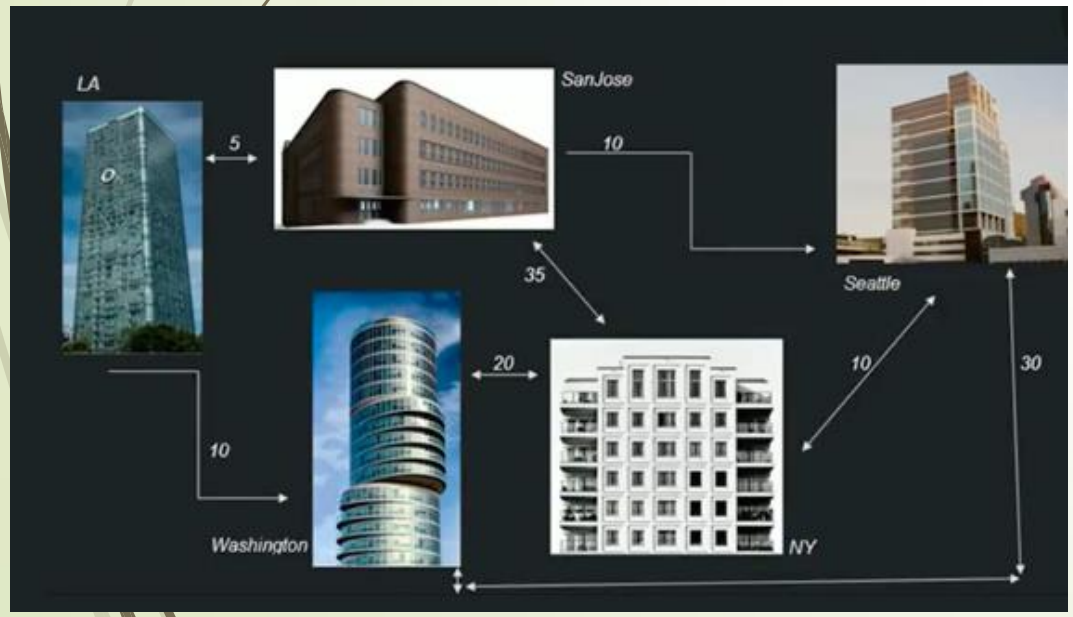
Data Structure and Algorithms

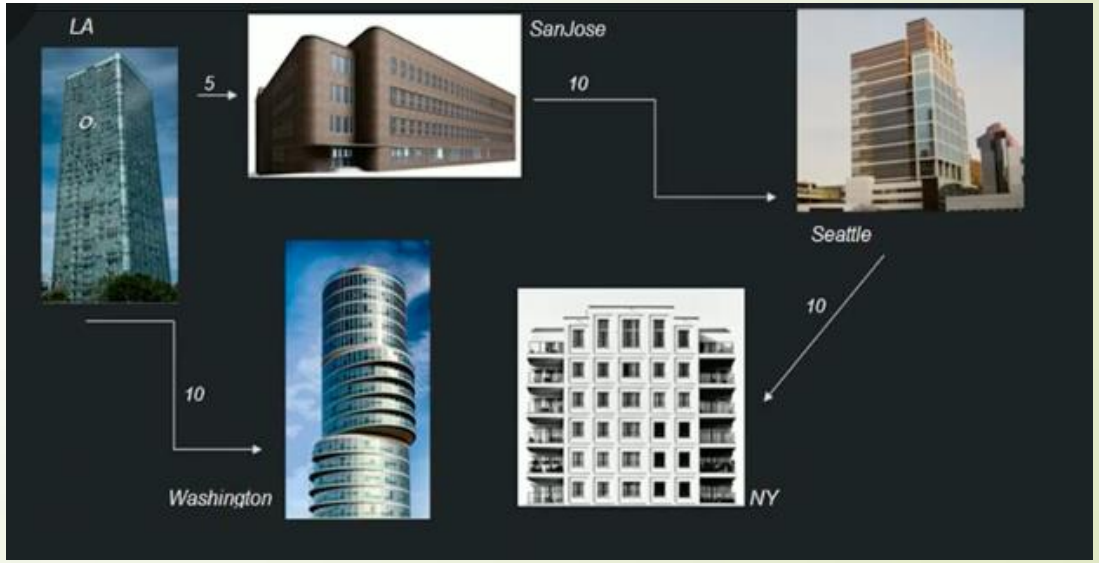
Session-28

Dr. Subhra Rani Patra
SCOPE, VIT Chennai

What is 'Single Source Shortest Path' ?

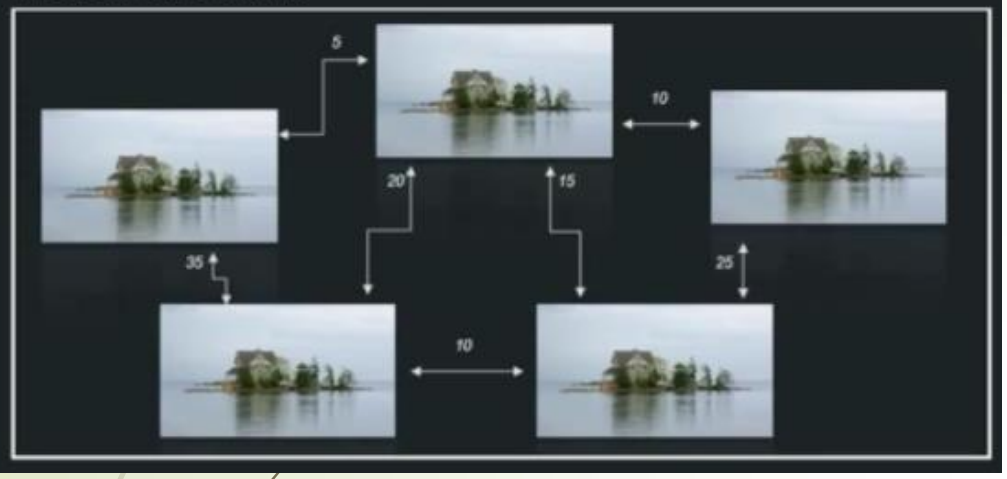
- ✓ Definition:
  - ✓ Single source shortest path problem is about finding a path between a given vertex (called 'Source') to all other Vertices in a graph such that, the total distance between them (source & Destination) is minimum.
- ✓ Example
  - ✓ Let's say we have office in 5 different cities and we need to travel from Head office to all other offices.
  - ✓ Flight charges between cities are known (as given in below diagram).
- ✓ What is the cheapest way to reach each office from HQ ?



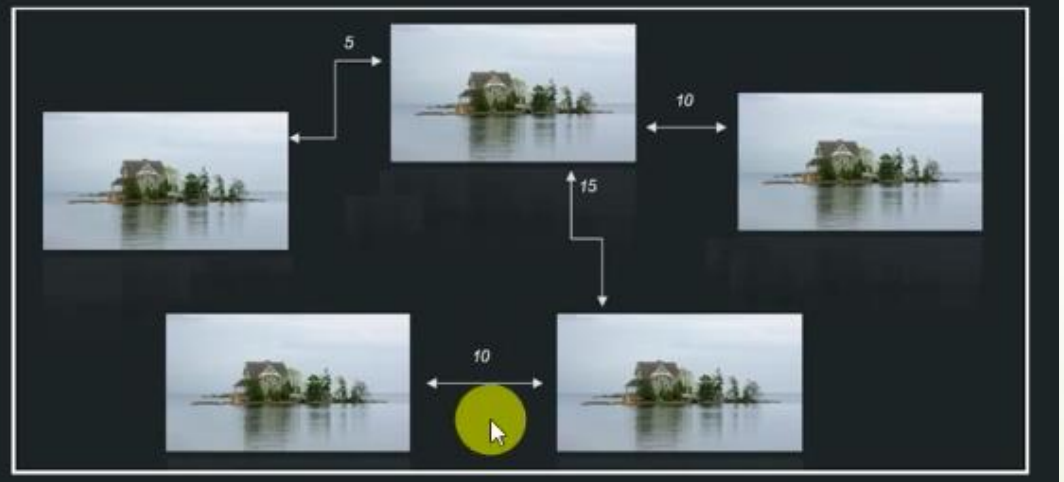


Single Source Shortest Path vs Minimum Spanning Tree(MST) :

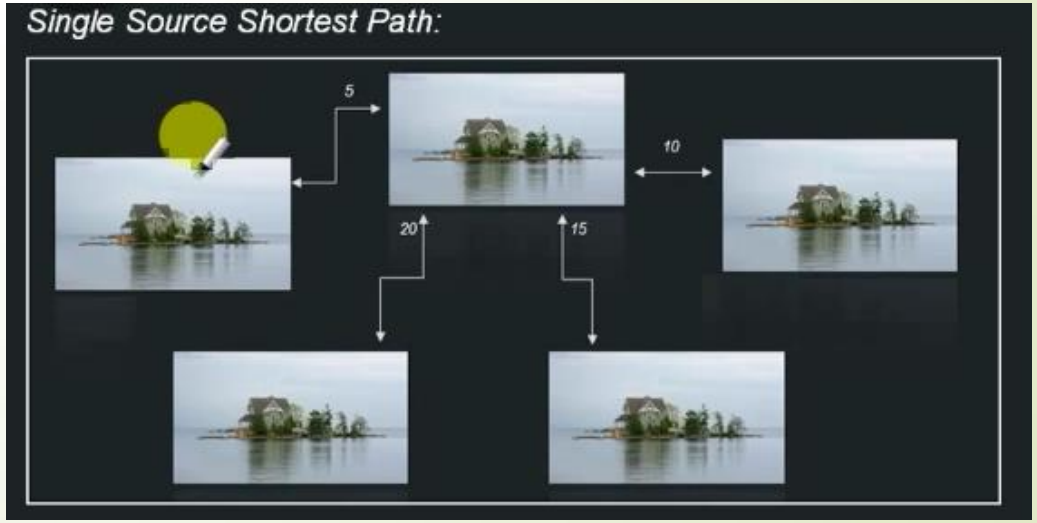
Problem Statement:



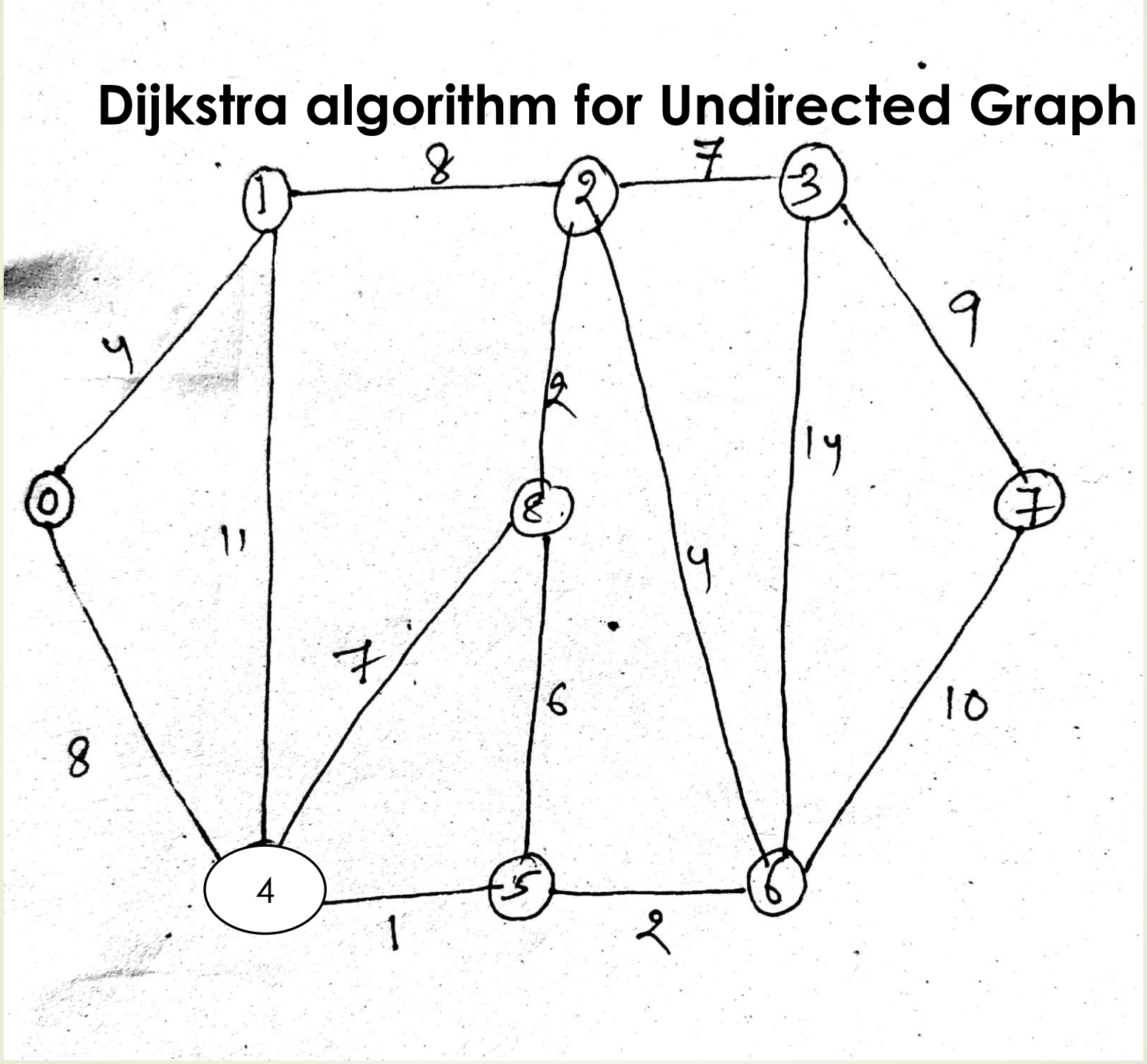
Minimum Spanning Tree:

