**Assignment No: B-11**

**Title: - To** study and implement the construction of minimum spanning tree.

**Index Terms:** class, objects, MST, Prims, Cycle

**Problem Statement:** You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

**Theory:**

**Spanning Tree:** Any tree, which consists solely of edges in graph G and includes all the vertices in G is called as a spanning tree. Thus for a given connected graph there are multiple spanning trees possible. For a maximal connected graph having n vertices the number of different possible spanning trees is equal to (n!).

**Minimum Spanning Tree:** The minimum cost or minimum weightage spanning tree is called as Minimum Spanning Tree. To obtain a minimum spanning tree for a given connected graph, we can use Prim,s algorithm (vertex by vertex) or Kruskal‟ s algorithm (edge by edge).

**Prim’s Algorithm:** In computer science, Prim's algorithm is a [greedy algorithm that f](http://en.wikipedia.org/wiki/Greedy_algorithm)inds a [minimum spanning tree for a](http://en.wikipedia.org/wiki/Minimum_spanning_tree) [connected weighted undirected graph. This m](http://en.wikipedia.org/wiki/Connected_graph)eans it finds a subset of the [edges that f](http://en.wikipedia.org/wiki/Edge_%28graph_theory%29)orms a [tree that i](http://en.wikipedia.org/wiki/Tree_%28graph_theory%29)ncludes every [vertex, w](http://en.wikipedia.org/wiki/Vertex_%28graph_theory%29)here the total weight of all the [edges i](http://en.wikipedia.org/wiki/Graph_theory)n the

tree is minimized.

All vertices of a connected graph are included in minimum cost spanning tree. Prims algorithm starts from one vertex and grows the rest of tree by adding one vertex at a time by adding associated edge in set T. This algorithm builds a tree by iteratively adding edges until all vertices are visited. The resultant tree is a minimum spanning tree. At each iteration it **selects the vertex** and associated edge having minimum cost or weightage that does not create a cycle. The algorithm is:

**Applications of spanning trees:**

* To find independent set of circuit equations for an electrical network. By adding an edge from set B to spanning tree we get a cycle and then Kirchhoff‟ s second law is used on the resulting cycle to obtain a circuit equation. Thus the total number of independent circuit equation we get

is equal to the number of edges in set B.

* Using the property of spanning trees we can select the spanning tree with (n-1) edges such that total cost is minimum if each edge in a graph represents cost (weightage). For example, a communication network between number of cities.

