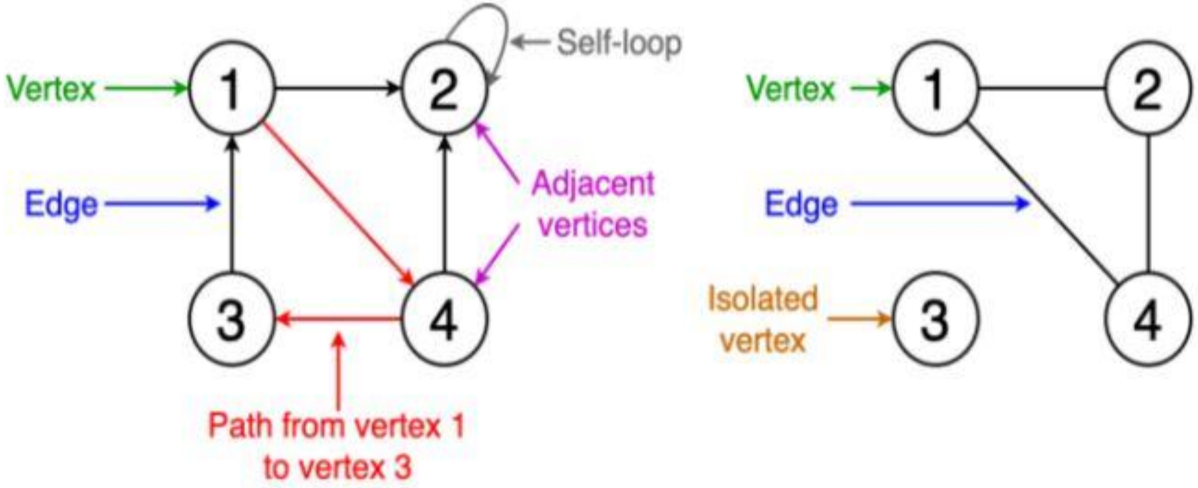
Group 2

Bhavya Gandhi, Anushka Dutta, Aditi Chansarkar, Hritika Doshi, Nishant Deo

WHAT IS GRAPH

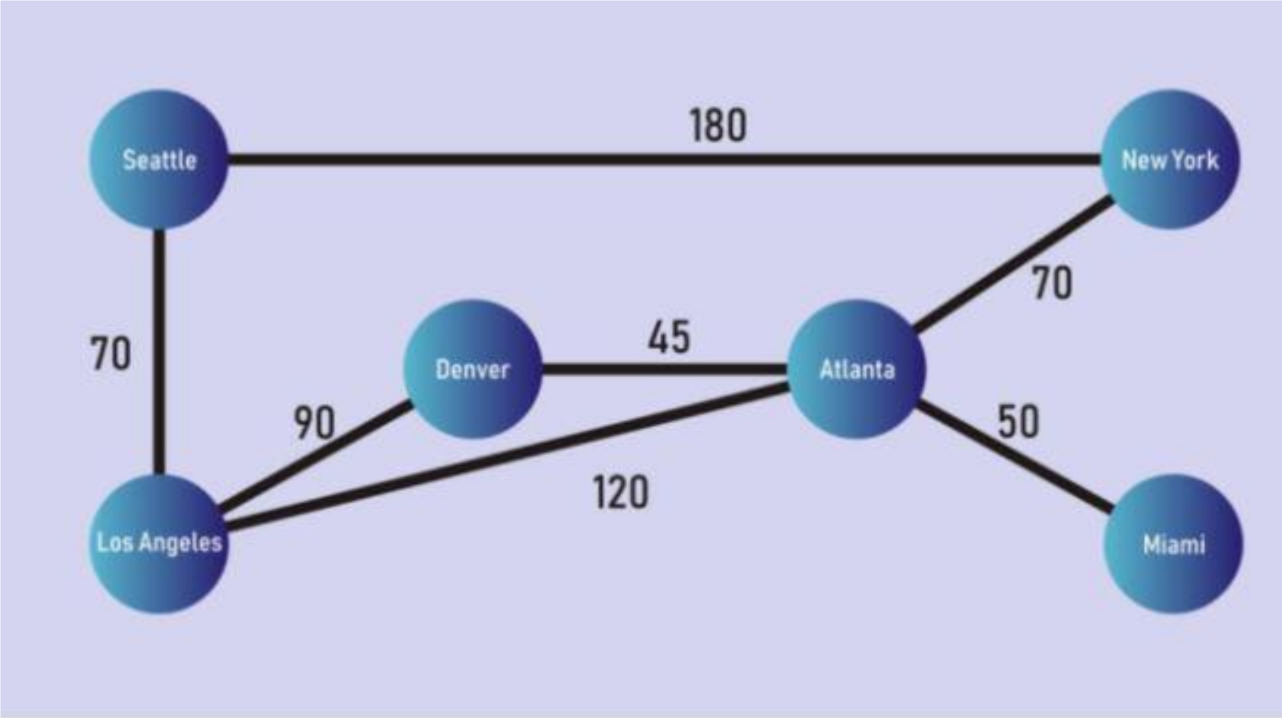
A graph consists of a finite set of vertices and nodes and a set of vertices connecting these vertices. Two vertices are said to be adjacent if they are connected to each other by the same edge.



