* Assignment No. 6 *

• Title :- Program for constructing the dijkstra algorithm

• objective :-

i) To Understand Concept of graph.

ii) To Understand Concept of Undireated weighted and graph and dijktras algorithm.

* problem statement :-

You have a business with serval offices, you want to lease phase lines to connect them up with each other and the phase Company charges different amount of Company pair of cities. you want a set of line that connect all your offices. with a minimum totel cost solve the problem by Suggesting appropriate data structure.

* Outcomes :-

Input : Number of vertices of offices and provide edge between vertices offices as a telepnone leases line.

Output : The path which have minimum distance.

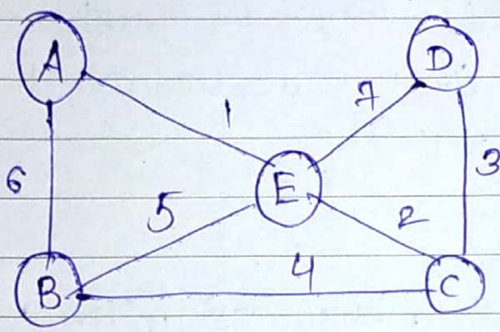
Required :

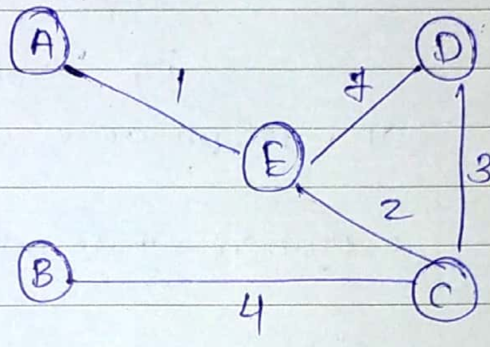
* Hardware Requirement :

8 G.B ram, 500GB/1TB Dual core processor.

◦ Theory :-

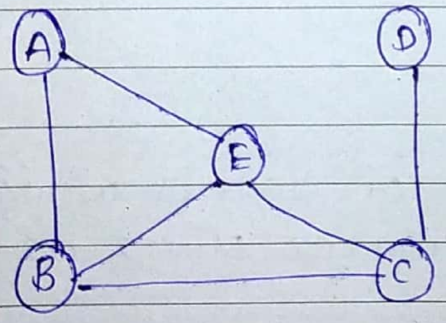
Spanning Tree: A spanning tree of A connected graph G is there. T contaning the vertices of.

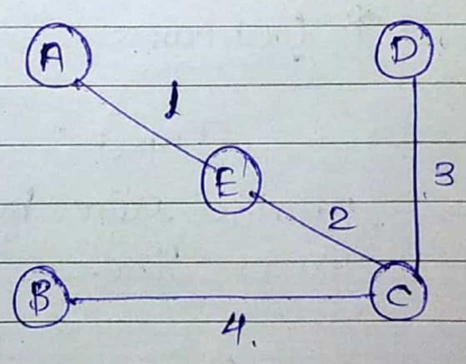




* Minimum Spanning tree :-

A spanning tree whose we of edges is minimum. is called as minimum spanning tree.





Application of minimum spanning Tree :-

minimum spanning tree are useful in constreeting hardware, by describing the way to connect a set of sities using, the smallest totaal amount of weight.

The shortest path algorithm is dijkstra algorithm.

*Dijkstra Algorithm:

Dijkra algorithm is called single Source shortest path algorithm. It it is based on" greedy techique". The algorithm maintein a list visited[] vertices whose shortest distance from the source is shortest distance of vertex is ~~it~~. already known Initially visited[i] is marked as 1 for source vertex..

At each step we mark visted [V] ass 3 vertex V is a vertex at shortest distance from the source vertex at each step of the algorithm shortest distance of each vertex is stored in an array 'distance[]'.

- Time complexity:-

There is a another move complicated property implementation called fibbonaci Heap that implements incremen prioporiory in O(1).