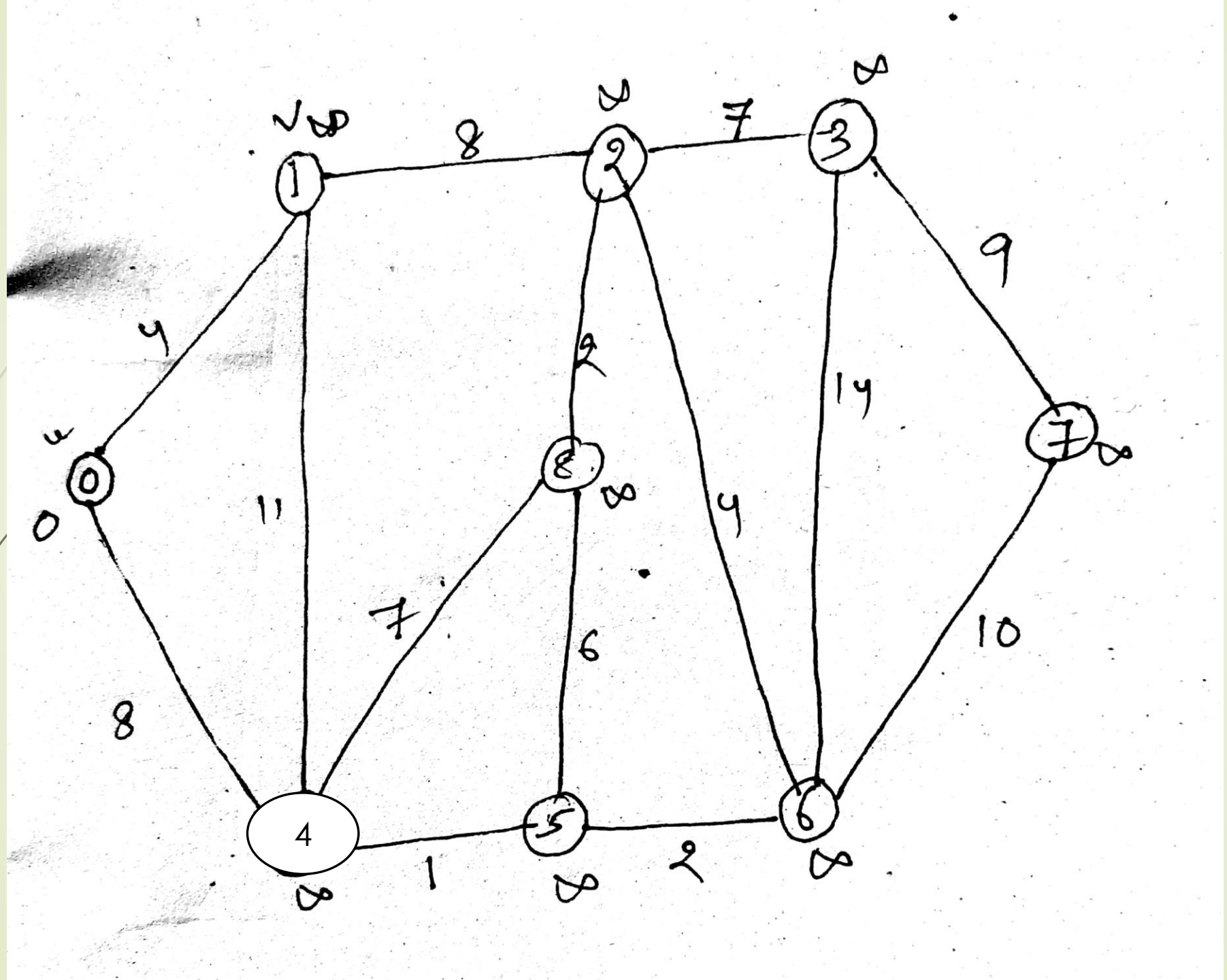


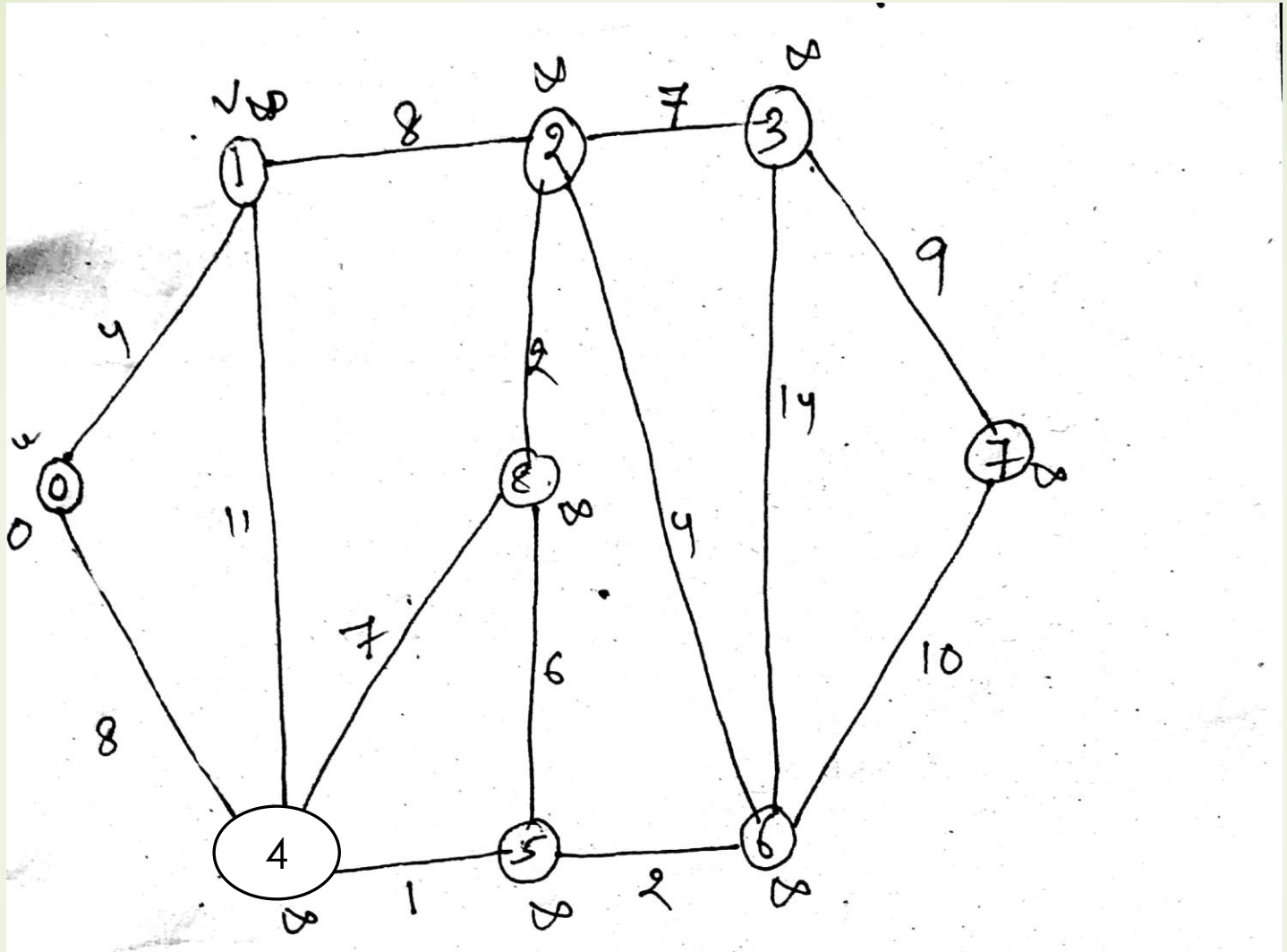
Single Source Shortest Path:



Dijkstra algorithm for Undirected Graph

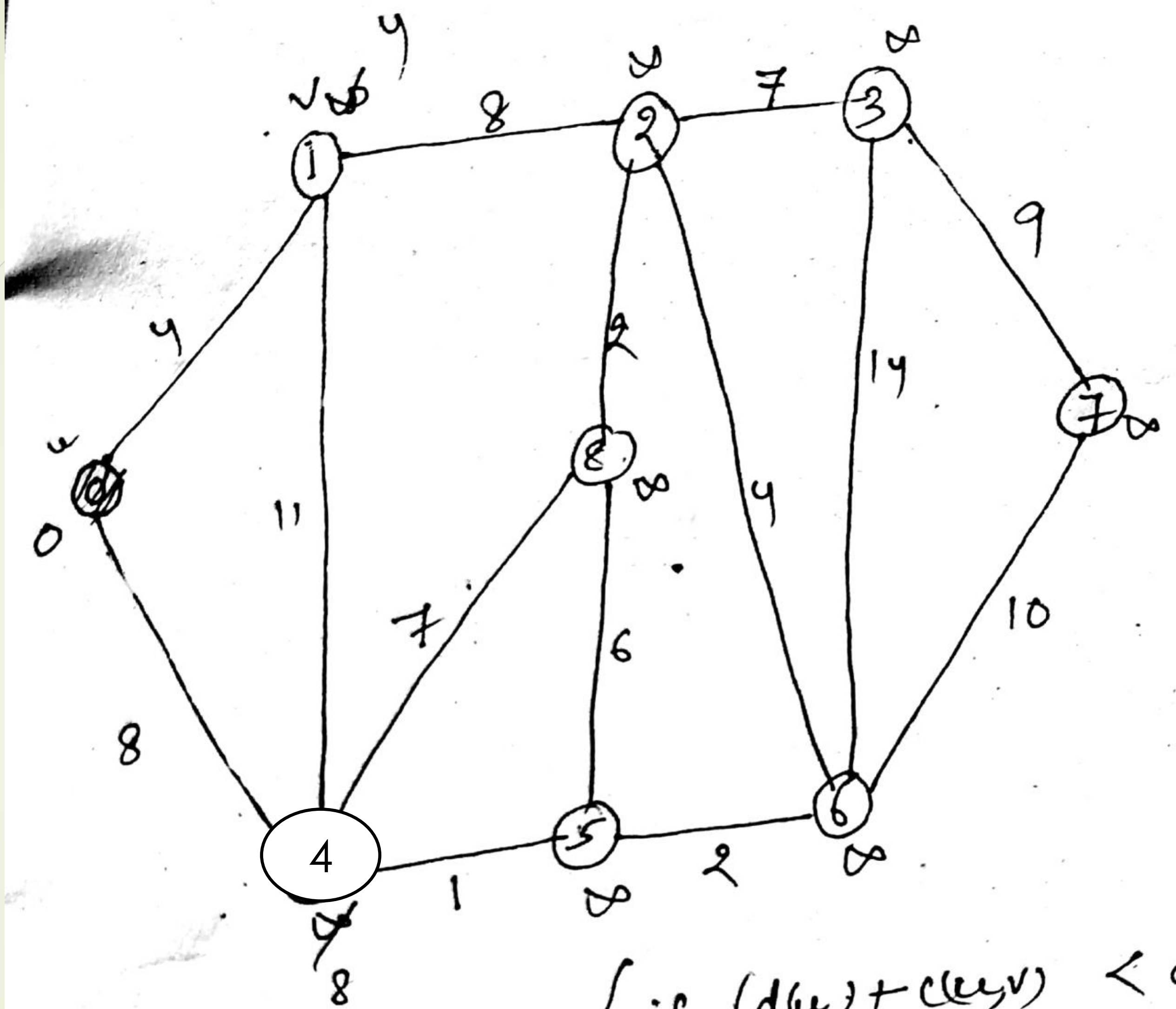



0
1
2
3
4
5
6
7
8
8
11
7
6
4
14
9
10
1
2
4

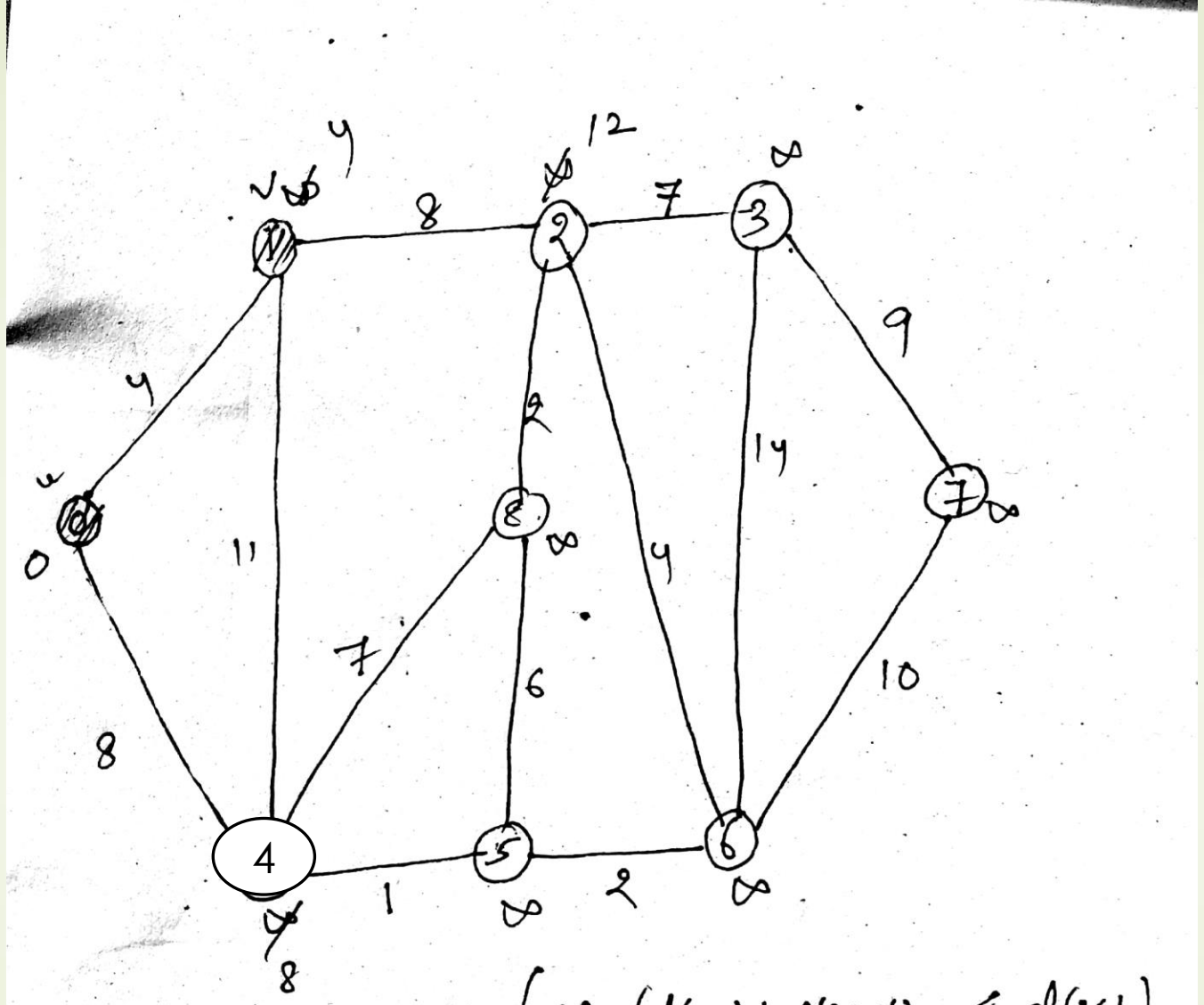


if (d(u) + c(u,v) < d(v))