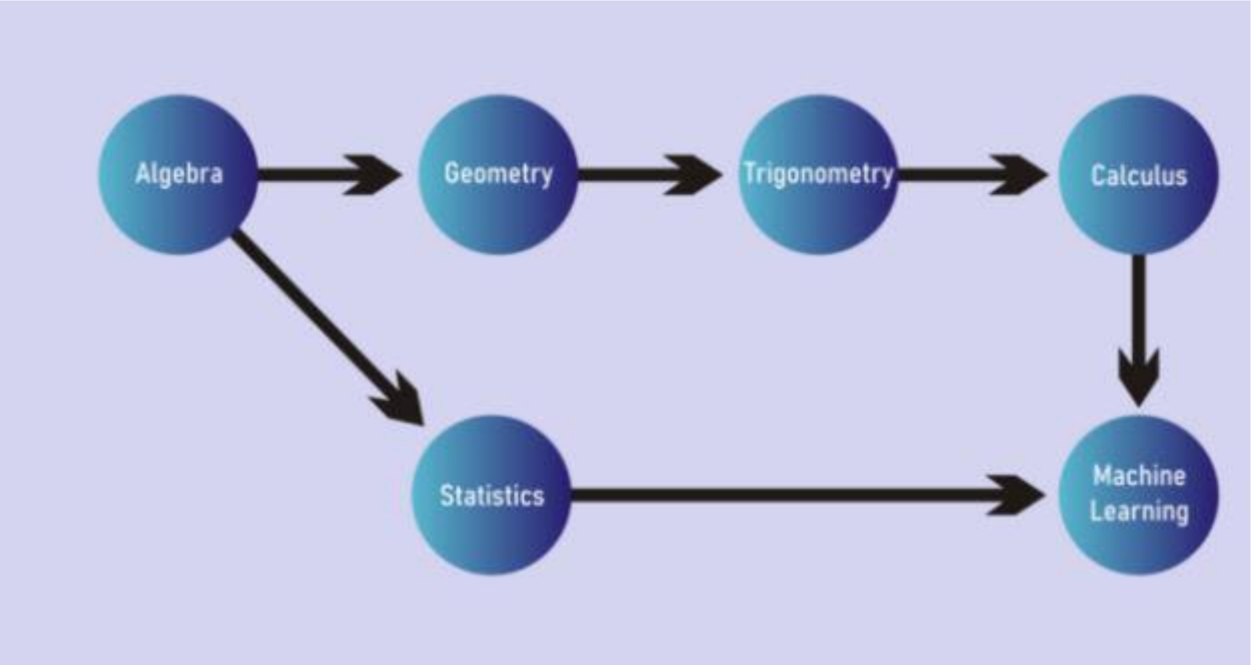
Terminology in Graphs

- **Order:** The number of vertices in the graph
- **Size:** The number of edges in the graph
- **Vertex degree:** The number of edges that are incident to a vertex
- **Isolated vertex:** A vertex that is not connected to any other vertices in the graph
- **Self-loop**: An edge from a vertex to itself
- **Directed graph:** A graph where all the edges have a direction indicating what is the start vertex and what is the end vertex
- **Undirected graph:** A graph with edges that have no direction
- **Weighted graph:** Edges of the graph has weights
- **Unweighted graph:** Edges of the graph has no weights

