   3<5   min=999

↳ cost[1][1] !=0 → fail

↳ cost[1][2]<999 & !=0

min=5

a=u=1   b=v=2

↳ cost[1][3] upto cost[3][3] → loop fails

↳ cost[3][4] ⇒ 4<5 & !=0

cost=4   a=u=3   b=v=4

↳ cost[3][5] upto cost[5][5] → loop fails

parent[3]≯0   parent[4]≯0

↳ u!=v → 3!=4 true

$$\begin{array}{ccccc} 0 & 5 & 7 & 999 & 999 \\ 5 & 0 & 999 & 6 & 999 \\ 7 & 999 & 0 & 4 & 4 \\ 999 & 6 & 4 & 0 & 5 \\ 999 & 999 & 4 & 5 & 0 \end{array}$$

Pb: 3 → min edge (3,4) & its cost = 4 $\boxed{ne=4}$

parent[4]=3

min_cost=9

	1	2	3	4	5
parent	2	0	0	3	1

cost[3][4]=cost[4][3]=999

$$\begin{array}{ccccc} 0 & 5 & 7 & 999 & 999 \\ 5 & 0 & 999 & 6 & 999 \\ 7 & 999 & 0 & 999 & \boxed{4} \\ 999 & 6 & 999 & 0 & 5 \\ 999 & 999 & 4 & 5 & 0 \end{array}$$

ne=4   4<5   min=999

↳ cost[1][1]!=0 → fail

↳ cost[1][2] ⇒ 5<999 & !=0

~~co~~ min=5   a=u=1   b=v=2

↳ cost[1][3] upto cost[3][4] → loop fails

↳ cost[3][5] ⇒ 4<5 & !=0

min=4   a=u=3   b=v=5

↳ parent[u]>0   parent[3]≯0

↳ parent[v]>0   parent[4]>0 ⇒ v=parent[5]=1

v=1

↳ 3 != 1

P6: 4 > min edge (3, 5) & it's cost : 4   ne=5

parent[5] = 3 ✓ ✓ min_cost = 13

~~cost[3][5] = cost[5][3]~~ = 999

ne = 5

5 ≮ 5

return 13