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



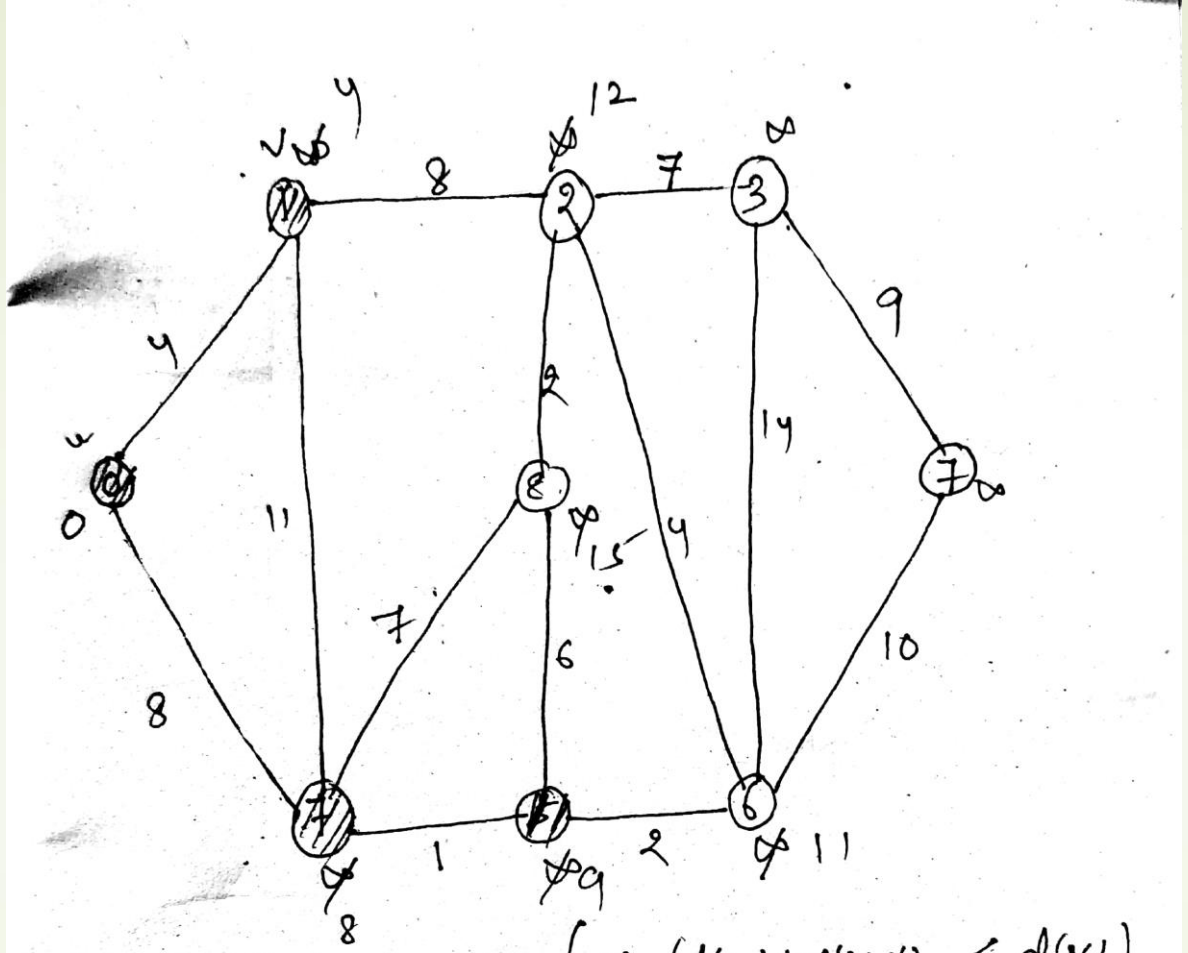
$$\{\text{if } (d(u) + c(u,v) < d(v))$$
$$d[v] = d[u] + c(u,v)\}$$



{ if (d(u) + c(u,v) < d(v))

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

4 + 8 < ∞



{ if (d(u) + c(u,v) < d(v))

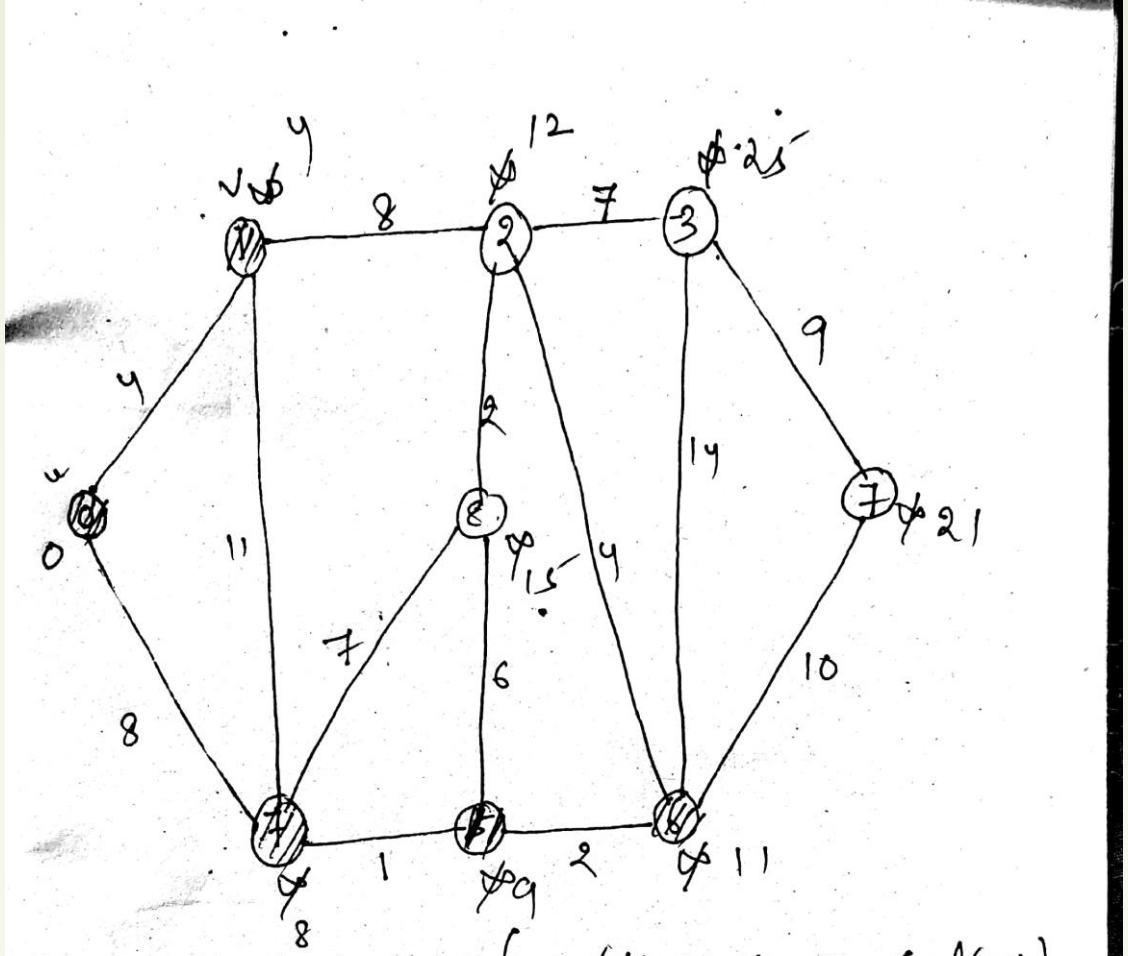
d[v] = d[u] + c(u,v)]

4 + 8 < ∞

3 + 1 < 4

7 + 8 < 15

9 + 2 < 11



{ if (d(u) + c(u,v) < d(v))

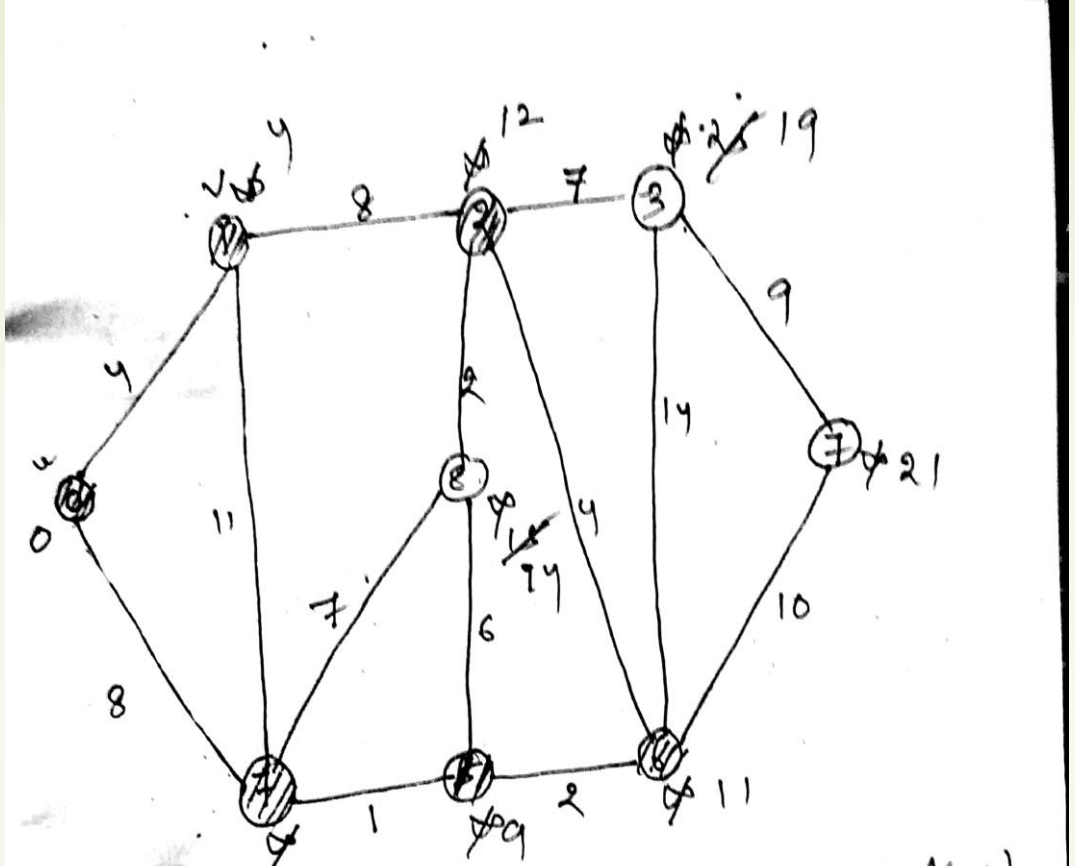
d[v] = d[u] + c(u,v)]

4 + 8 < ∞

3 + 1 < ∞

7 + 2 < ∞

9 + 2 < ∞



{ if (d(u) + c(u,v) < d(v))

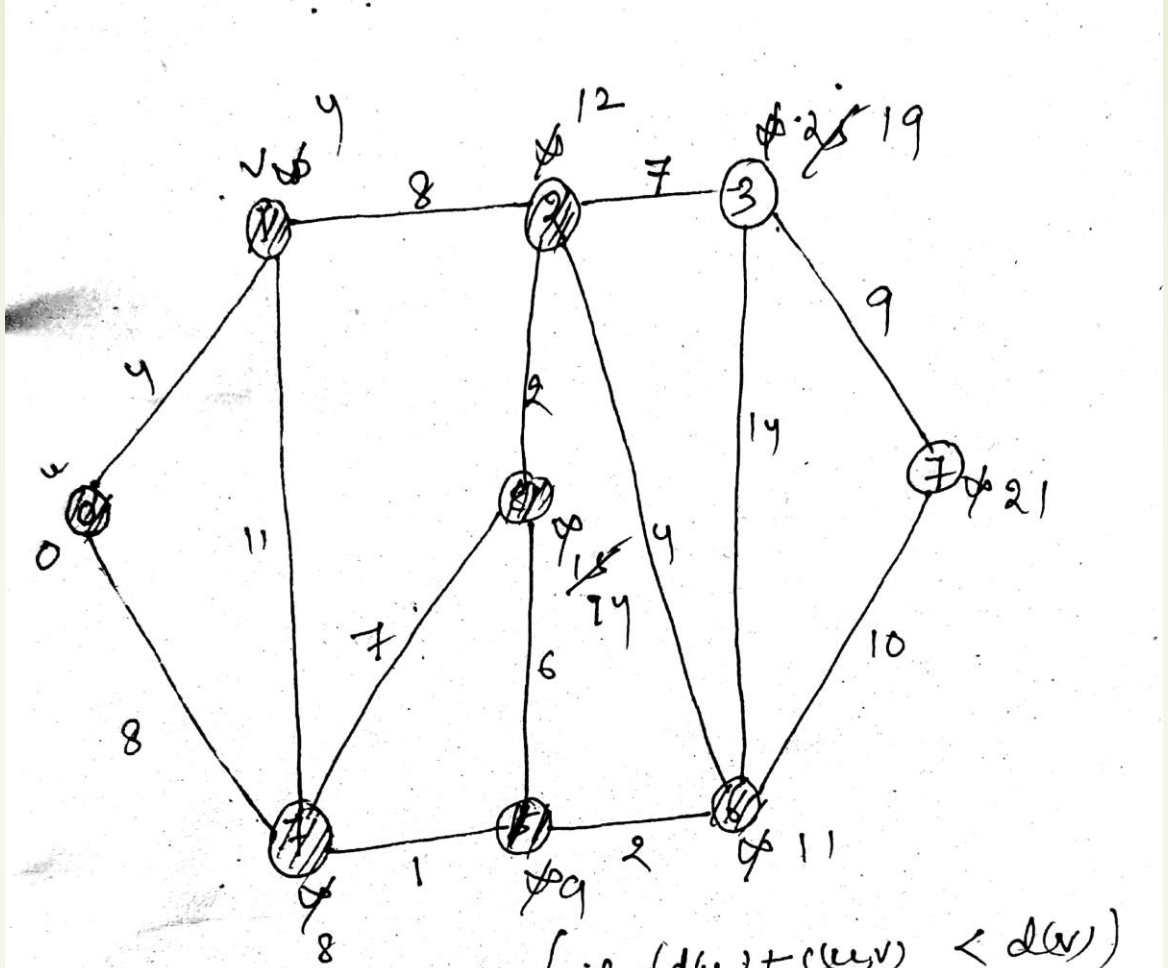
d[v] = d[u] + c(u,v)]

