Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Experiment 4

(Greedy Algorithm)

NAME : DIVYESH KHUNT
SAPID:60009210116

Aim: Implementation of Prism's & Kruskal's method.

Prim's algorithm:

Theory:

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph?

Algorithm:

Step 1:

- Randomly choose any vertex.
- The vertex connecting to the edge having least weight is usually selected.

Step 2:

- Find all the edges that connect the tree to new vertices.
- Find the least weight edge among those edges and include it in the existing tree.
- If including that edge creates a cycle, then reject that edge and look for the next least weight edge.

Step 3:

- Keep repeating step-02 until all the vertices are included and Minimum Spanning Tree (MST) is obtained.

Example:

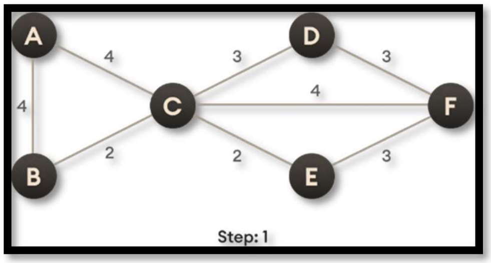


Figure 1. Start with a weighted graph

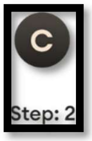


Figure 2.Choose a vertex

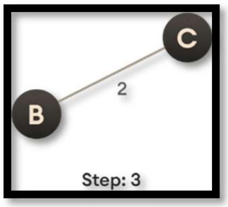


Figure 3.Choose the shortest edge from this vertex and add it

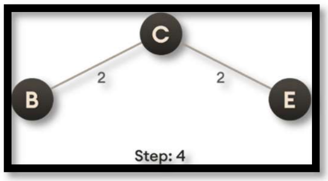


Figure 4.Choose the nearest vertex not yet in the solution

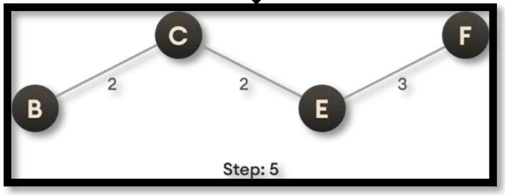


Figure 5.Choose the nearest edge not yet in the solution, if there are multiple choices, choose one at random