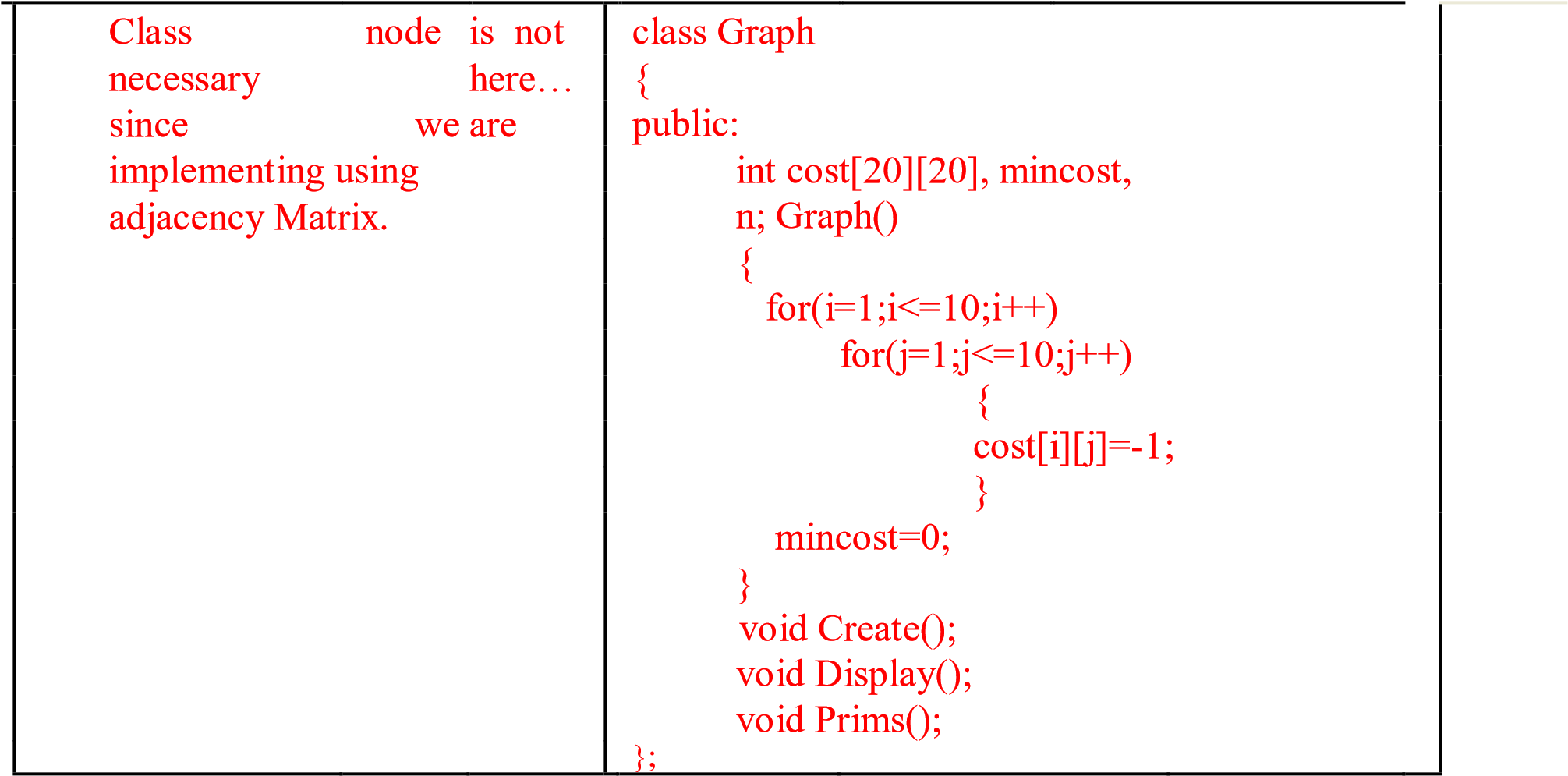
***Example run***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Image** |  | **U** | **Edge(u,v)** | **V \ U** | **Description** |
|  |  |  |  |  | This is our |
|  | {} |  |  | {A,B,C,D, E,F,G} | original weighted graph. The numbers near the edges indicate their weight. |
|  |  |  |  |  | Vertex **D** has been arbitrarily chosen as a starting point. Vertices **A**, |
|  | {D} |  | (D,A) = 5 **V**  (D,B) = 9  (D,E) = 15  (D,F) = 6 | {A,B,C,E, F,G} | **B**, **E** and **F** are connected to **D** through a single edge. **A** is the vertex nearest to **D** and will be chosen as the second vertex along with the |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | The next vertex chosen is the vertex nearest to |
|  | {A,D} | (D,B) = 9  (D,E) = 15  (D,F) = 6 **V** | {B,C,E,F,  G} | *either* **D**or**A**.**B**is 9 away from **D** and 7 away from **A**, **E** is 15, and **F** is 6. **F** |
|  |  | (A,B) = 7 |  | is the smallest distance away, so we highlight the vertex **F** and the arc **DF**. |
|  | {A,D,F} | (D,B) = 9  (D,E) = 15  (A,B) = 7 **V**  (F,E) = 8  (F,G) = 11 | {B,C,E,G  } | The algorithm carries on as above. Vertex **B**, which is 7 away from **A**, is highlighted. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | In this case, we |
|  |  | (B,C) = 8 |  | can choose between **C**, **E**, and |
|  |  | (B,E) = 7 **V** |  | **G**. **C** is 8 away |
|  | {A,B,D,F} | (D,B) = 9 cycle | {C,E,G} | from **B**, **E** is 7 away from **B**, and |
|  |  | (D,E) = 15 |  | **G** is 11 away |
|  |  | (F,E) = 8 |  | from **F**. **E** is |
|  |  | (F,G) = 11 |  | nearest, so we highlight the vertex **E** and the |
|  | {A,B,D,E,F} | (B,C) = 8 (D,B) = 9 cycle (D,E) = 15 cycle (E,C) = 5 **V**  (E,G) = 9 (F,E) = 8 cycle (F,G) = 11 | {C,G} | Here, the only vertices available are **C** and **G**. **C** is 5 away from **E**, and **G** is 9 away from **E**. **C** is chosen, so it is highlighted along with the |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | (B,C) = 8 cycle |  | Vertex **G** is the |
|  |  | (D,B) = 9 |  | only remaining |
|  | {A,B,C,D,E,F  } | cycle (D,E) = 15 cycle | {G} | vertex. It is 11 away from **F**, and |
|  |  | (E,G) = 9 **V** |  | 9 away from **E**. **E** |
|  |  | (F,E) = 8 |  | is nearer, so we |
|  |  | cycle |  | highlight **G** and |
|  |  | (F,G) = 11 |  | the arc **EG**. |
|  |  | (B,C) = 8 |  |  |
|  | {A,B,C,D,E,F, G} | cycle (D,B) = 9 cycle (D,E) = 15 cycle (F,E) = 8 cycle (F,G) = 11 cycle | {} | Now all the vertices have been  selected and the [minimum](http://en.wikipedia.org/wiki/Minimum_spanning_tree)  [spanning tree is](http://en.wikipedia.org/wiki/Minimum_spanning_tree) shown in green. In this case, it has weight 39. |
|  |  |  |  |  |



