**Kruskal's Algorithm**

**For**

**minimum spanning tree**

Kruskal's algorithm is a greedy algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. 

It finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

This algorithm is directly based on the MST( minimum spanning tree) property.

**Properties of Kruskal's Algorithm:**

1. It works not only with directed graphs.
2. It works with weighted and not weighted graphs. But, is more interesting, when edges are weighted.
3. It is a type of greedy algorithm that produces optimal solutions of the MST.

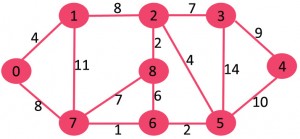
**Steps of Kruskal's Algorithm:**

The main steps of the Kruskal's Algorithm are as follows:

1. Arrange the edges by weight: least weight first and heaviest last.
2. Choose the lightest not examined edge from the diagram. Add this chosen edge to the tree, only if doing so will not make a cycle.
3. Stop the process whenever n - 1 edges have been added to the tree.

**Solve the following by applying Kruskal's Algorithm**

**Example 1**

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-0.jpg)

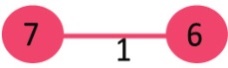
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having (9 – 1) = 8 edges. Now edges are sorted in ascending order by weight.

Procedure for finding Minimum Spanning Tree

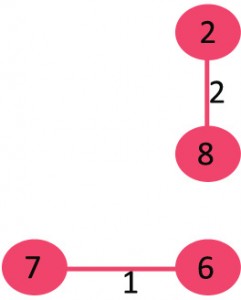
|  |  |  |
| --- | --- | --- |
| **Edge No.** | **Vertex Pair** | **Edge Weight** |
| E1 | (6, 7) | 1 |
| E2 | (2, 8) | 2 |
| E3 | (5, 6) | 2 |
| E4 | (0, 1) | 4 |
| E5 | (2, 5) | 4 |
| E6 | (6, 8) | 6 |
| E7 | (2, 3) | 7 |
| E8 | (7, 8) | 7 |
| E9 | (0, 7) | 8 |
| E10 | (1, 2) | 8 |
| E11 | (3, 4) | 9 |
| E12 | (4, 5) | 10 |
| E13 | (1, 7) | 11 |
| E14 | (3, 4) | 14 |

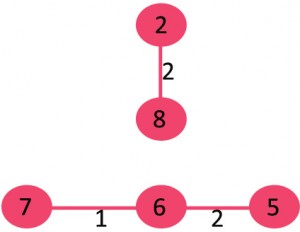
Now pick all edges one by one from sorted list of edges

**1.** *Pick edge 7-6:* No cycle is formed, include it.

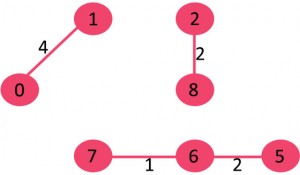
[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-1.jpg)

**2.** *Pick edge 8-2:* No cycle is formed, include it.

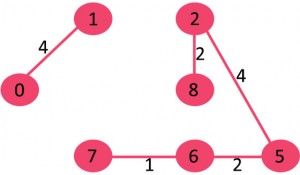
**3.** *Pick edge 6-5:* No cycle is formed, include it.[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-2.jpg)

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-3.jpg)

**4.** *Pick edge 0-1:* No cycle is formed, include it.

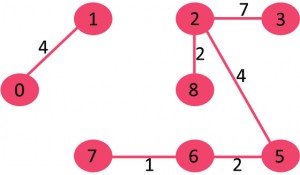
[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-4.jpg)

**5.** *Pick edge 2-5:* No cycle is formed, include it.

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-5.jpg)

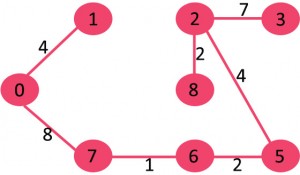
**6.***Pick edge 8-6:*Since including this edge results in cycle, discard it.