4 + 8 < ∞

2 + 1 < ∞

7 + 8 < ∞

9 + 2 < ∞



{ if (d(u) + c(u,v) < d(v))

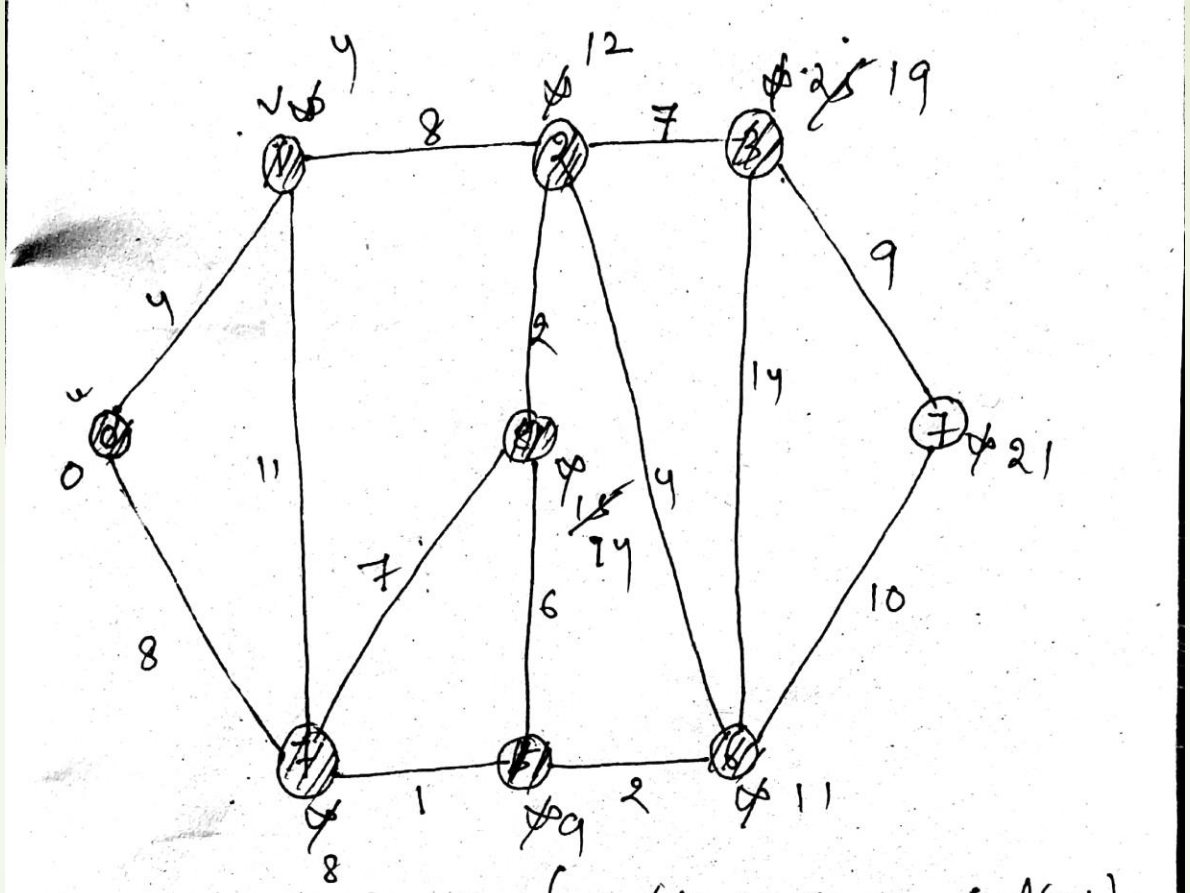
d[v] = d[u] + c(u,v)]

4 + 8 < ∞

2 + 1 < ∞

7 + 8 < ∞

9 + 2 < ∞



{ if (d(u) + c(u,v) < d(v))

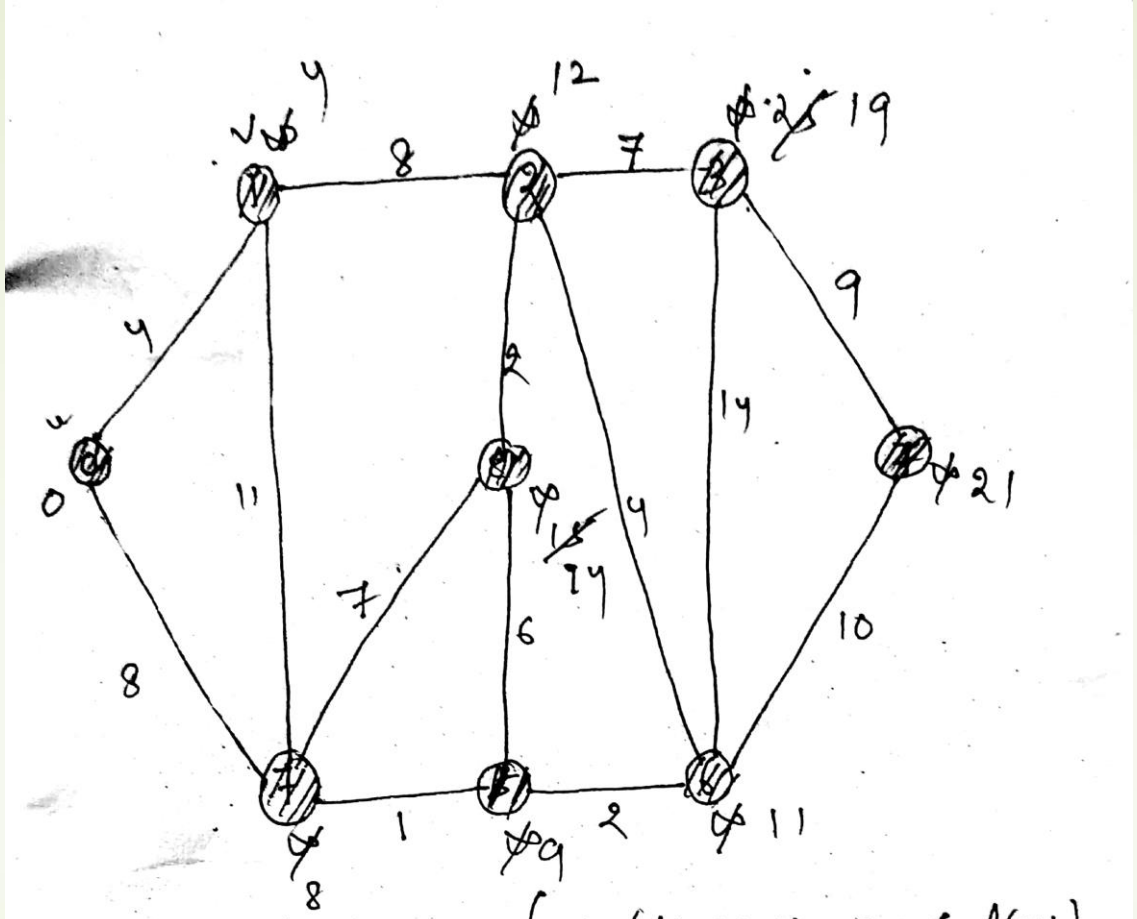
d[v] = d[u] + c(u,v)]

4 + 8 < ∞

2 + 1 < ∞

7 + 8 < ∞

9 + 2 < ∞



{ if (d(u) + c(u,v) < d(v))

