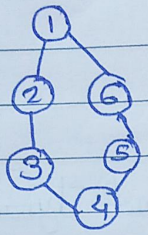
4-2023
Thursday

4. Graph Algorithm.

PAGE NO.:
DATE / /

Minimum Cost spanning Tree

It follows the concept of greedy method.



$G = (V, E)$

$|V| = 6 = \{1, 2, 3, 4, 5, 6\}$.

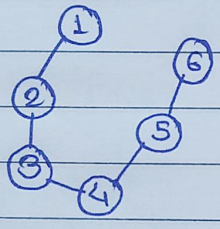
$|E| = 6 = \{(1,2), (1,6), (2,3), (3,4), (4,5), (5,6)\}$.

$|E| = |V| - 1$

$= 6 - 1$

$|E| = 5$

∴ There 5 edges of Subgraph.



So this is spanning Tree.

To represent spanning tree or subgraph

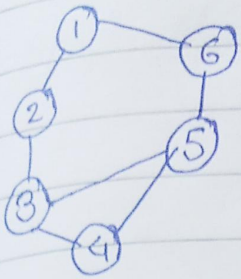
$G = (V', E')$

∴ $\boxed{\begin{aligned} V' &= V \\ E' &= |V| - 1 \end{aligned}}$

No. of spanning Tree = $|V| C_{E = |V|-1}$

$= {}^{6}C_{5} = \boxed{6}$

∴ There are 6 spanning tree we can generate from above original graph.

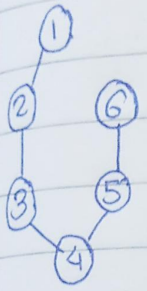


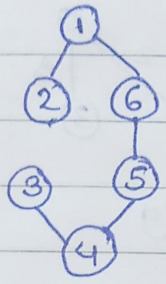
$|V| = 6$

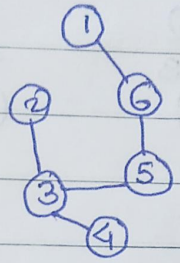
$|E| = 7$

$\therefore$ No. of spanning tree $= {}^{6}C_5$

$= 5$







Here we can't determine the exact size of spanning tree.
So change the formula.

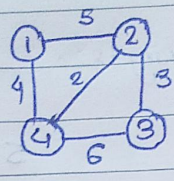
$\therefore {}^{|V|}C_{|V|-1}$ - No of cycle.

- If In a graph there are more than one cycle are present then use new formula of spanning tree
otherwise use $|V|C_{E=|V|-1}$.

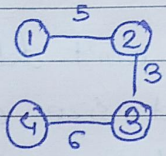
$\therefore {}^{|V|}C_{|V|-1}$ - No of cycle.

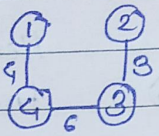
$\therefore {}^{6}C_5 - 2$

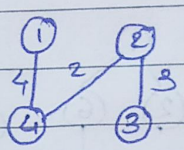
$\therefore 4$

2)  = cost = 20

Spanning tree :-







Cost = 14 = 13 = 9

Constraints :-

- Avoid

objective :-

- Find minimum cost of subgraph of original graph.

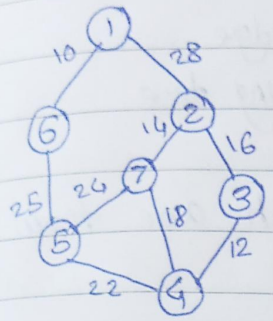
There are two method suggested from greedy method to find minimum cost spanning tree.