**7.** *Pick edge 2-3:* No cycle is formed, include it.

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-6.jpg)

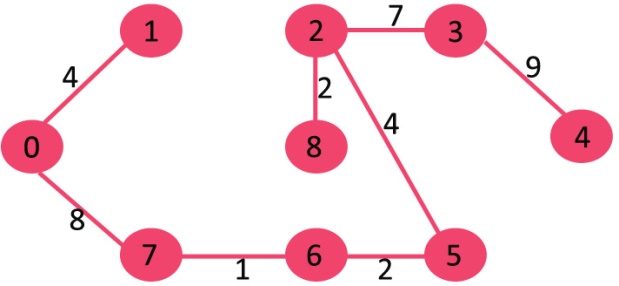
**8.** *Pick edge 7-8:* Since including this edge results in cycle, discard it.

**9.** *Pick edge 0-7:* No cycle is formed, include it.

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Fig-7.jpg)

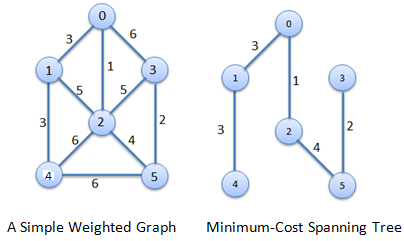
**10.** *Pick edge 1-2:*Since including this edge results in cycle, discard it.

**11.** *Pick edge 3-4:* No cycle is formed, include it.

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/fig8new.jpeg)

Since the number of edges included equals (V – 1), the algorithm stops here.

Example 2



|  |  |
| --- | --- |
|  |  |
|  |  |

Procedure for finding Minimum Spanning Tree

**Step1.** Edges are sorted in ascending order by weight.

|  |  |  |
| --- | --- | --- |
| **Edge No.** | **Vertex Pair** | **Edge Weight** |
| E1 | (0,2) | 1 |
| E2 | (3,5) | 2 |
| E3 | (0,1) | 3 |
| E4 | (1,4) | 3 |
| E5 | (2,5) | 4 |
| E6 | (1,2) | 5 |
| E7 | (2,3) | 5 |
| E8 | (0,3) | 6 |
| E9 | (2,4) | 6 |
| E10 | (4,5) | 6 |

**Step2.** Edges are added in sequence.

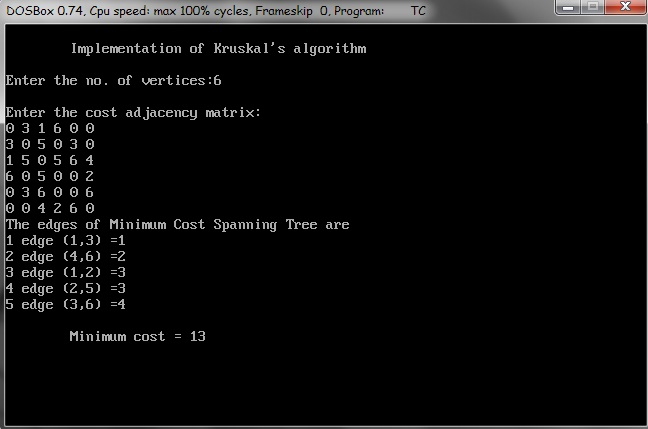
|  |  |
| --- | --- |
| Graph | http://scanftree.com/Data_Structure/kstep.png |
| Add Edge E1 | http://scanftree.com/Data_Structure/kstep1.png |
| Add Edge E2 | http://scanftree.com/Data_Structure/kstep2.png |
| Add Edge E3 | http://scanftree.com/Data_Structure/kstep3.png |
| Add Edge E4 | http://scanftree.com/Data_Structure/kstep4.png |
| Add Edge E5 | http://scanftree.com/Data_Structure/kstep5.png |