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

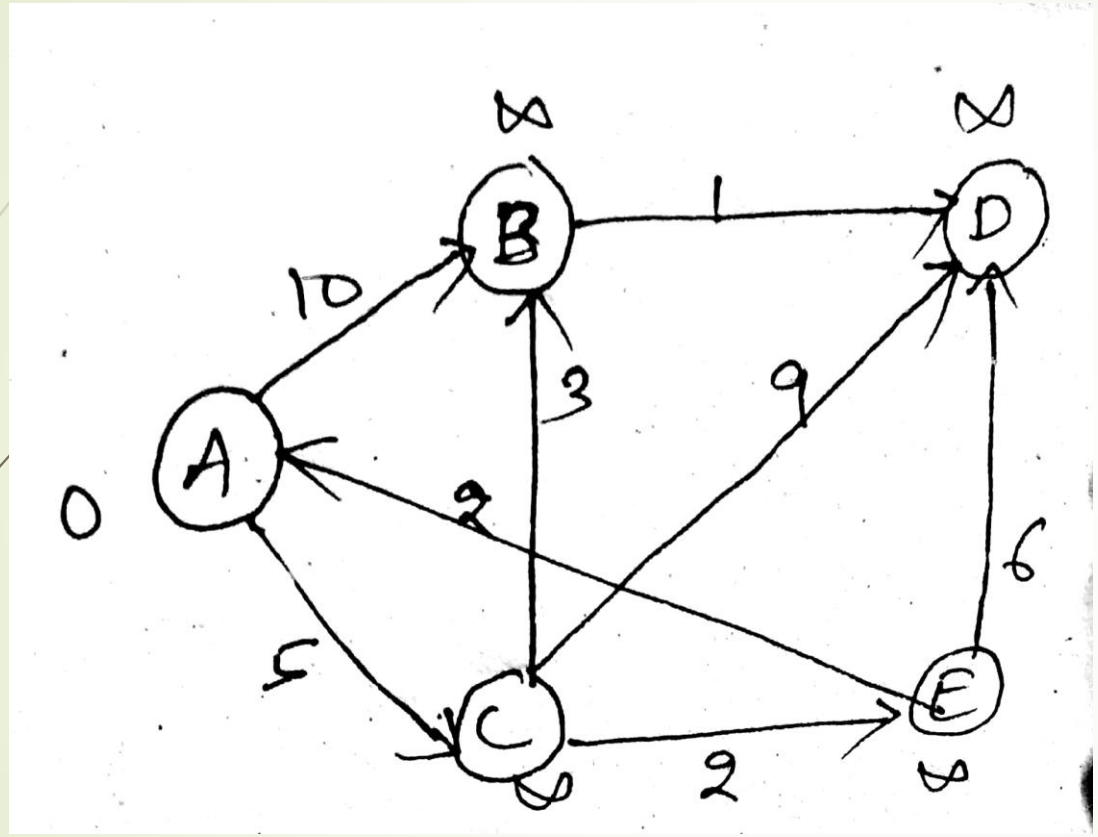
4 + 8 < ∞

2 + 1 < ∞

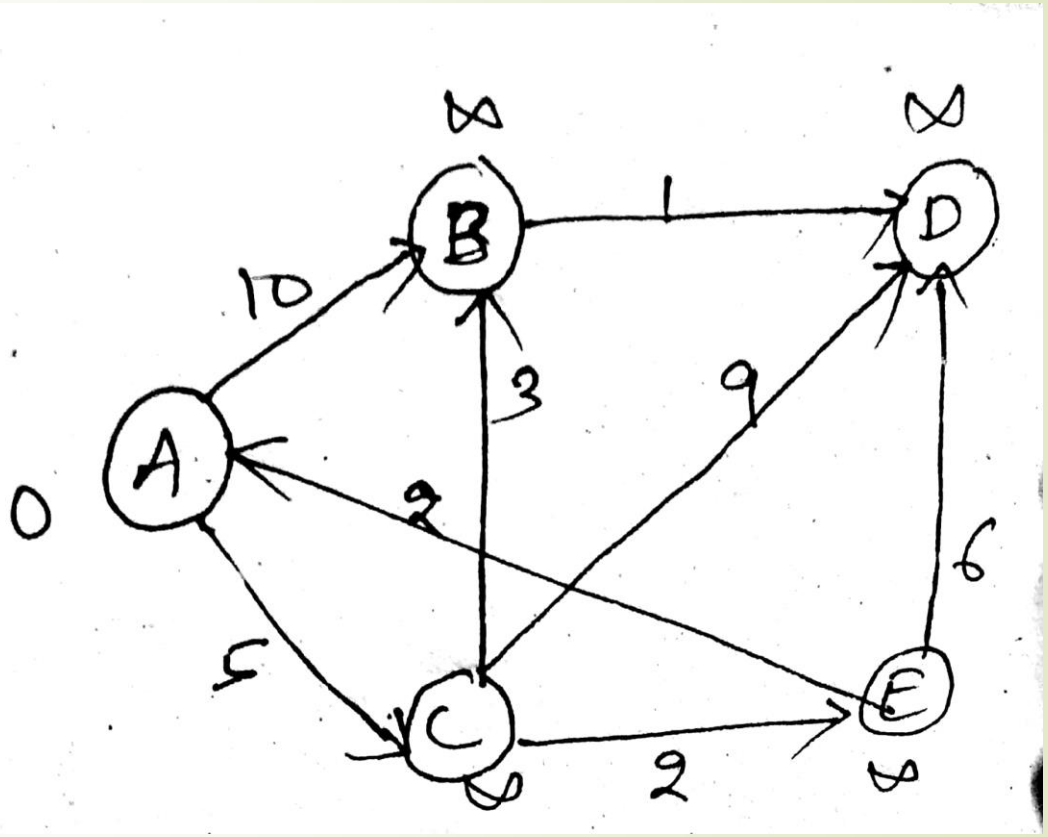
7 + 8 < ∞

9 + 2 < ∞

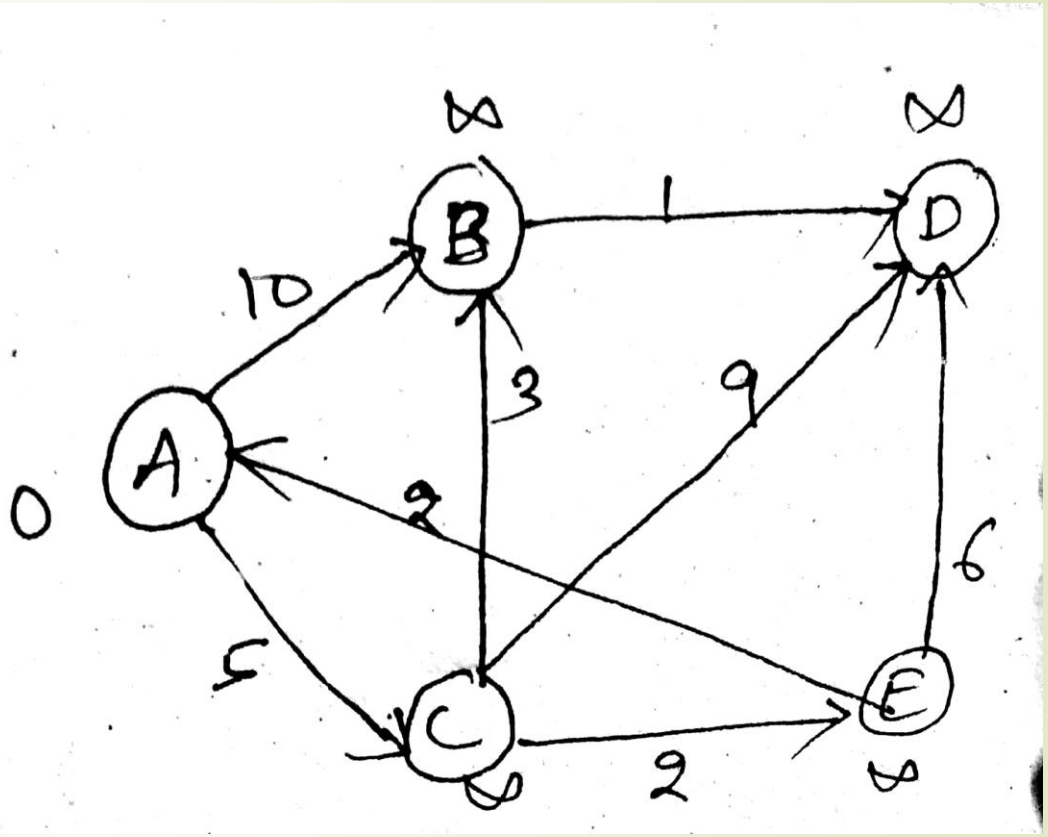
Directed Graph



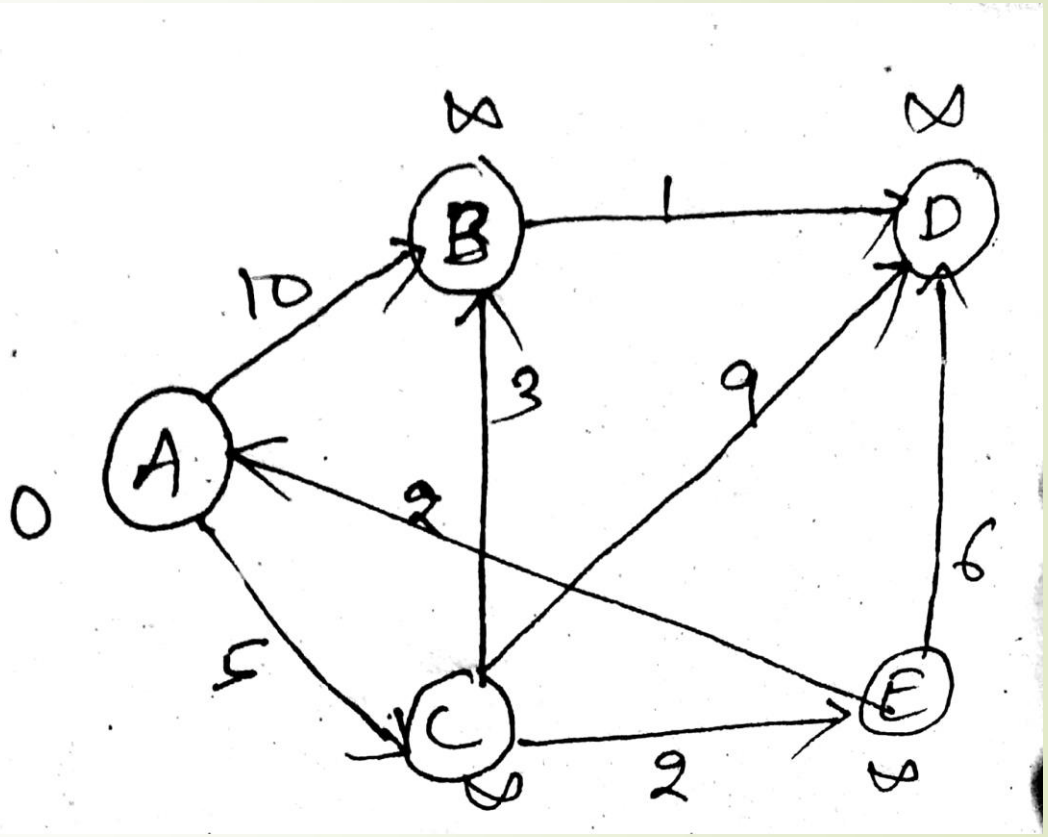
	A	B	C	D	E
:	0	∞	∞	∞	∞
:					



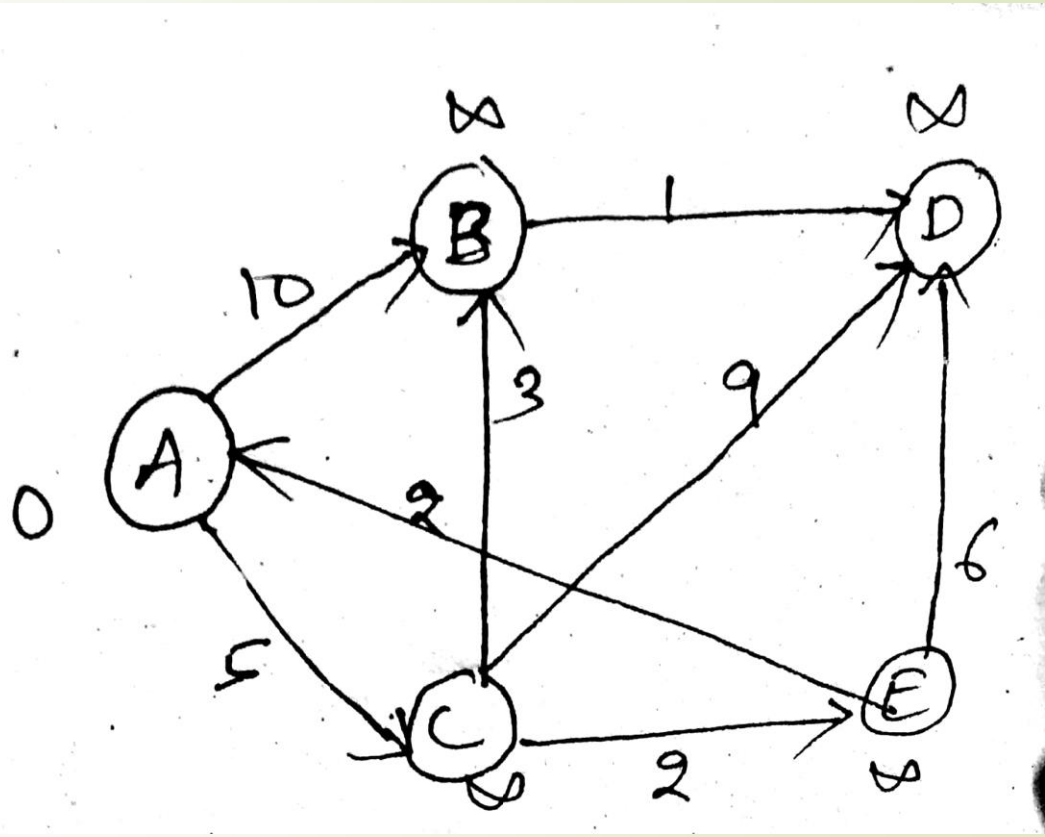
	A	B	C	D	E
A	0	∞	∞	∞	∞



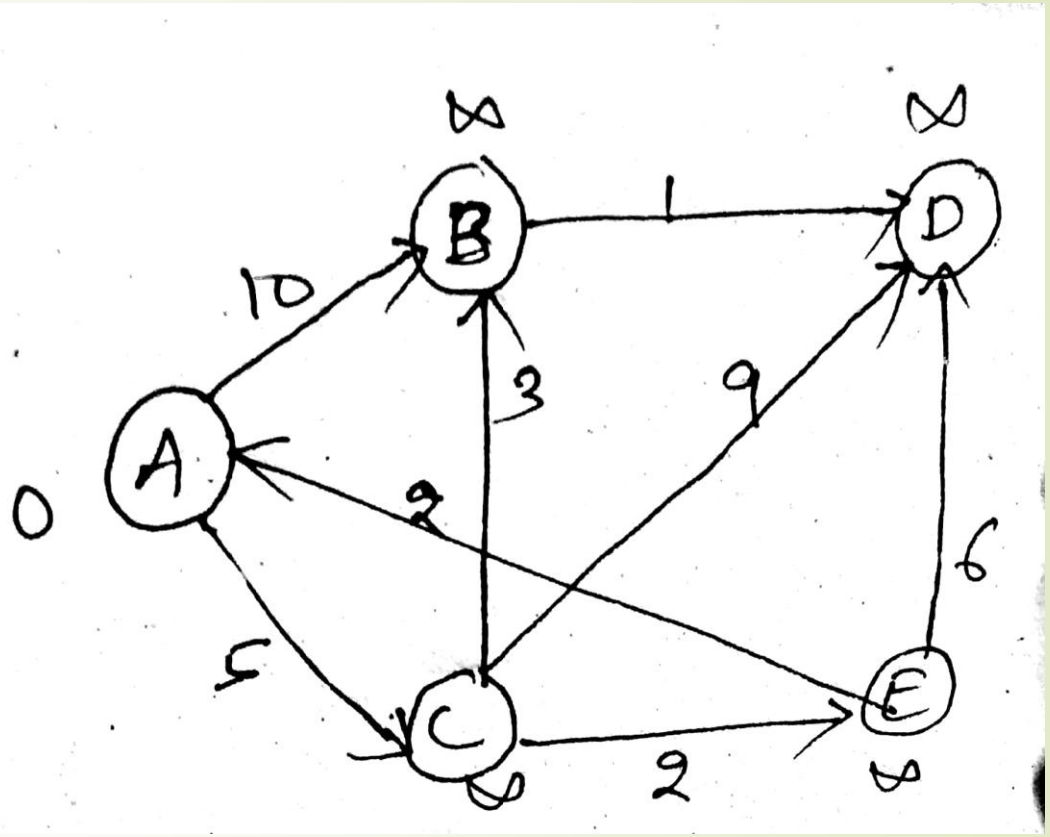
	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞



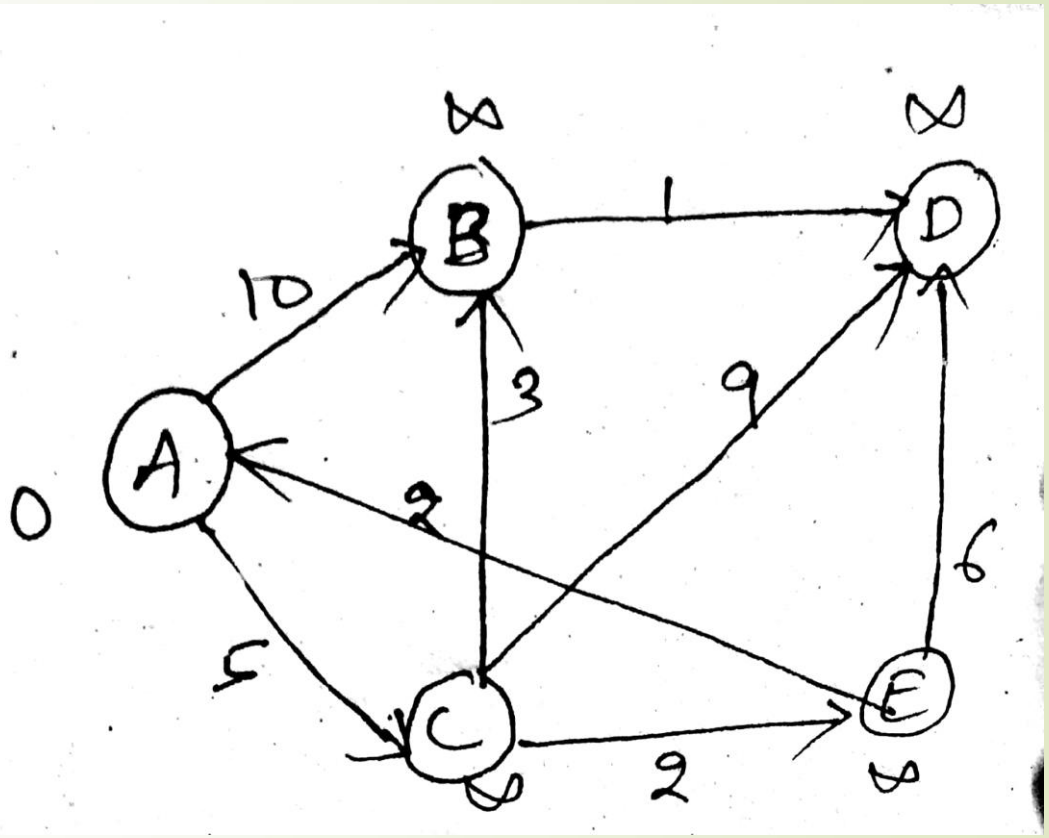
	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8		14	[7]