i) Prim's Algorithm
ii) Kruscle

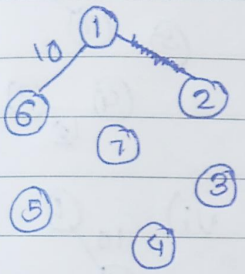
PAGE NO.:
DATE / /

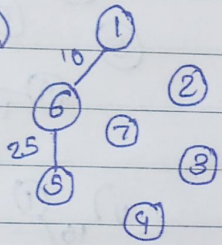
i) Prim's Algorithm :-

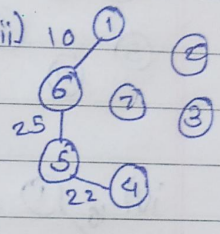
Rules:-

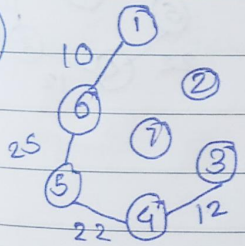


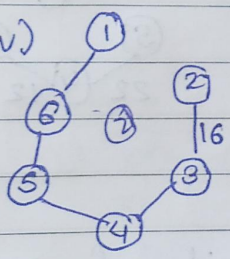
i) Select smallest edge.
ii) Take this vertices and also draws the minimum cost of edge from both vertex.
iv) Can't consider that edges which are form a cycle.

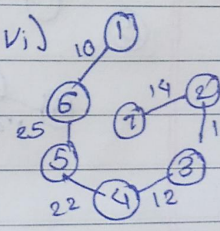
i)


ii)


iii)


iv)


v)


vi)


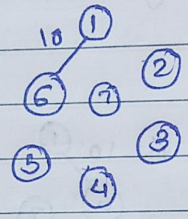
Rules of prim's Algorithm.

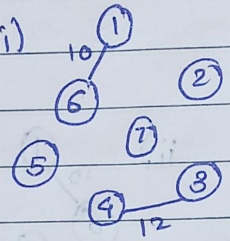
i) Consider a smallest edge from a given graph.
ii) Consider all the connected or adjacent edges from the present vertices and choose the minimum one.
iii) Repeat the same process until get minimum cost spanning tree [until and unless we cover all the vertices of the given graph and [Edges = |V| - 1]].

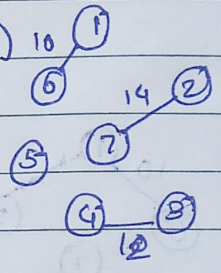
ii) Kruskal Algorithm

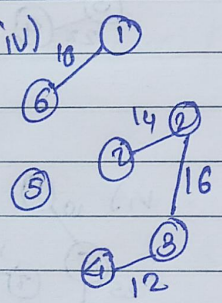
i) Get minimum cost edge
ii) Get second minimum cost edge
iii) Repeat until we get spanning tree.

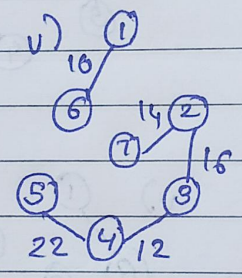
Not consider that edges which are form a cycle.

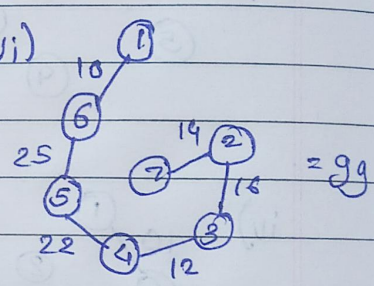
i) 

ii) 

iii) 

iv) 

v) 

vi) 

= gg

- **Time Complexity of Minimum cost Spanning Tree**

V = n
E = n-1

∴ O(|V| · |E|)
O(n · (n-1)) = O(n²).

Min Heap
T = log n

O(V · E)
O(n · log n)
O(log n)

Min Heap is used to reduce the time complexity. There don't need of comparison.

PAGE NO.:
DATE / /

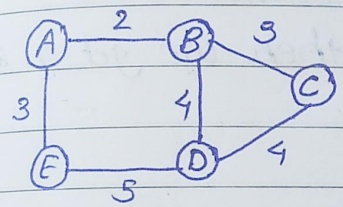
Note:-

MinHeap :-

Basically minHeap is data structure it use to give you a retrive minimum cost edge from a given Graph without making a comparison of remaining edges of a given graph.

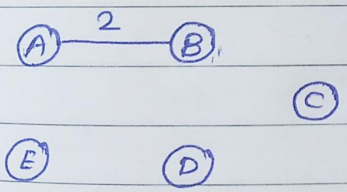
logn = time taken by minHeap to retrive minimum cost edge.