**Total Cost** = 1+2+3+3+4 = 13s

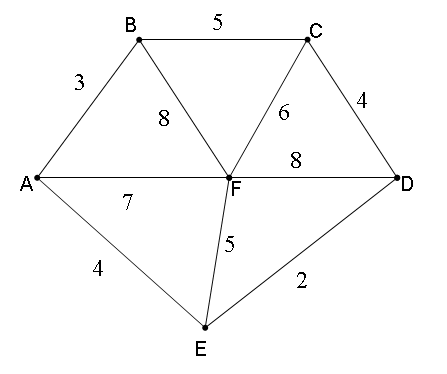
C IMPLEMETATION of Kruskal's Algorithm

1. #include<stdio.h>
2. #include<conio.h>
3. #include<stdlib.h>
4. inti,j,k,a,b,u,v,n,ne=1;
5. intmin,mincost=0,cost[9][9],parent[9];
6. int find(int);
7. intuni(int,int);
8. void main()
9. {
10. clrscr();
11. printf("\n\tImplementation of Kruskal's algorithm\n");
12. printf("\nEnter the no. of vertices:");
13. scanf("%d",&n);
14. printf("\nEnter the cost adjacency matrix:\n");
15. for(i=1;i<=n;i++)
16. {
17. for(j=1;j<=n;j++)
18. {
19. scanf("%d",&cost[i][j]);
20. if(cost[i][j]==0)
21. cost[i][j]=999;
22. }
23. }
24. printf("The edges of Minimum Cost Spanning Tree are\n");
25. while(ne < n)
26. {
27. for(i=1,min=999;i<=n;i++)
28. {
29. for(j=1;j <= n;j++)
30. {
31. if(cost[i][j] < min)
32. {
33. min=cost[i][j];
34. a=u=i;
35. b=v=j;
36. }
37. }
38. }
39. u=find(u);
40. v=find(v);
41. if(uni(u,v))
42. {
43. printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
44. mincost +=min;
45. }
46. cost[a][b]=cost[b][a]=999;
47. }
48. printf("\n\tMinimum cost = %d\n",mincost);
49. getch();
50. }
51. int find(int i)
52. {
53. while(parent[i])
54. i=parent[i];
55. return i;
56. }
57. intuni(inti,int j)
58. {
59. if(i!=j)
60. {
61. parent[j]=i;
62. return 1;
63. }
64. return 0;
65. }

Output



**Example 3**

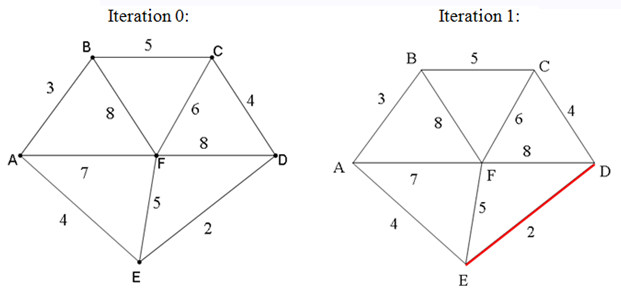


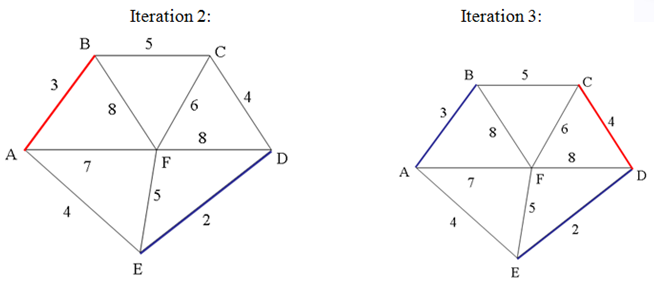
**Solution**

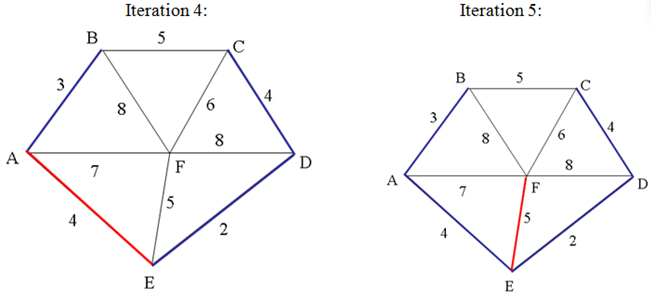
The list of edges in order of their weights or sizes would be as follows:

**Edge    Weight    Edge    Weight  
ED**          2               **EF**         5  
**AB**          3               **CF**         6  
**AE**          4               **AF**         7  
**CD**         4                **BF**         8  
**BC**         5                **CF**         6

The various iterations would be as follows:







All the vertices have been connected now, hence, the last iteration number 5, gives us the optimal solution, and the minimum length would be the sum of all weights given to these edges, as 2 + 3 + 4 + 4 + 5 = 18 is the length of the shortest path by applying Kruskal's Algorithm.  
  
That is, the solution is  
ED  2  
AB  3  
CD  4   
AE  4  
EF  5,   
with Total weight of tree equal to 18

Example 4