Example of weighted Graph

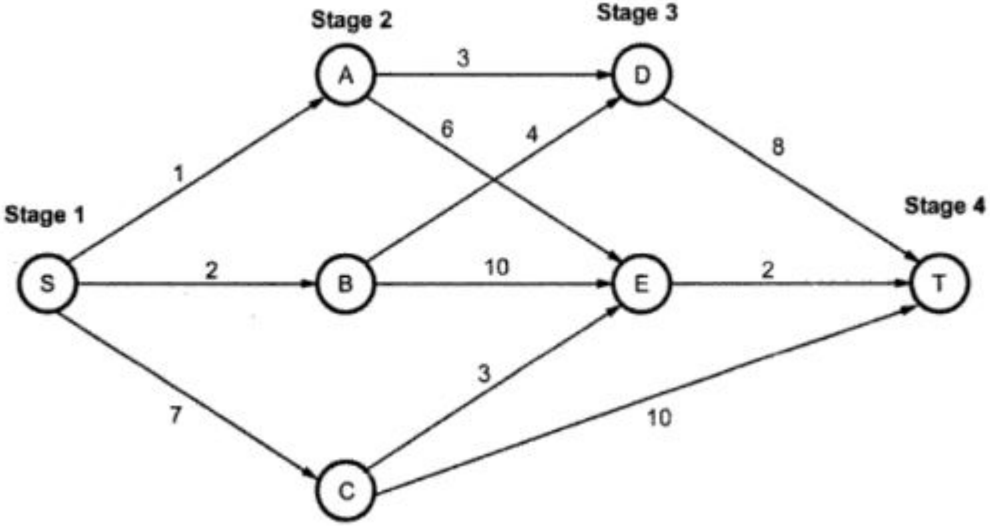


Example of unweighted Graph



WHAT IS A MULTISTAGE GRAPH ?

A Multistage graph is a directed graph in which the nodes can be divided into a set of stages such that all edges are from a stage to the next stage only.



Multistage graph

Forward approach or Dynamic programming approach

- Dynamic programming approach is similar to divide and conquer in breaking down the problem into smaller and yet smaller possible sub-problems.
- Dynamic Programming is an algorithm design method that can be used when the solution to a problem may be viewed as the result of a sequence of decisions
- Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimisation

Hence,

- The problem should be able to be divided into smaller overlapping sub-problem.
- An optimum solution can be achieved by using an optimum solution of smaller sub-problems.
- Dynamic algorithms use Memoization.

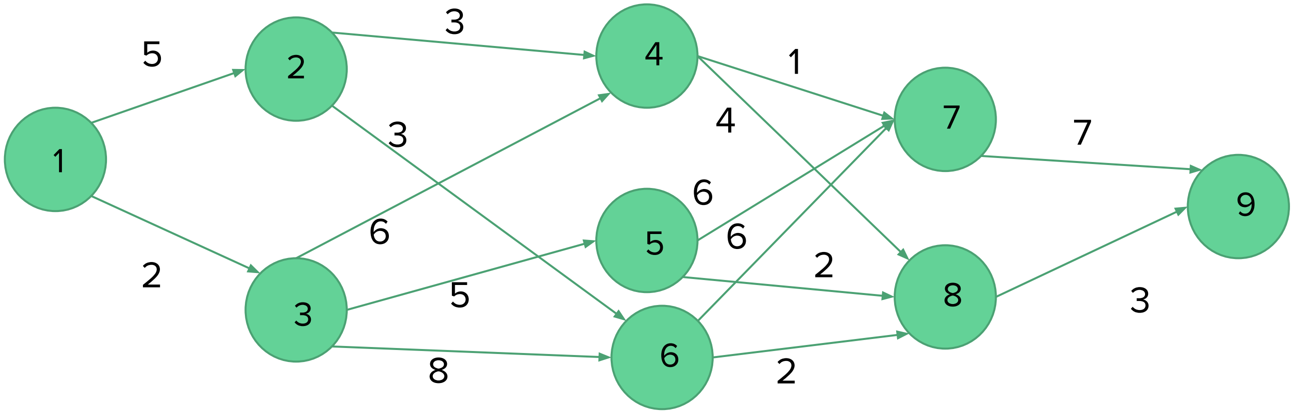
Forward approach or Dynamic programming approach

- In multi stage graph we divide the problem into no. of stages or multiple stages then we try to solve whole problem.
- In forward approach we will find the path from destination to source, and in backward approach we find the path from source to destination.
- The recurrence relations are formulated using the forward approach then the relations are solved backwards . i.e., beginning with the last decision
- To solve a problem by using dynamic programming:

1. Find out the recurrence relations.
2. Represent the problem by a multistage graph.

Example 1

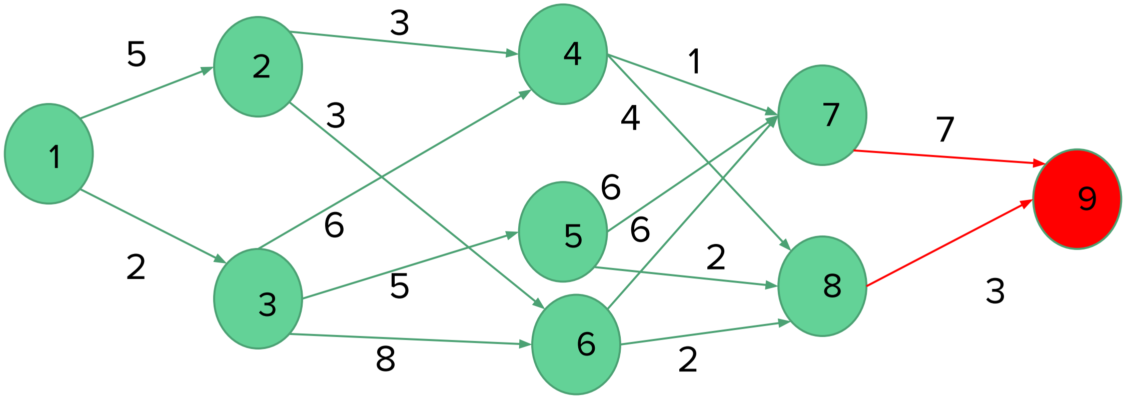
Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



Vertex	1	2	3	4	5	6	7	8	9
Cost									0
Dest.									9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

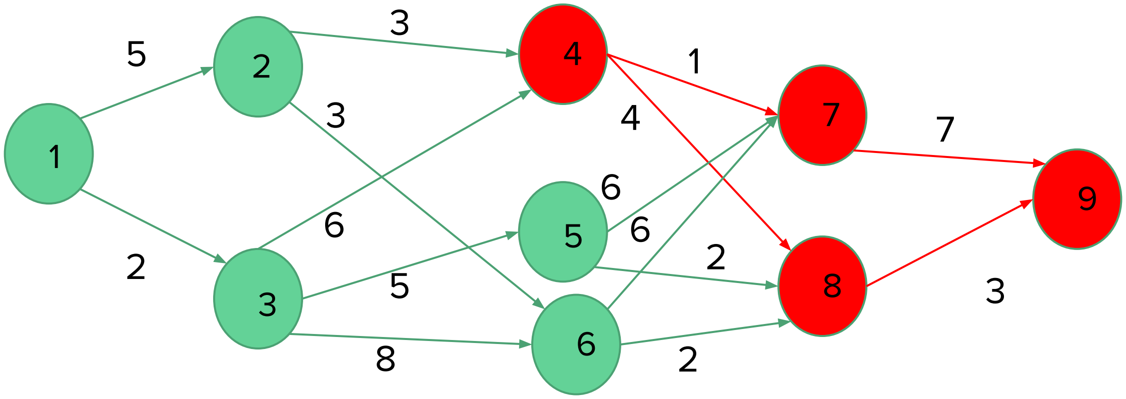


- Cost[4,7]= min{C (7,9)+ cost [5,9]= 7+0=7
- Cost[4,8]= min{C(8,9)+cost[5,9]=3+0=3

Vertex	1	2	3	4	5	6	7	8	9
Cost							7	3	0
Dest.							9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