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8		14	[7]
B		[8]		13	



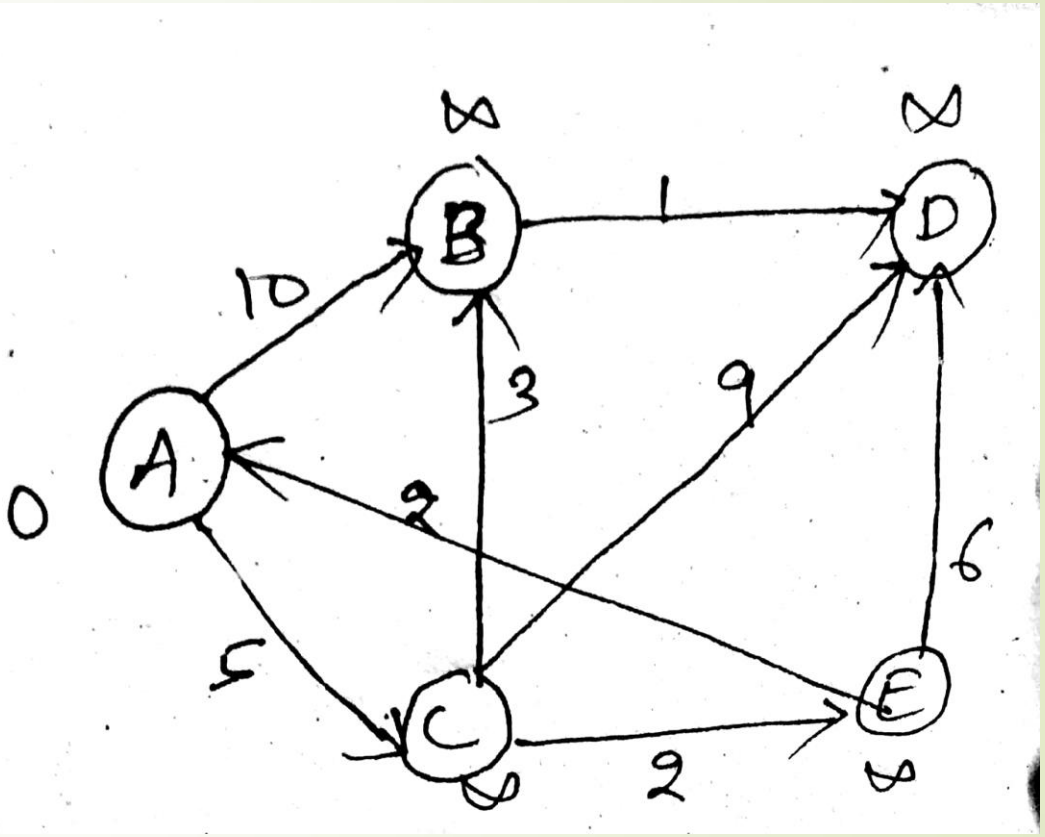
	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8		14	[7]
B		[8]		13	
D				[9]	



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8		14	[7]
B		[8]		13	
D				[9] ←	

path from A-D

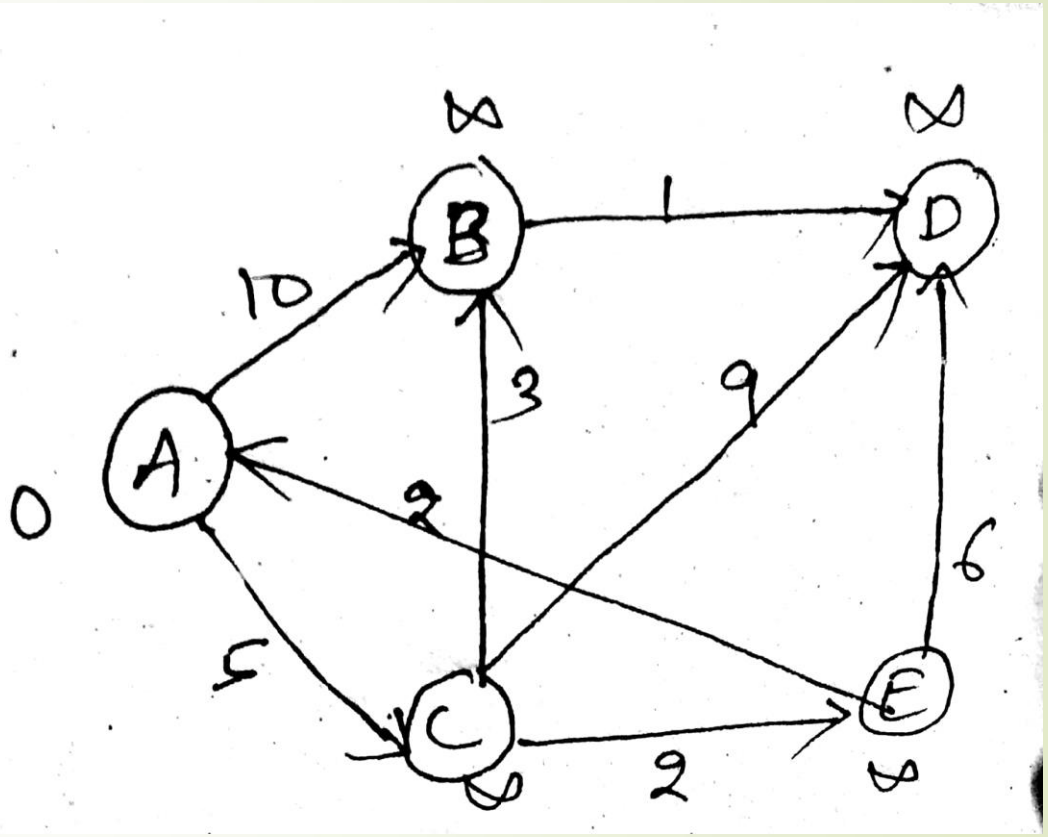
path :- D



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8		14	[7]
B		[8]		13	←
D				[9]	←

path from A-D

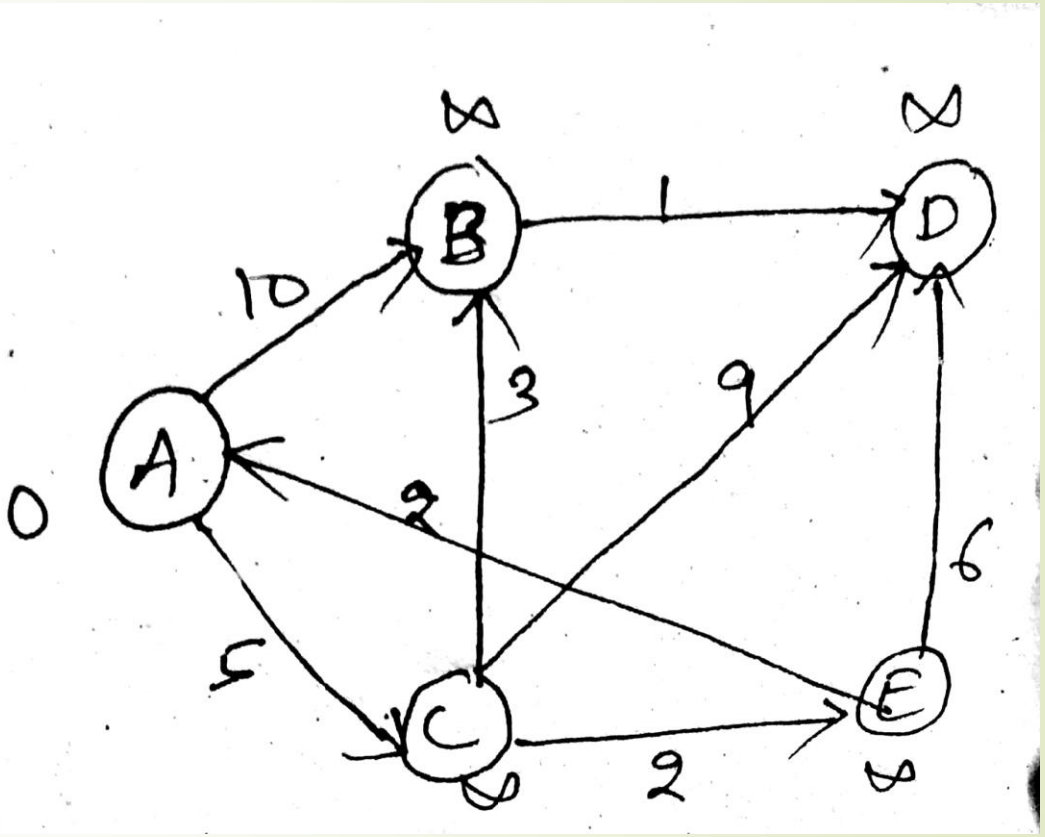
path :- D



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8		14	[7]
B		[8] ←		13 ←	
D				[9] ←	

Path From A-D

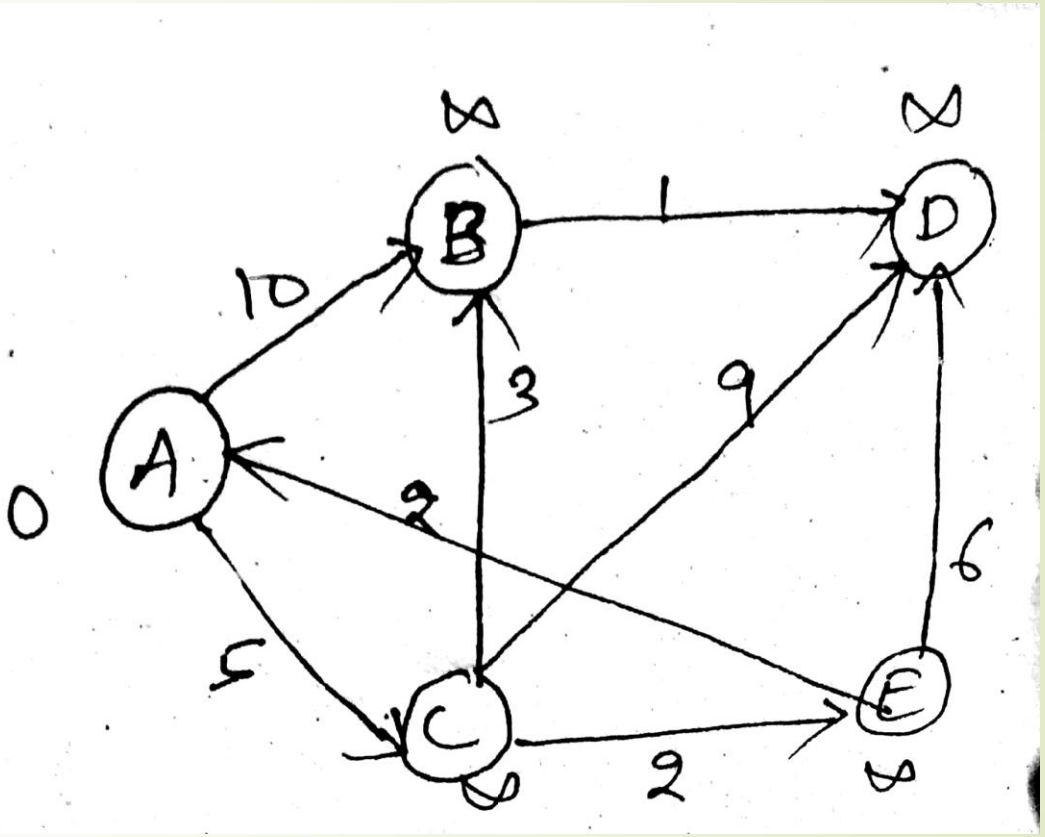
path :- DB



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10	[5]	∞	∞
E		8 ←		14	[7]
B		[8] ←		13 ←	
D				[9] ←	

path from A-D

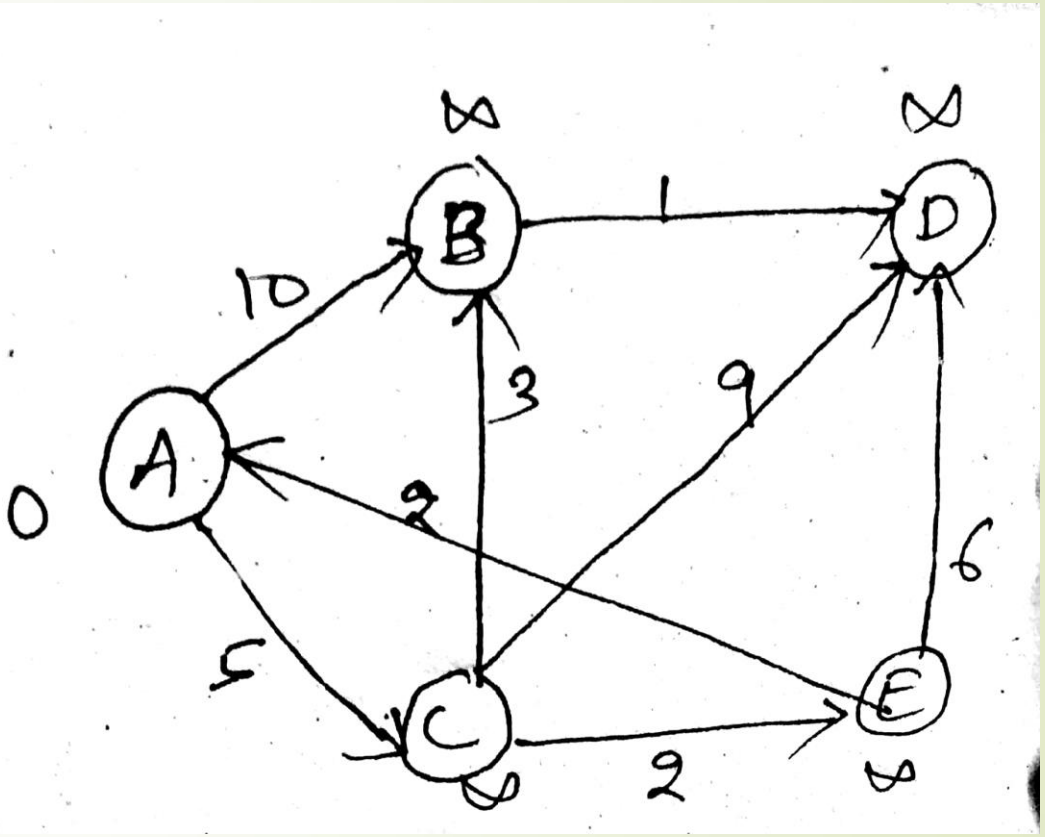
path :- DB



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10 ←	[5]	∞	∞
E		8 ←		14	[7]
B		[8] ←		13 ←	
D				[9] ←	