**Algorithm Create ()**

// **This algorithm is used to read input undirected graph from user.**

1. {
2. Read( no of nodes as n);
3. Repeat()
4. {
5. Write(“Enter starting and ending vertices and its cost”);
6. Read(v1 , v2 and c);
7. cost[v1][v2]=cost[v2][v1]=c;
8. Write(“Do You Want To Enter More Edges ");
9. }until(false);
10. }

**Algorithm Display ()**

// **This algorithm is used to print undirected graph given by user.**

1. {
2. for( i=1 to n) do
3. for(j=1 to n ) do
4. Write(cost[i][j]);
5. }

**Algorithm Prims()**

// **This algorithm is used to find MST and its cost using Prims Logic**

1. {
2. for(i=1 to n) do
3. visit[i]=0;
4. Read (“Enter Staring Vertex in s”);
5. Visit[s]=1;
6. for(k=1 to n-1) do
7. {
8. min=999;
9. for(i=1 to n) do {
10. for(j=1 to n) do
11. {
12. if(visit[i]==1 && visit[j]==0) // look for unvisited vertex from visited
13. {
14. if(cost[i][j]!=-1 && min>cost[i][j]) find near with min cost
15. {
16. min=cost[i][j];
17. row=i;
18. col=j;
19. }
20. }
21. }
22. }
23. Write(Selected Edge in MST is row and col);
24. mincost=mincost+min;
25. visit[col]=1;
26. cost[row][col]=-1;
27. cost[col][row]=-1;
28. }
29. Write(Total Min Cost as mincost);
30. }

***Frequently Asked Questions:***

**1) What is a minimum spanning tree?**

**2) What is weighted graph?**

1. **What are the Prim’s algorithms?**
2. **What are the applications of the minimal spanning tree?**
3. **How does Prim's algorithm work?**
4. [**What is the Complexity of prim’s algorithm?**](http://wzus1.ask.com/r?t=p&d=us&s=usseo&c=qasp&app=a16&dqi=&askid=&l=dir&o=102140&sv=0a5c407a&ip=0e8c2e45&id=DC85157B5EBED88F8F99A4F4B0F43795&q=Prims-Algorithm&p=1&qs=121&ac=590&g=260edotZarfGhv&en=af&io=3&ep=&eo=&b=qai&bc=&br=&tp=bot&ec=7&pt=What%20is%20the%20Complexity%20of%20kruskal%20and%20prim%27s%20algorithm%3F&ex=&url=&u=http://wiki.answers.com/Q/What_is_the_Complexity_of_kruskal_and_prim%27s_algorithm)
5. **What is the application of prim’s algorithm?**
6. **State the advantages of the prim’s algorithm?**

***Flowcharts:***

***Conclusions:***