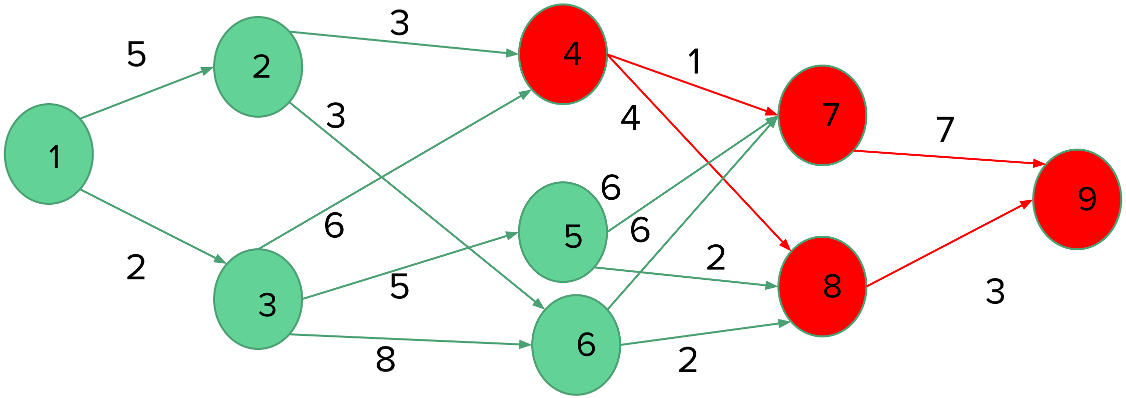


- Cost[3,4]= min{C (4,7)+ cost [4,7], C (4,8)+cost[4,8]}
  1+7=8
  4+3=7

Vertex	1	2	3	4	5	6	7	8	9
Cost							7	3	0
Dest.							9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

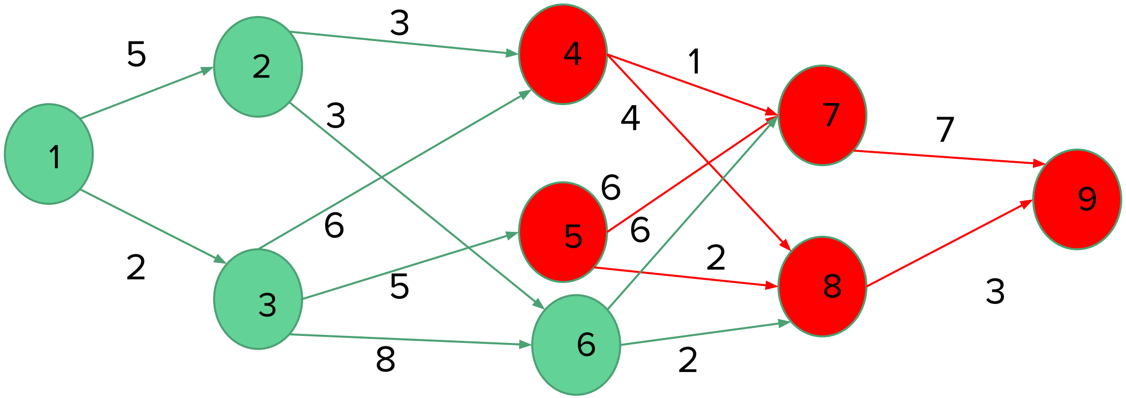


- Cost[3,4]= min{C (4,7)+ cost [4,7], C (4,8)+cost[4,8]}
  1+7=8
  4+3=7

Vertex	1	2	3	4	5	6	7	8	9
Cost				7			7	3	0
Dest.				8			9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

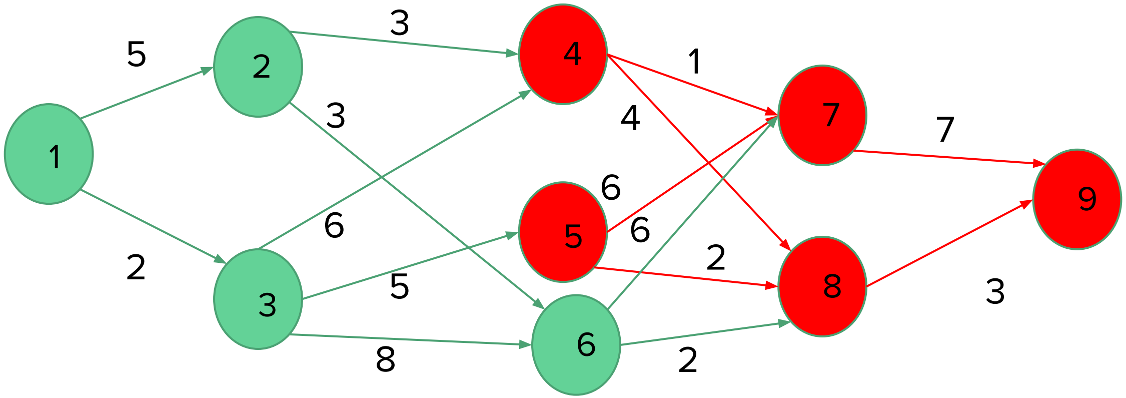


- Cost[3,5]= min{C (5,7)+ cost [4,7], C (5,8)+cost[4,8]}
  6+7=8
  2+3=5

Vertex	1	2	3	4	5	6	7	8	9
Cost				7			7	3	0
Dest.				8			9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