path From A-D

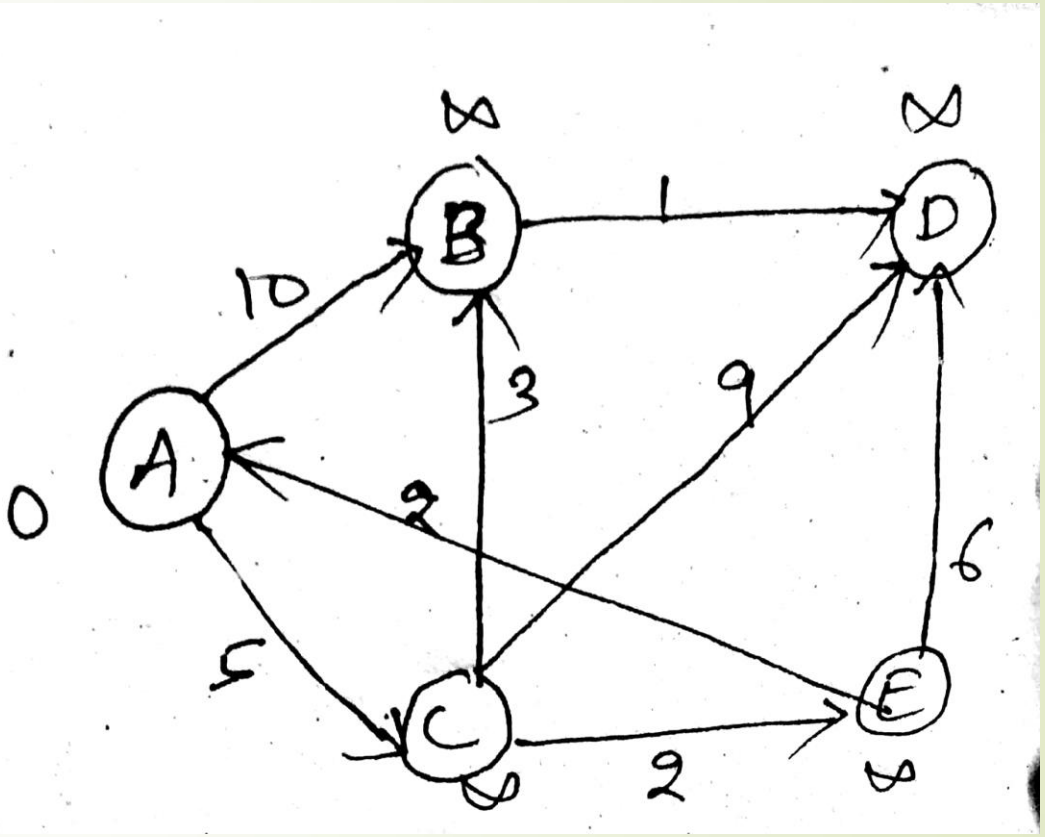
path :- DB



	A	B	C	D	E
A	[0]	∞	∞	∞	∞
C		10 ←	[5] ←	∞	∞
E		8 ←		14	[7]
B		[8] ←		13 ←	
D				[9] ←	

path From A-D

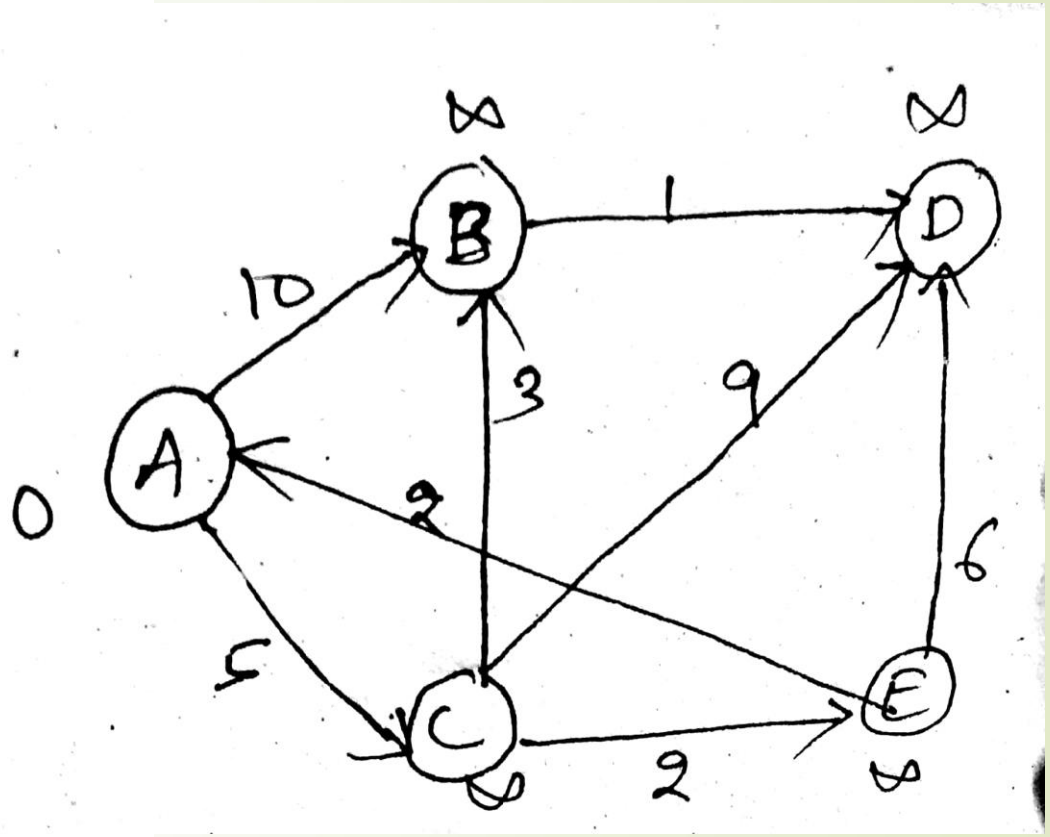
path :- DBC



	A	B	C	D	E
A	0	∞	∞ ←	∞	∞
C		10 ←	5 ←	∞	∞
E		8 ←		14	7
B		8 ←		13 ←	
D				9 ←	

Path from A-D

path :- DBCA

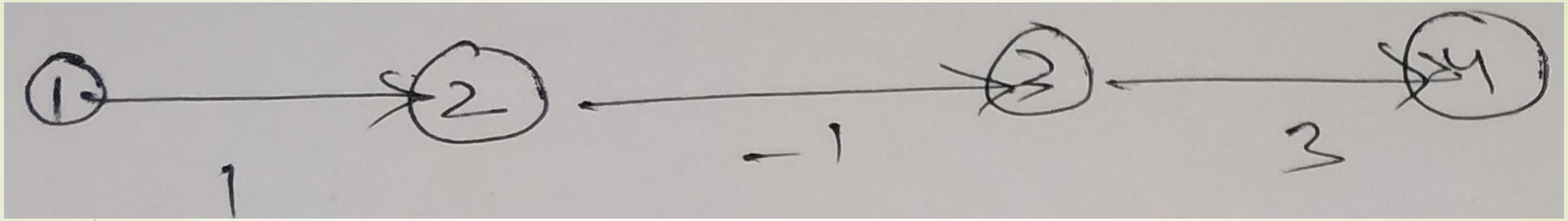


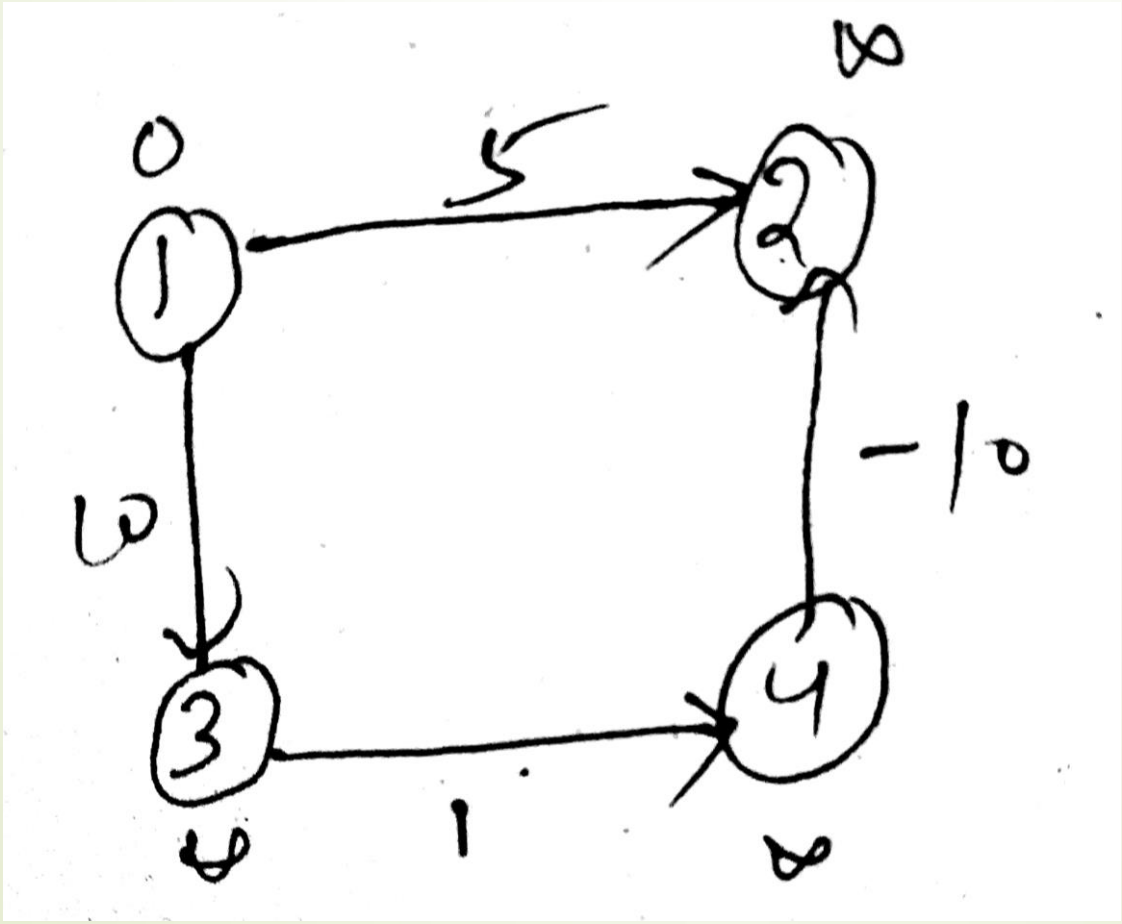
Algorithms - Single Source Shortest Path:

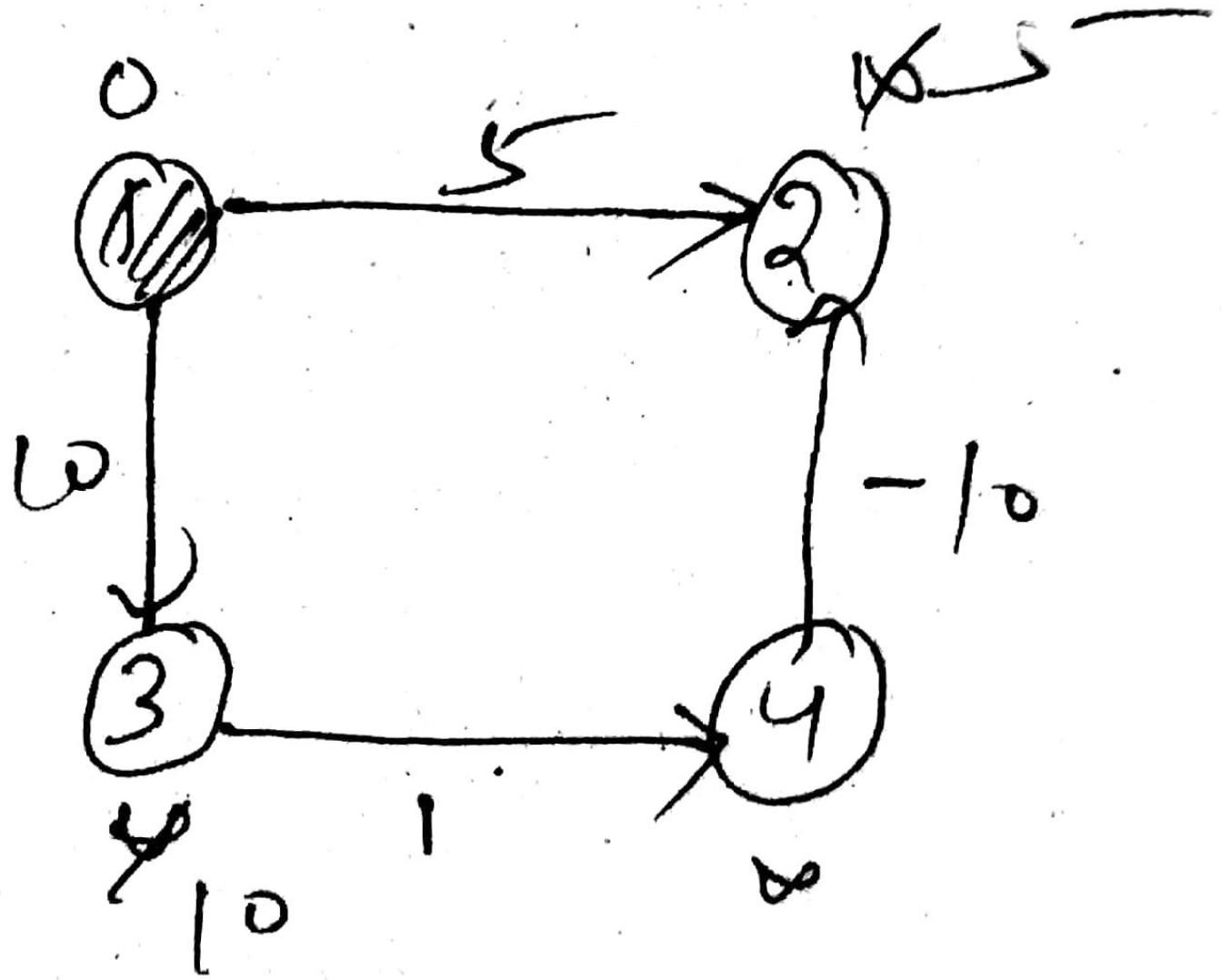
Dijkstra's Algorithm

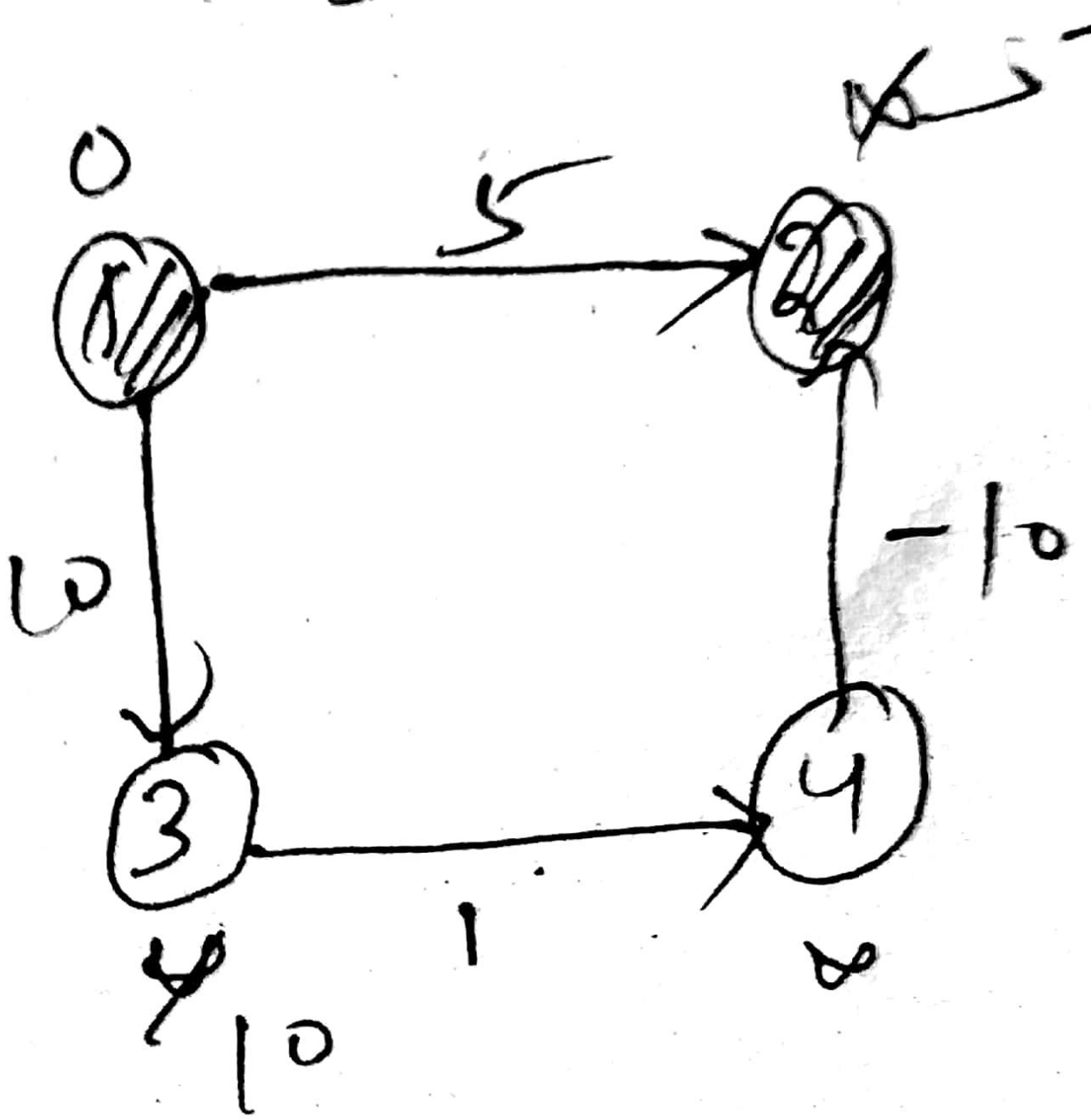
```
Input:
  a weighted digraph G=(V,E) with positive edge weights
  a source node s ∈ V
Initialization:
  d[s]=0
  for each vertex x ∈ V-s
        d[x]=infinity
  Mark all the vertices as unprocessed
Iteration:
  for i=1 to |V|
        Choose an unprocessed vertex x from V with minimum d[x]
        Mark x as processed
        for all y ∈ adj(x)
              if d[y] > d[x]+w(x,y)
                    d[y] = d[x]+w(x,y)
```

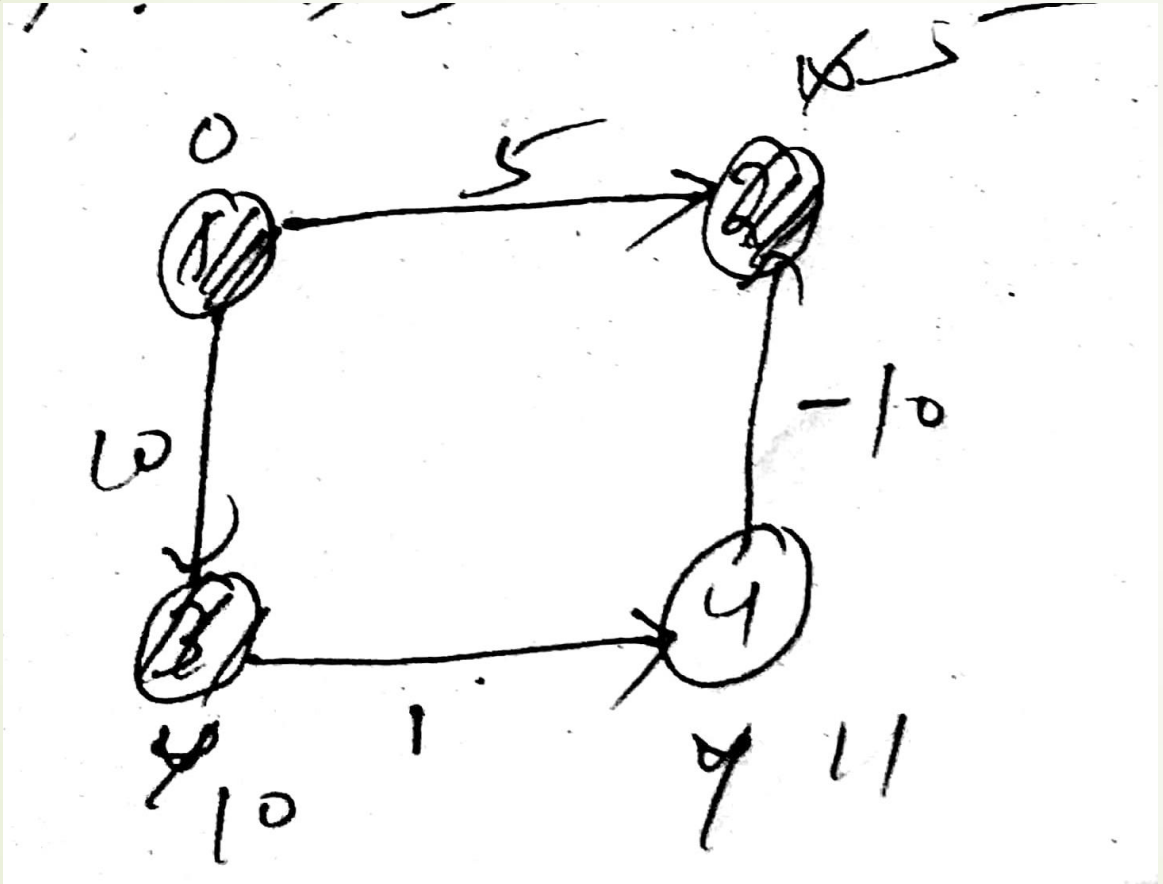
Negative Weights

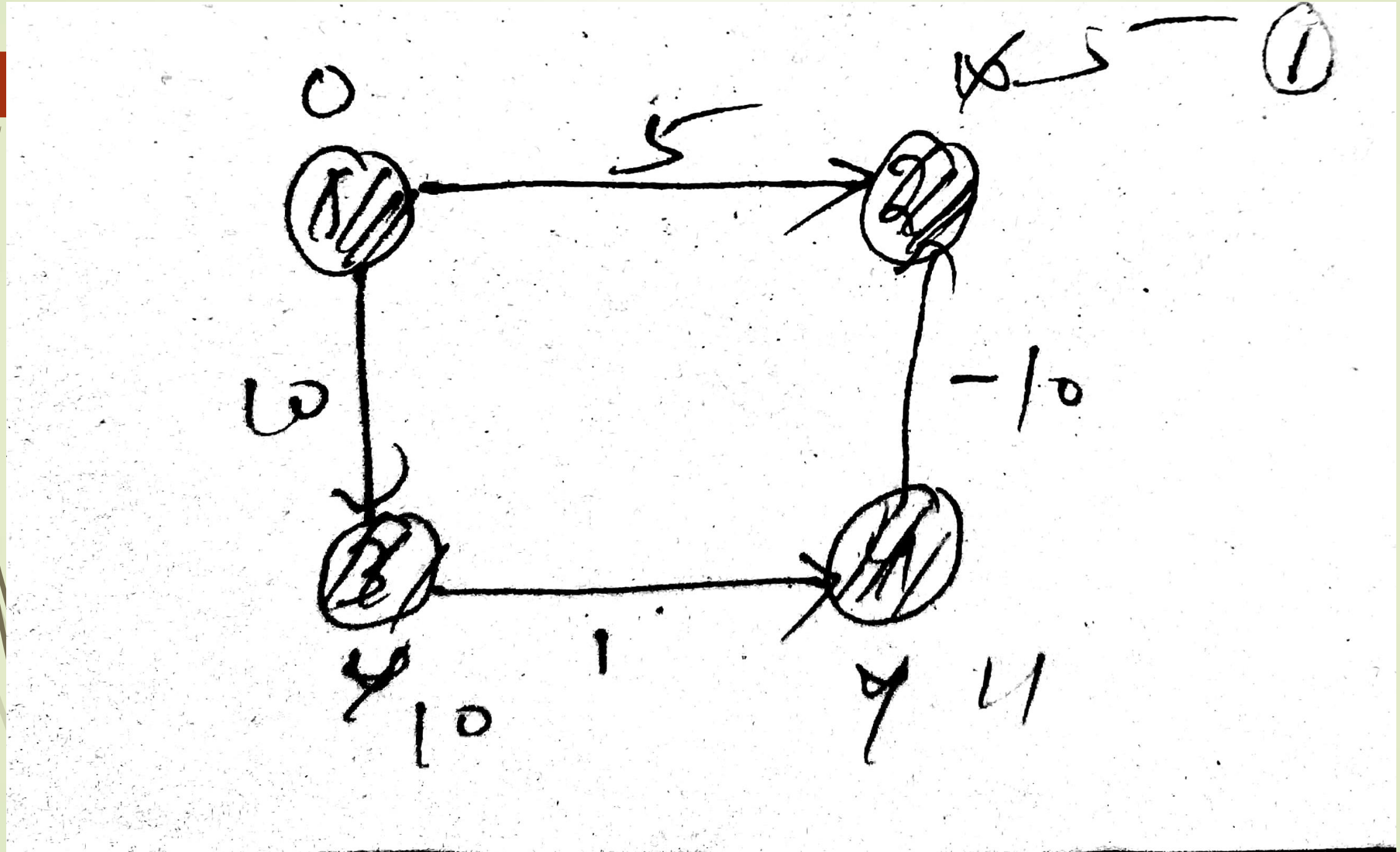





0
1
2
3
4
5
10
1
8
10

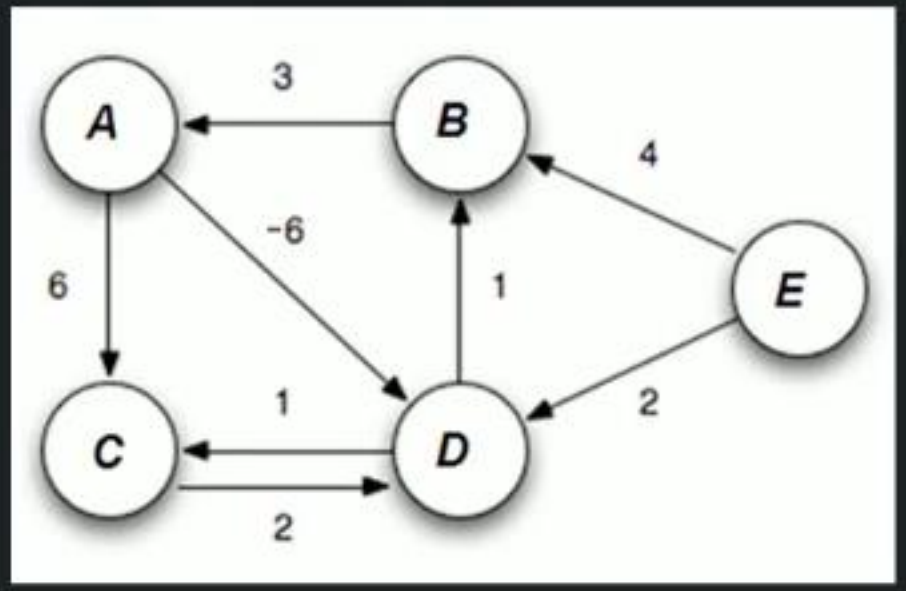

0
5
-10
3
1
10

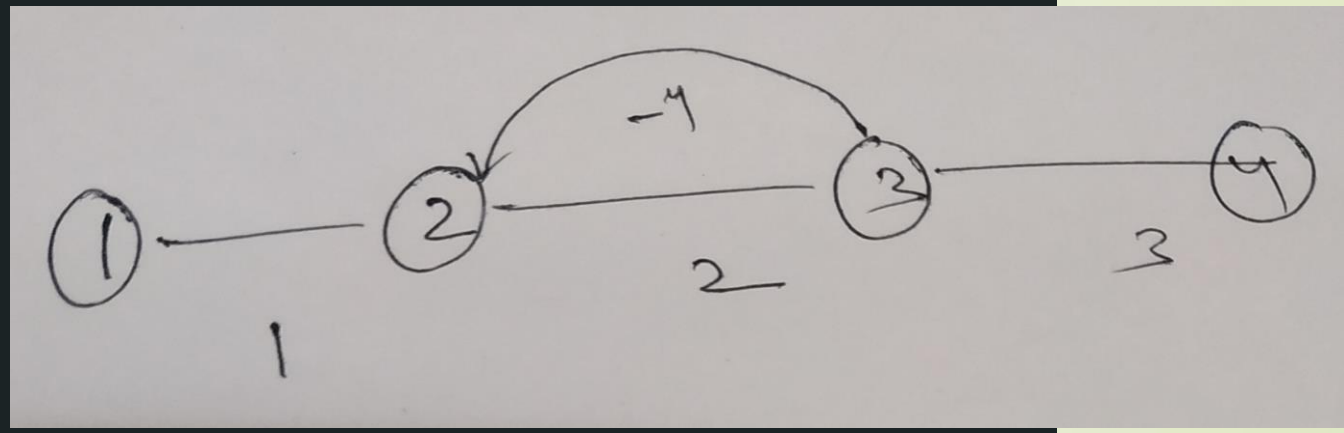

0
5
-10
3
1
10
7
11



Negative Cycle

- ✓ *A path is called "Negative Cycle" if :*
  - ✓ *There is a cycle (A cycle is a path of edges and vertices wherein a vertex is reachable from itself).*
  - ✓ *Total weight of cycle should be a negative number*






Thank
you