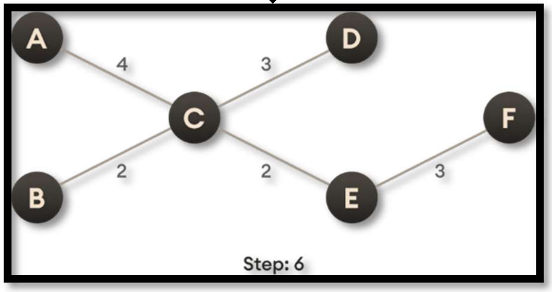
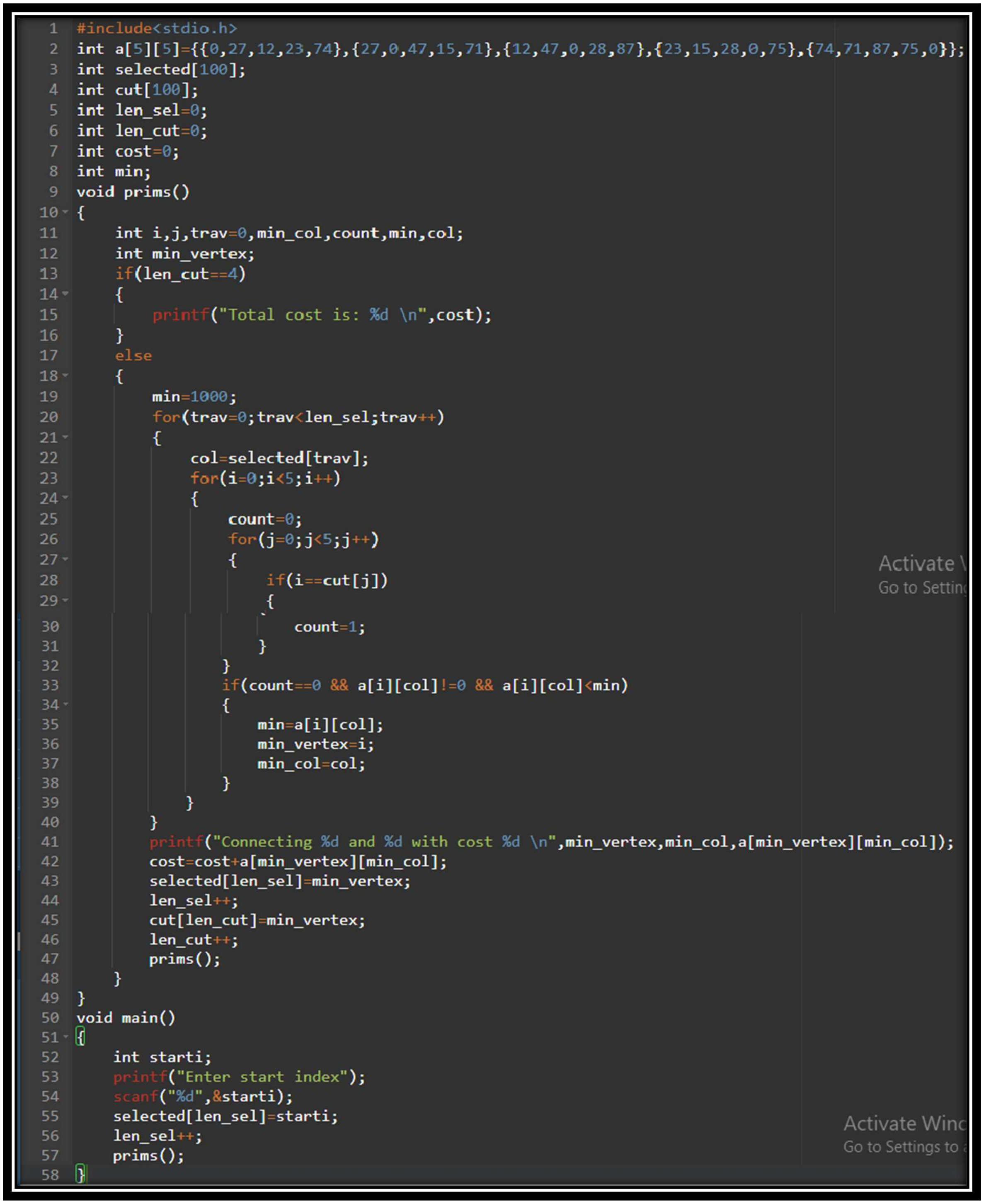


Figure 6.Repeat until you have a spanning tree

Complexity:

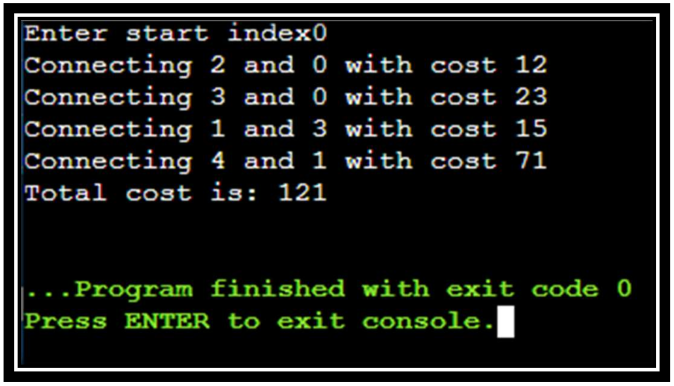
The time complexity of Prim's algorithm is O(E log V).

Prim's code –



```c
#include<stdio.h>
int a[5][5]={{0,27,12,23,74},{27,0,47,15,71},{12,47,0,28,87},{23,15,28,0,75},{74,71,87,75,0}};
int selected[100];
int cut[100];
int len_sel=0;
int len_cut=0;
int cost=0;
int min;
void prims()
{
    int i,j,trav=0,min_col,count,min,col;
    int min_vertex;
    if(len_cut==4)
    {
        printf("Total cost is: %d \n",cost);
    }
    else
    {
        min=1000;
        for(trav=0;trav<len_sel;trav++)
        {
            col=selected[trav];
            for(i=0;i<5;i++)
            {
                count=0;
                for(j=0;j<5;j++)
                {
                    if(i==cut[j])
                    {
                        count=1;
                    }
                }
                if(count==0 && a[i][col]!=0 && a[i][col]<min)
                {
                    min=a[i][col];
                    min_vertex=i;
                    min_col=col;
                }
            }
        }
        printf("Connecting %d and %d with cost %d \n",min_vertex,min_col,a[min_vertex][min_col]);
        cost=cost+a[min_vertex][min_col];
        selected[len_sel]=min_vertex;
        len_sel++;
        cut[len_cut]=min_vertex;
        len_cut++;
        prims();
    }
}
void main()
{
    int starti;
    printf("Enter start index");
    scanf("%d",&starti);
    selected[len_sel]=starti;
    len_sel++;
    prims();
}
```