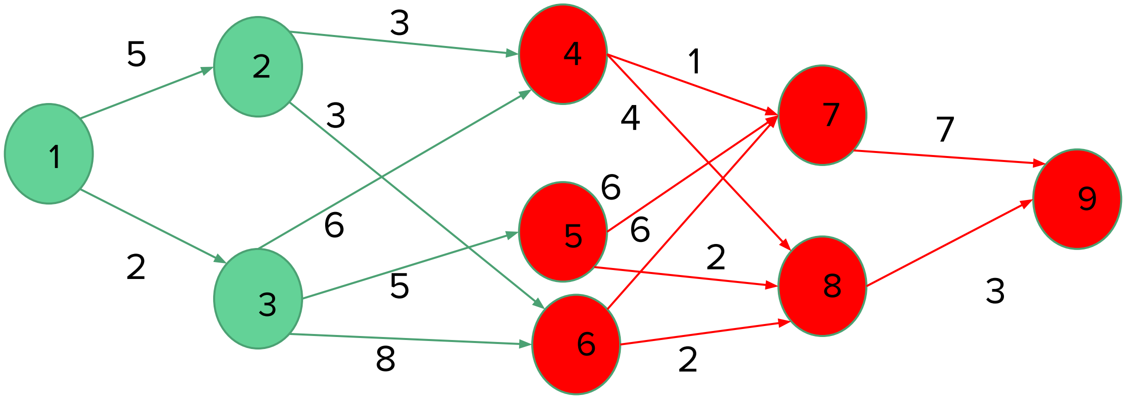


- Cost[3,5]= min{C (5,7)+ cost [4,7], C (5,8)+cost[4,8]}
  6+7=8
  2+3=5

Vertex	1	2	3	4	5	6	7	8	9
Cost				7	5		7	3	0
Dest.				8	8		9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

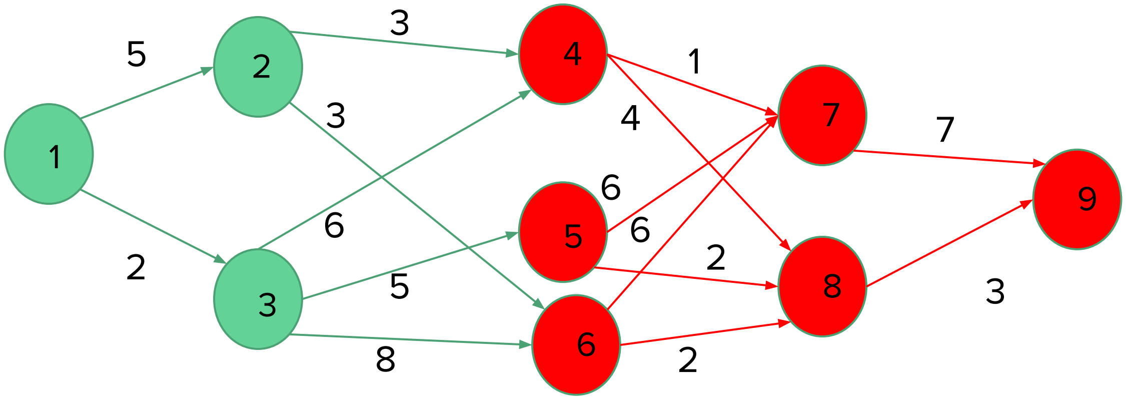


- Cost[3,6]= min{C (6,7)+ cost [4,7], C (6,8)+cost[4,8]}
  6+7=8
  2+3=5

Vertex	1	2	3	4	5	6	7	8	9
Cost				7	5		7	3	0
Dest.				8	8		9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