21-4-2023 Friday

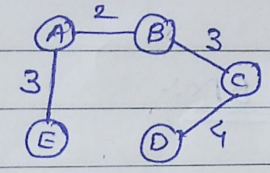


Prims Algorithm.

i)



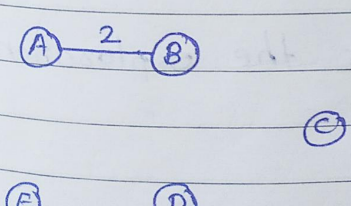
ii)



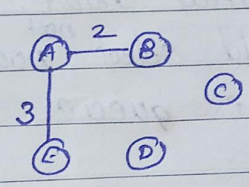
cost = 12.

Kriscale Algorithm.

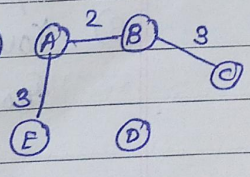
i)



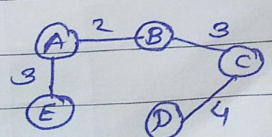
ii)



iii)

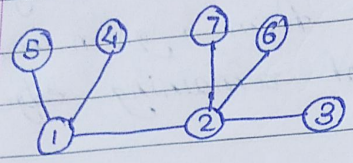


iv)



Cost = 12.

- Deapth first search Algorithm (DFS)
- Breadth first search Algorithm (BFS)

1) 

BFS:

1, 2, 4, 5, 7, 6, 3. or 1, 4, 2, 5, 7, 6, 3

Here first we start from vertex 1 and explore them then we get 5, 4, 2 vertex.
Now we explore 2 vertex then we get 7, 6, 3.
Here we explore all vertex

DFS:

1, 2, 3, 6, 7, 4, 5

Rule of BFS:-

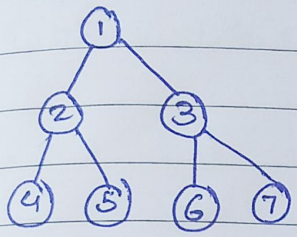
1) You may start a graph traversing from any node
2) Onee you select the vertex try to completely explore that node.
3) You should select next node for the exploration fforom the queue.

PAGE NO.:
DATE / /

Rule of DFS:

1) You may start a Graph traversing from any node.
2) Instead of exploring all the vertex considering only one adjacent node.
3) You should select one node from the stack.

2)



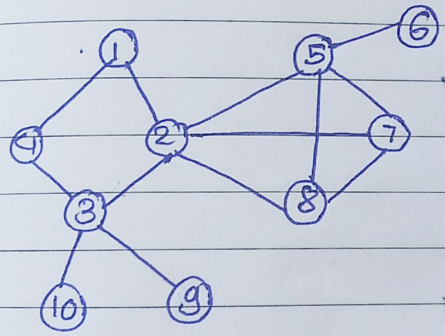
BFS:

1, 2, 3, 4, 5, 6, 7 } levelorder

DFS:

1, 2, 4, 5, 3, 6, 7 } Preorder

3)



BFS:

i) BFS: 1

Q | 1 | | | |

ii) BFS: 1, 4, 2

Q: | 1 | 4 | 2 | |

iii) BFS: 1, 4, 2, 3

Q: | x | 4 | 2 | 3 |

iv) BFS: 1, 4, 2, 3

Q: | x | 4 | 2 | 3 |

v) BFS: 1, 4, 2, 3, 5, 7, 8

Q: | x | x | 2 | 3 | 5 | 7 | 8 |

vi) BFS: 1, 4, 2, 3, 5, 7, 8, 9, 10

Q: | x | x | x | 3 | 5 | 7 | 8 | 9 | 10 |

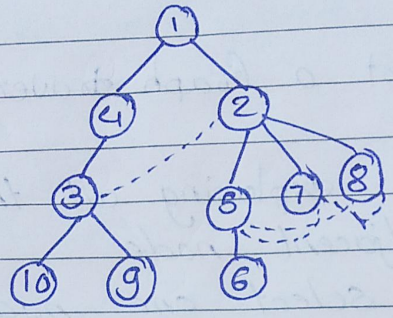
vii) BFS: 1, 4, 2, 3, 5, 7, 8, 9, 10, 6

Q: | x | 4 | x | 3 | 5 | 7 | 8 | 9 | 10 |

6

DATE / /

Tree :-



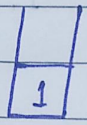
dotted line :-
already present is so/?