Output –



```
Enter start index0
Connecting 2 and 0 with cost 12
Connecting 3 and 0 with cost 23
Connecting 1 and 3 with cost 15
Connecting 4 and 1 with cost 71
Total cost is: 121


...Program finished with exit code 0
Press ENTER to exit console.
```

Kruskal's algorithm:

Theory:

Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph?

Algorithm:

Step 1:

- Sort all the edges from low weight to high weight.

Step 2:

- Take the edge with the lowest weight and use it to connect the vertices of graph.
- If adding an edge creates a cycle, then reject that edge and go for the next least weight edge.

Step 3:

- Keep adding edges until all the vertices are connected and a Minimum Spanning Tree (MST) is obtained.

Example:

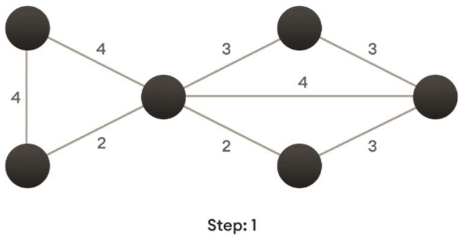


Figure 7. Start with a weighted graph

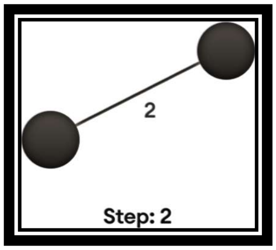


Figure 8. Choose the edge with the least weight, if there are more than 1, choose anyone