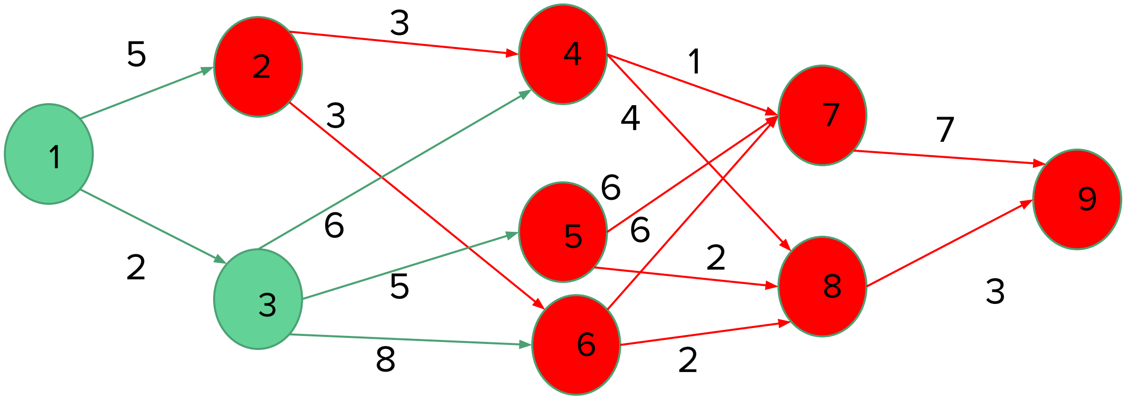


- Cost[3,6]= min{C (6,7)+ cost [4,7], C (6,8)+cost[4,8]}
  6+7=8
  2+3=5

Vertex	1	2	3	4	5	6	7	8	9
Cost				7	5	5	7	3	0
Dest.				8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

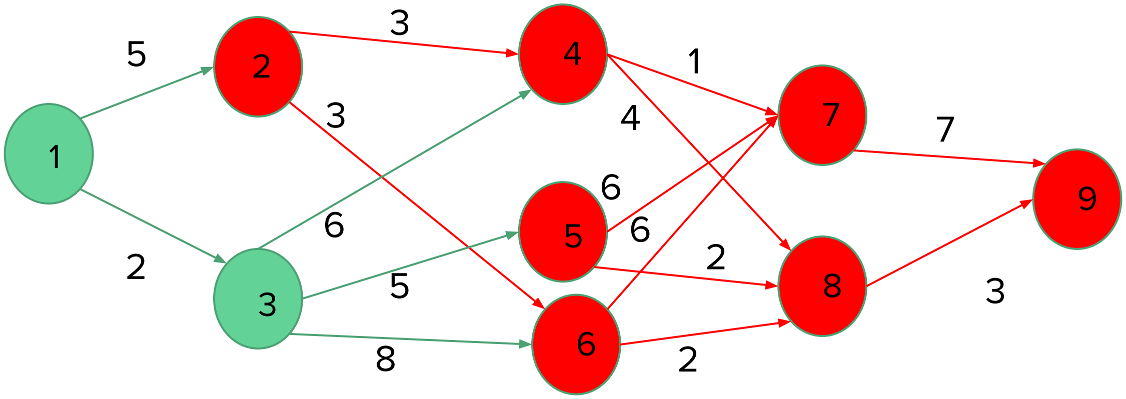


- Cost[2,2]= min{C (2,4)+ cost [3,4], C (2,6)+cost[3,6]}
  3+7=10
  3+5=8

Vertex	1	2	3	4	5	6	7	8	9
Cost				7	5	5	7	3	0
Dest.				8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