*Step ① Visited node Pune

visited →	1	0	0	0	0
node →	Pune	mumbai	channai	Delhi	Latur.
predecessor →	-	1	1	1	1
Distance →	0	5	∞	∞	∞

↑ min distance

* step ② visited node mumbai.

visited →	1	1	0	0	0.
node →	Pune	mumbai	channi	Delhi	latur.
precessor →	-	1	2	2	1
Distance →	0	5	10	15	∞

↑ min distance.

Step③ Visited node channai

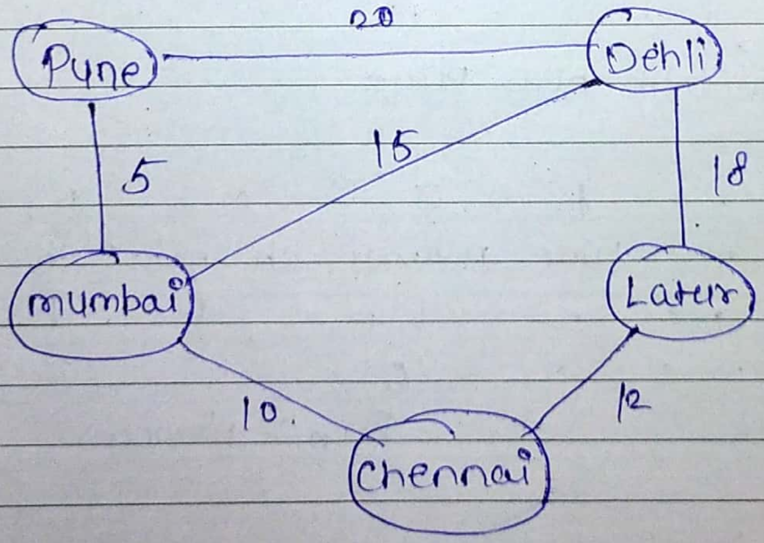
Visited →	1	1	1	0	0
node →	pune	mumbai	channai	Delhi	Latur.
perdessor →	-	1	2	2	3
Distance →	0	5	10	15	18
				↑ mindistance	

Step④ Visited node Delhi

Visited →	1	1	1	1	1
node →	pune	mumbai	chennai	Delhi	Latur.
predecessor →	-	1	2	2	3.
Distance →	0	5	10	15	18.

Spanning tree showing the shortest path and the minimum distence of each node from pune.

So that asymptanic complexity of dijsktra algo.
O(V log V+E)



* cost matrix of above graph :-

	Pune	mumbai	chennai	Delhi	Latur.
pune	0	5	0	20	0
mumbai	5	0	10	15	0
chennai	0	10	0	0	12
Delhi	20	15	0	0	18
Latur.	0	0	12	18	0.

* Solution by Dijstra algorithm:

- Initial status:

Visited →	0	0	0	0	0
node →	pune	mumbai	chennai	Delhi	Latur.
predessor →	-	-	-	-	-
Distance →	0	∞	∞	∞	∞

↑ starting node.

○ Algorithm

1) Start.

2) Create cost matrix from adjancecy matrix.

3) If there is no edge between i and j the c[i][j] is infinity.

4) Array visited is initialize to zero arr[]

```
for(i=1; i<n; i++)
    visited[i]=0.
```

5) If the vertex 0 is source vertex then visited of 0 is mark as 1.

6) create distance matrix by storing the cost of vertices from vertex no 0 to n-1 from the Source vertex 0.

7) Initial distance of Source vertex is taken as

8) choose a vertex co. such a that distance for(i=0;i<n;i++)

distance [i] = cost[0][i];
i.e distance [0] =0.

9) Stop.

* test case-:

Test case No	Test case	Expected result	Actual result	Status
T.1)	check the input after user enter input is valid.	User must enter valid i/p i.e every edge contain weight or cost.	Actually user entered input is valid.	pass
T.2)	check the i/p cost or weight of every edge is positive	please enter i/p is positive otherwise algo. no work.	User entered positive cost or weight for every edge.	pass.

* Conclusion -:

Thus we have studied and implemented the shortest path distance by using dijkstra algorithm.