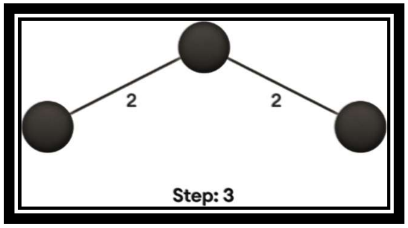


Figure 9. Choose the next shortest edge and add it

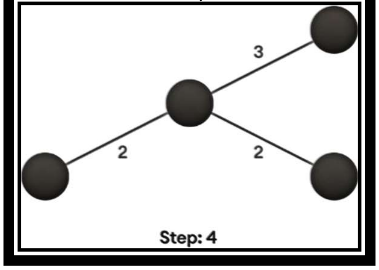


Figure 10. Choose the next shortest edge that doesn't create a cycle and add it

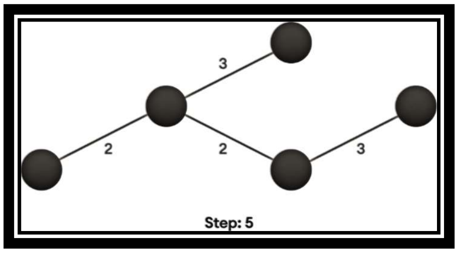


Figure 11. Choose the next shortest edge that doesn't create a cycle and add it

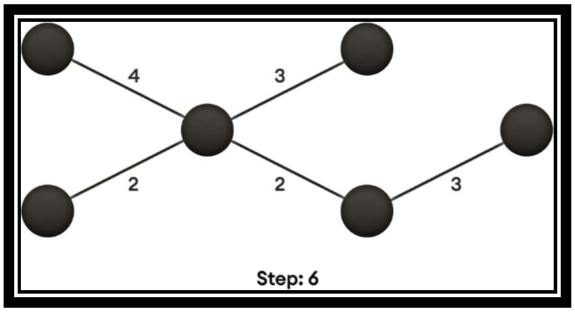
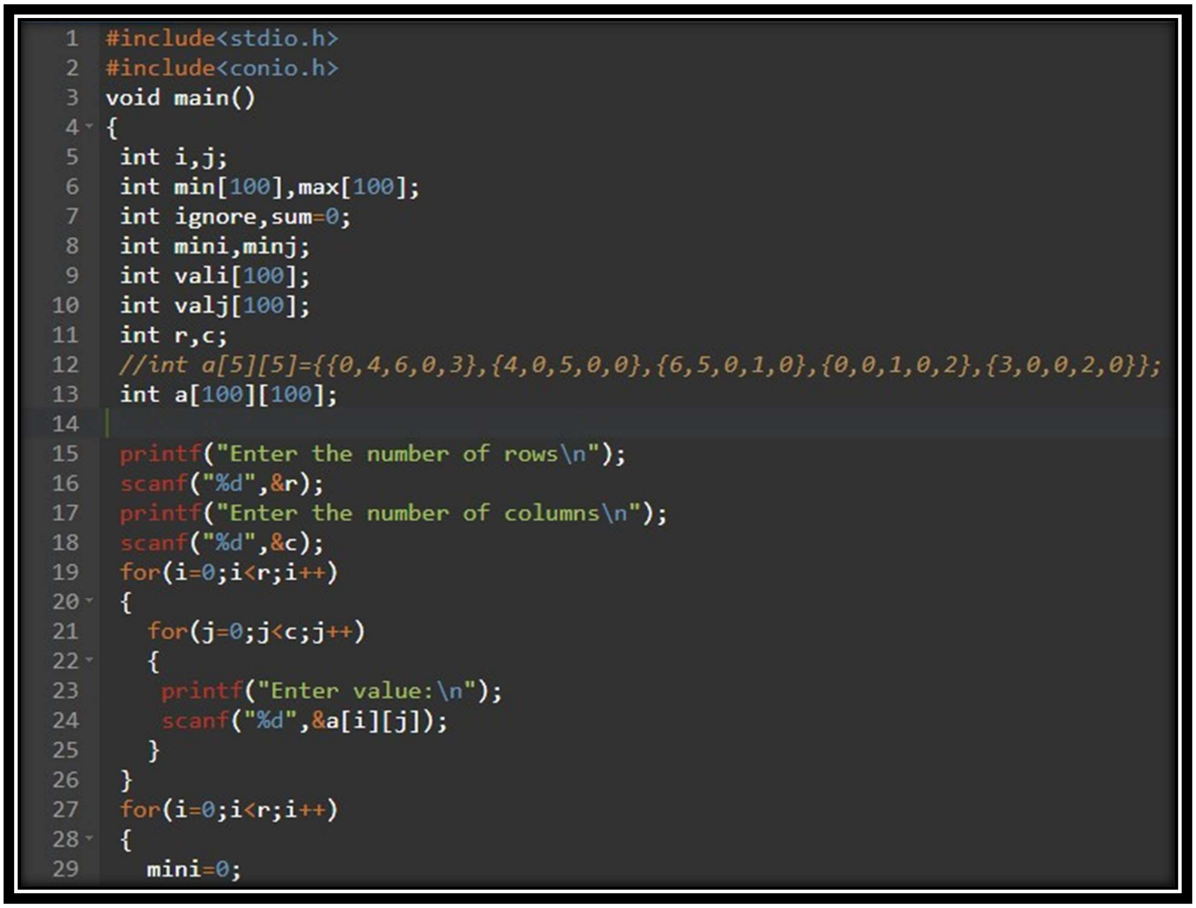


Figure 12.Repeat until you have a spanning tree

Complexity:

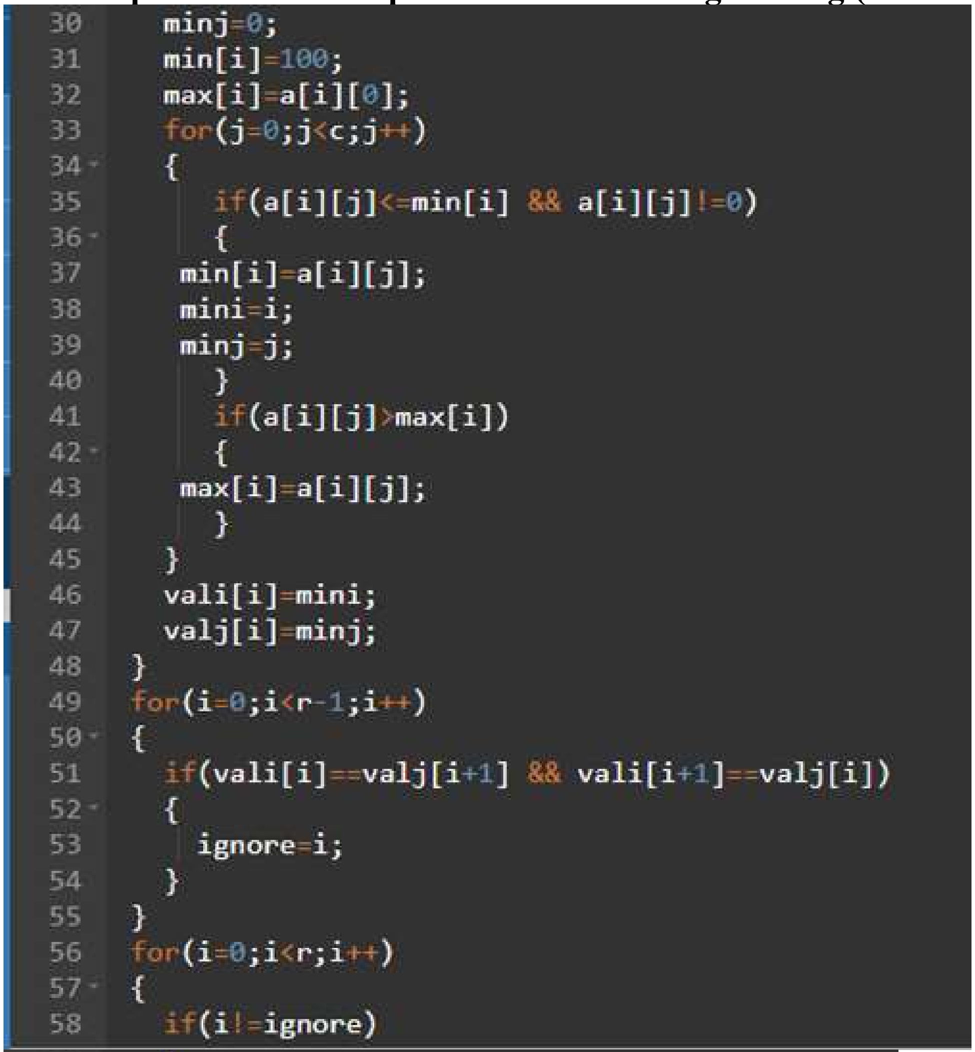
The time complexity Of Kruskal's Algorithm is: O(E log E).

Kruskal’s code –

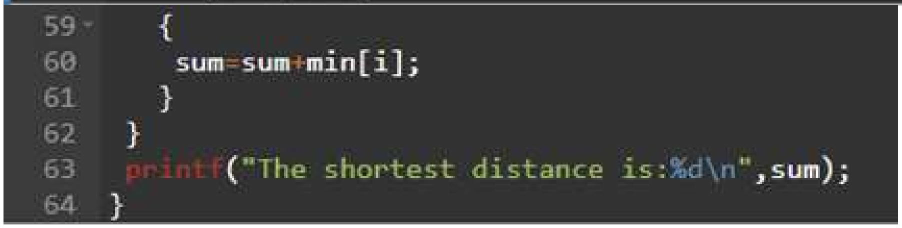


```c
#include<stdio.h>
#include<conio.h>
void main()
{
 int i,j;
 int min[100],max[100];
 int ignore,sum=0;
 int mini,minj;
 int vali[100];
 int valj[100];
 int r,c;
 //int a[5][5]={{0,4,6,0,3},{4,0,5,0,0},{6,5,0,1,0},{0,0,1,0,2},{3,0,0,2,0}};
 int a[100][100];

 printf("Enter the number of rows\n");
 scanf("%d",&r);
 printf("Enter the number of columns\n");
 scanf("%d",&c);
 for(i=0;i<r;i++)
 {
   for(j=0;j<c;j++)
   {
    printf("Enter value:\n");
    scanf("%d",&a[i][j]);
   }
 }
 for(i=0;i<r;i++)
 {
   mini=0;
```