DFS :-

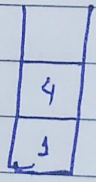
i) DFS : 1

Stack :

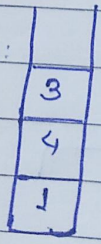
ii) DFS :- 1, 4 S :



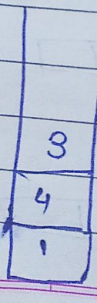
iii) DFS :- 1, 4, 3 S :



iv) DFS :- 1, 4, 3, 10 S :



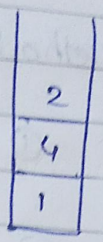
v) DFS :- 1, 4, 3, 10, 9, 5 S :



PAGE NO.:
DATE / /

v.) DFS: 1, 4, 3, 10, 9, 2, 8

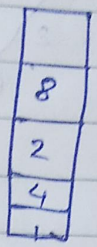
S:



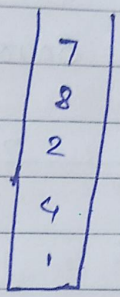
Here POP 3.

vi.) DFS: 1, 4, 3, 10, 9, 2, 8, 7.

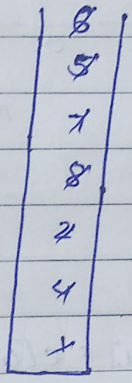
S:



viii) DFS: 1, 4, 3, 10, 9, 2, 8, 7, 5 : S:

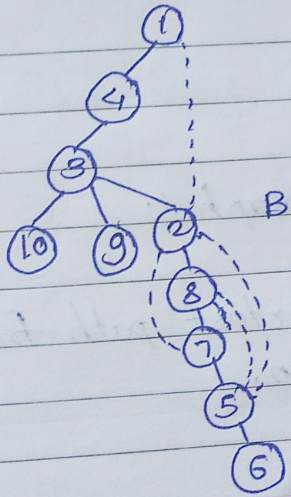


ix) DFS: 1, 4, 3, 10, 9, 2, 8, 7, 5, 6 S:



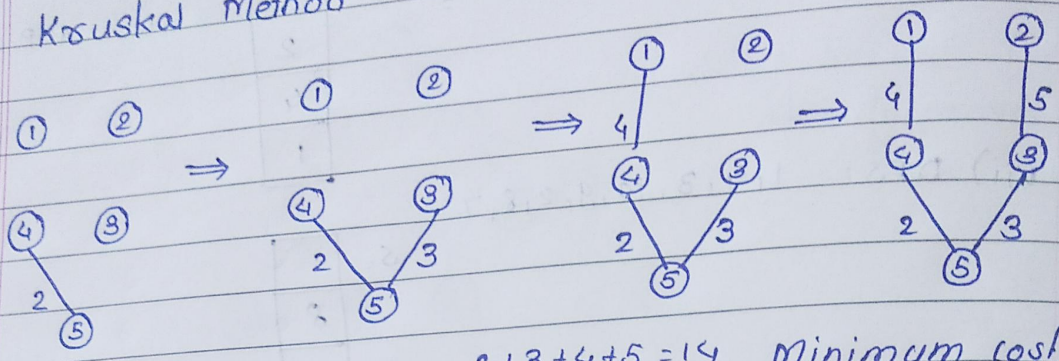
Here explore all so pop all vertex.

Tree :-



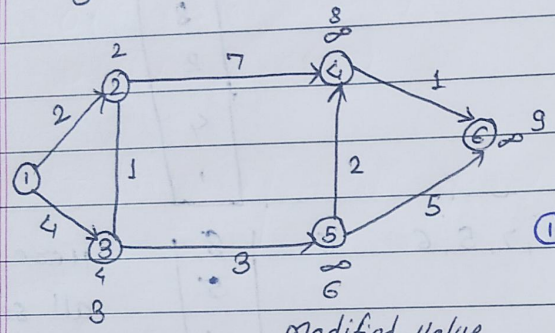
Back edges.