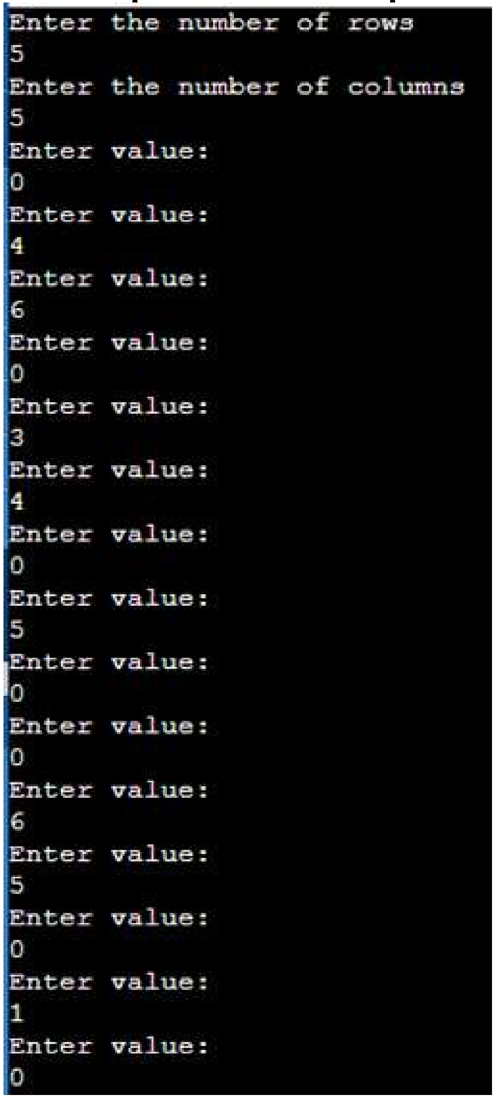


```
    minj=0;
    min[i]=100;
    max[i]=a[i][0];
    for(j=0;j<c;j++)
    {
       if(a[i][j]<=min[i] && a[i][j]!=0)
       {
     min[i]=a[i][j];
     mini=i;
     minj=j;
       }
       if(a[i][j]>max[i])
       {
     max[i]=a[i][j];
       }
    }
    vali[i]=mini;
    valj[i]=minj;
  }
  for(i=0;i<r-1;i++)
  {
    if(vali[i]==valj[i+1] && vali[i+1]==valj[i])
    {
      ignore=i;
    }
  }
  for(i=0;i<r;i++)
  {
    if(i!=ignore)
```

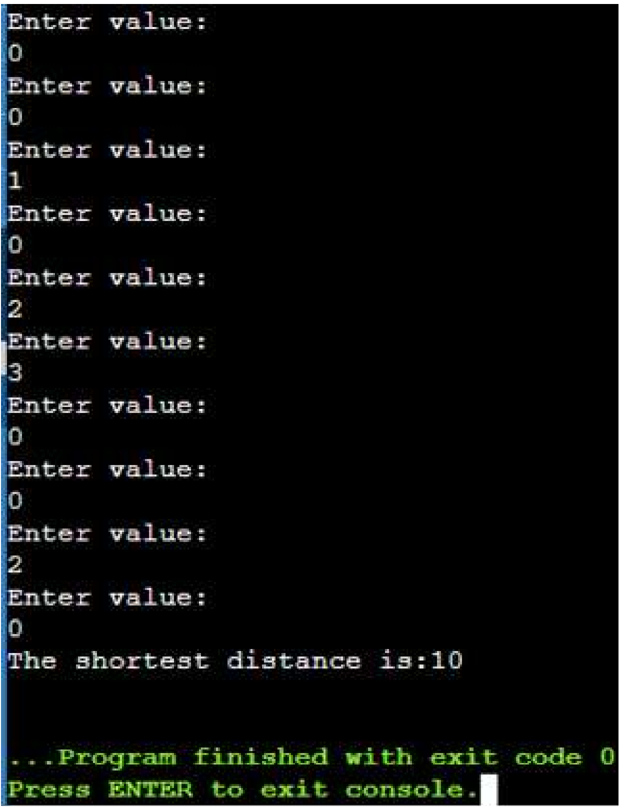


```
    {
     sum=sum+min[i];
    }
  }
  printf("The shortest distance is:%d\n",sum);
}
```

Output –



```
Enter the number of rows
5
Enter the number of columns
5
Enter value:
0
Enter value:
4
Enter value:
6
Enter value:
0
Enter value:
3
Enter value:
4
Enter value:
0
Enter value:
5
Enter value:
0
Enter value:
0
Enter value:
6
Enter value:
5
Enter value:
0
Enter value:
1
Enter value:
0
```



```
Enter value:
0
Enter value:
0
Enter value:
1
Enter value:
0
Enter value:
2
Enter value:
3
Enter value:
0
Enter value:
0
Enter value:
2
Enter value:
0
The shortest distance is:10


...Program finished with exit code 0
Press ENTER to exit console.
```