- Kruskal Method



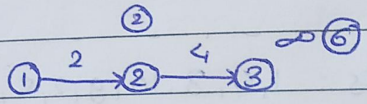
2+3+4+5 = 14 Minimum cost

- Single Source shortest Path (Dijekstra's Algorithm):



Modified value

Here 2 to 3 there is a no direct edge.



6 is less than ∞ so, we can change / modified value of vertex - 3 it is called Relaxation.

if (d[u] + c[u,v] ≤ d[v])

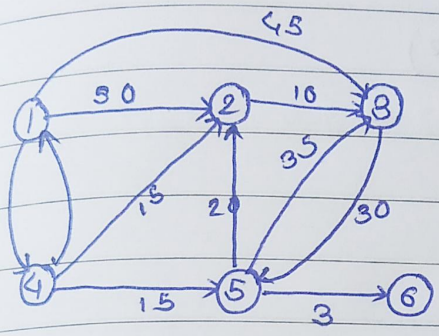
2 + 4 = 6 ≤ ∞

d[v] = d[u] + c[u,v]

Applied on weighted graph :-

Objective :- Find shortest path from source to destination.

V	d[v]
2	2
3	3
4	8
5	6
6	9



	2	3	4	5	6	
From Node						
[4]	50	45	10	∞	∞	
smallest cost	50	45	10	25	∞	Vertex 4 cost 10+15=25
[5]	50	45	(10)	(25)	∞	
[2]	45	45	(10)	(25)	∞	From 5th node value of 5 vertex is 25+20=45 45<50 so New value of vertex 2 is 45
[3]	(45)	(45)	(10)	(25)	∞	
[6]	(45)	(45)	(10)	(25)	(∞)	

Draw separate table for each stage.

V	d[V]
2	45
3	45
4	10
5	25
6	∞

- Time Complexity : $O(n^2)$

$O(E \cdot No\ of\ Relax)$

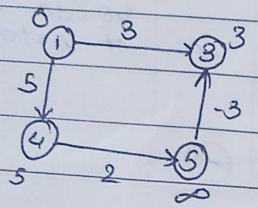
$O(n \cdot n) = O(n^2)$

$E = \left[\frac{n(n-1)}{2}\right]$ - if graph is complete graph.

$O\left(\left(\frac{n(n-1)}{2}\right) \cdot n\right) = O(n^2 \cdot n)$

$= O(n^3)$

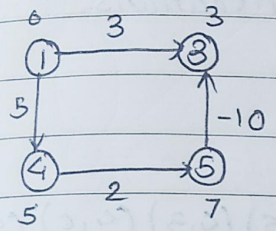
- Drawback / Limitation of dijstra's algorithm



start from vertex (1)

	3	4	5
3	③	5	∞
4	③	⑤	∞
5	③	⑤	⑦
	③	⑤	⑦

v	d[v]
3	3
4	5
5	7

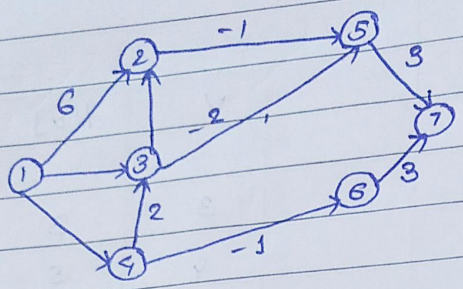


With this example with we got -3 value of vertex 8 so we can't apply dijkstra's algorithm on this example.

3	4	5	v	d[v]
			3	-3
③	5	∞	4	5
③	⑤	∞	5	7
③	⑤	⑦		
-3	⑤	⑦		

Dijkstra's algorithm does not allows negative cost of vertex so in above example dijkstra's algorithm does not work.

* Bellman - Ford Algorithm:



if (d[u] + c[u,v] < d[v])
d[v] = d[u] + c[u,v]

Relaxation of Edges = |V| - 1 = 7 - 1 = 6 iterations.
↓
No. of vertex

Edges : (1,2) (1,3) (1,4) (2,5) (3,2) (3,5) (4,3) (4,6) (5,7) (6,7)

4. May. 2023
Thursday

Time Complexity of Bellman - Ford

R = |V| - 1
O(|E| · (|V| - 1)) ... O(no. of edges x no of relaxation)
O(n · (n-1))
O(n²)

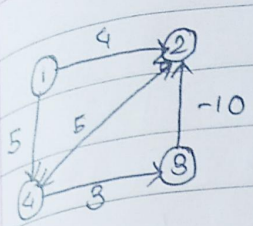
* Complete Graph = There is edge present between every pair of vertices. That is pretend as complete Graph.

$E = \frac{n(n-1)}{2}$

$O\left(\frac{n(n-1)}{2} \cdot (n-1)\right)$

$O(n^3)$

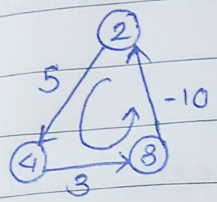
Drawaback - doesn't handle negative weight cycle.



Apply Bellman-Ford Algorithm.

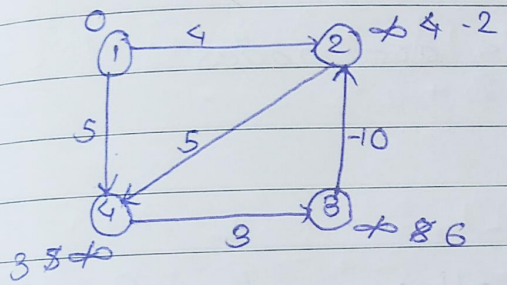
$E = \{(3,2), (4,3), (1,4), (1,2), (2,4)\}$.

$|V| - 1 = 4 - 1 = 3$



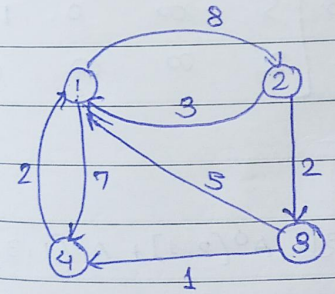
$5 + 3 + (-10) = -2$

∴ Bellman Ford doesn't handle negative weight cycle.



8 time Relaxation.

- All pair shortest path Algorithm. (Floyd warshell's Algo)
  - Determine or find shortest path we determine all of the edges.

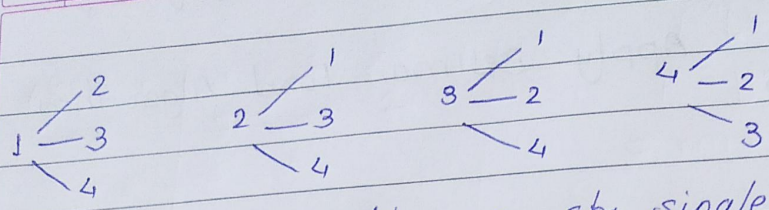


Form matrix.

A^0 :	1	2	3	4
1	0	8	∞	7
2	3	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

∴ Self loop are not present so diagonal elements are 0.

∴ ∞ - not directly connected.



This is look like as ~~the~~ single shortest path Algorithm

Time complexity of single ~~the~~ source shortest path Algorithm is :- $o(n^2)$

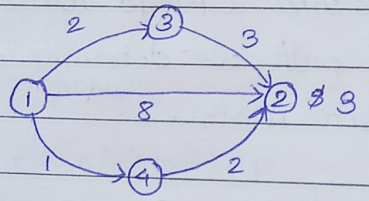
∴ How we apply single source shortest path on all ~~path~~ shortest path algorithm.

$o(4n^2)$ - Here 4 are nodes

$o(n \cdot n^2)$

$o(n^3)$

∴ Time complexity of all pair shortest path algorithm is $\underline{\underline{o(n^3)}}$



$A^0 =$

	1	2	3	4
1	0	8	∞	7
2	3	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

$A^1 =$

	1	2	3	4
1	0	8	∞	7
2	3	0	2	15
3	5	8	0	1
4	2	5	∞	0

$\therefore A^0[2,3] = A^0[2,1] + A^0[1,3]$

$2 = 8 + \infty$

$2 < \infty$

$\therefore A^0[2,4] = A^0[2,1] + A^0[1,4]$

$= 8 + 7$

$\infty, 7 = 15$

PAGE NO.:
DATE / /

$A^0[3,2] = A^0[3,1] + A^0[1,2]$
$= 5 + 3$
$\infty > = 8$

$A^0[3,4] = A^0[3,1] + A^0[1,4]$
$1 < 5 + 7$
$1 < 12$

$A^0[4,2] = A^0[4,1] + A^0[1,2]$
$\infty = 2 + 3$
$\infty > 5$

$A^0[4,3] = A^0[4,1] + A^0[1,3]$
$\infty > 2 + \infty$
∞

$A^2 =$

	1	2	3	4
1	0	3	5	7
2	8	0	2	15
3	5	8	0	1
4	2	5	7	0

$A^1[1,3] = A^1[1,2] + A^1[2,3]$
$\infty = 3 + 2$
$\infty > = 5$

$A^1[1,4] = A^1[1,2] + [2,4]$
$7 < 3 + 15$
$7 < 18$

$A^1[3,4] = A^1[3,2] + A^1[2,4]$
$= 1 < 8 + 15 \quad = 1 < 24$

$A^1[4,1] = A^1[4,2] + A^1[2,1]$
$2 < 5 + 8$
$2 < 13$

$A^1[3,1] = A^1[3,2] + A^1[2,1]$
$5 < 8 + 5$
$5 < 13$

$A^1[4,3] = A^1[4,2] + [2,3]$
$= 5 + 2$
$\infty > 7$

$A^3 =$

	1	2	3	4
1	0	3	5	6
2	7	0	2	8
3	5	8	0	1
4	2	5	7	0

$A^2[1,2] = A^2[1,3] + A^2[3,2]$
$3 = 5 + 8$
$3 <= 13$

$A^3[1,4] = A^2[1,3] + A^2[3,4]$
$7 > 5 + 1$
> 6

$A^2[2,1] = A^2[2,3] + A^2[3,1]$
$8 > 2 + 5 = 8 > 7$

$A^2[2,4] = A^2[2,3] + A^2[3,4]$
$15 > 2 + 1$
$15 > 3$

$A^2[4,1] = A^2[4,3] + A^2[3,1]$
$2 < 7 + 5$
$2 < 12$

$A^2[4,2] = A^2[4,3] + A^2[3,2]$

$A^4 =$

	1	2	3	4
1	0	3	5	6
2	5	0	2	3
3	3	6	0	1
4	2	5	7	0

$A^4[3,1] = \min\{A^3[3,1], A^3[3,4] + A^3[4,1]\}$
$= \min\{5, 1+2\} = 3$

$A^4[3,2] = \min\{A^3[3,2], A^3[3,4] + A^3[4,2]\}$
$= \min\{8, 1+5\} = 6$

$A^3(1,2) = A^3(1,4) + A^3(4,3)$
$3 = 6 + 7$
$3 <= 12$

$A^4(1,3) = A^3(\min\{A^3[1,3], A^3[1,4] + A^3[4,3]\})$
$= \min\{5, 6+7\} = 5$

$A^4(2,1) = \min\{A^3[2,1], A^3[2,4] + A^3[4,1]\}$
$= \min\{7, 3+2\} = 5$

$A^4[2,3] = \min\{A^3[2,3], A^3[2,4] + A^3[4,3]\}$
$= \min\{2, 3+7\} = 2$

Formula:-

$$A^k[i,j] = \min\{A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j]\}$$

Code:-

```
for (k=1; k<=n; k++)
{
    for (i=0; i<=n; i++)
    {
        for (j=1; j<=n; j++)
        {
            A^k[i,j] = min{A^(k-1)[i,j], A^(k-1)[i,k] + A^(k-1)[k,j]}
        }
    }
}
```

used to generate matrix.