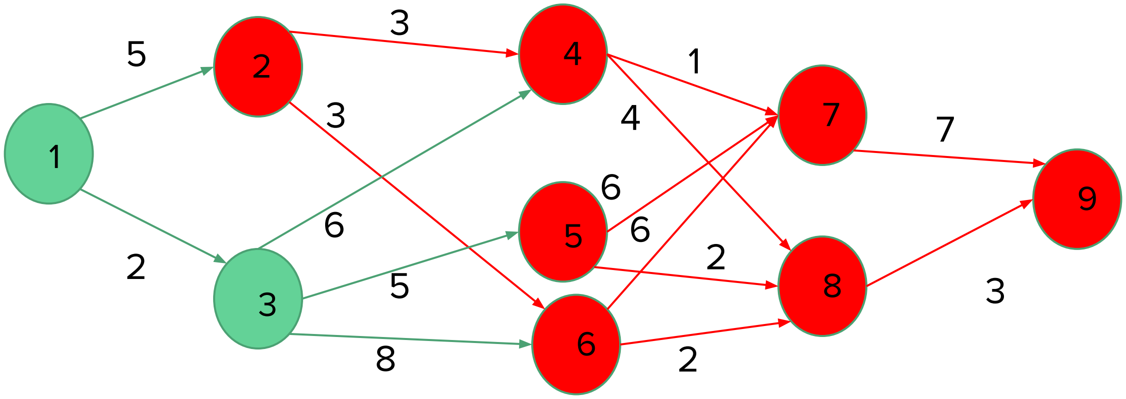


- Cost[2,2]= min{C (2,4)+ cost [3,4], C (2,6)+cost[3,6]}
  3+7=10
  3+5=8

Vertex	1	2	3	4	5	6	7	8	9
Cost		8		7	5	5	7	3	0
Dest.		6		8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

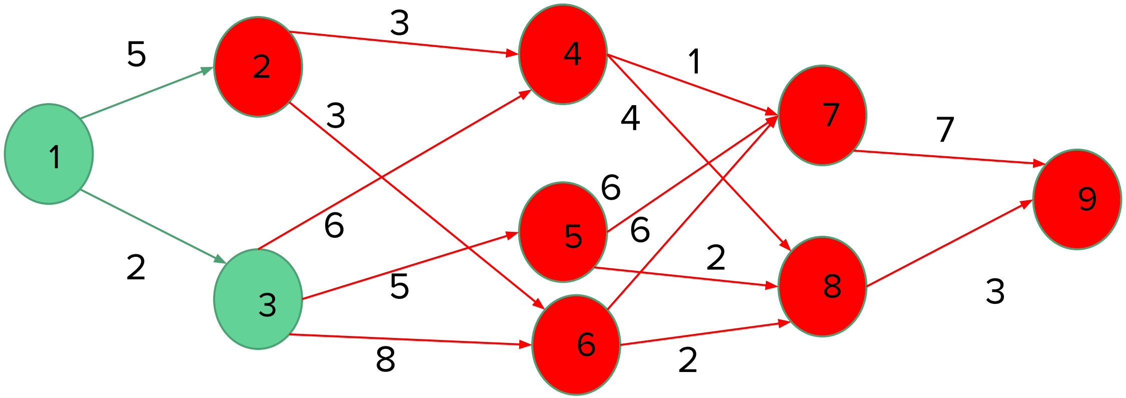


- Cost[2,2]= min{C (2,4)+ cost [3,4], C (2,6)+cost[3,6]}
  3+7=10
  3+5=8

Vertex	1	2	3	4	5	6	7	8	9
Cost		8		7	5	5	7	3	0
Dest.		6		8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

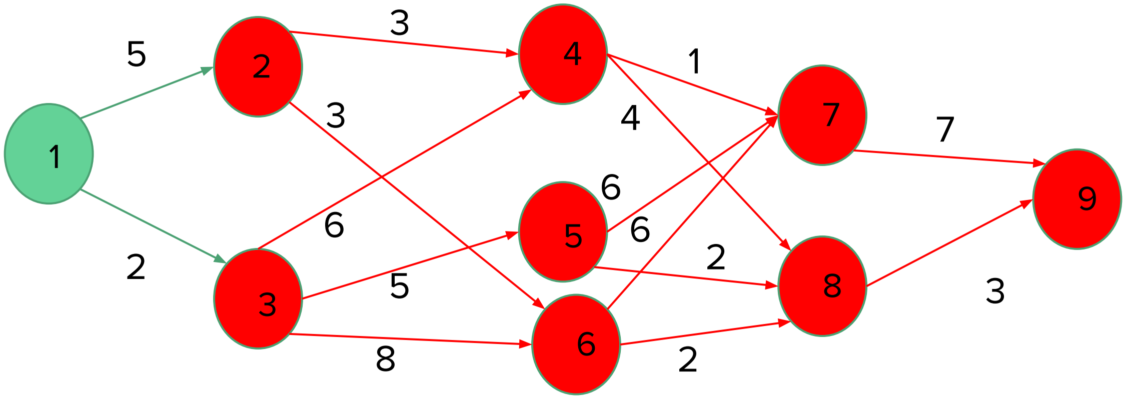


- Cost[2,3]= min{C (3,4)+ cost [3,4], C (3,6)+cost[3,6], C(3,5)+cost[3,5]}
  6+7=13
  5+5=10
  8+5=13

Vertex	1	2	3	4	5	6	7	8	9
Cost		8		7	5	5	7	3	0
Dest.		6		8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

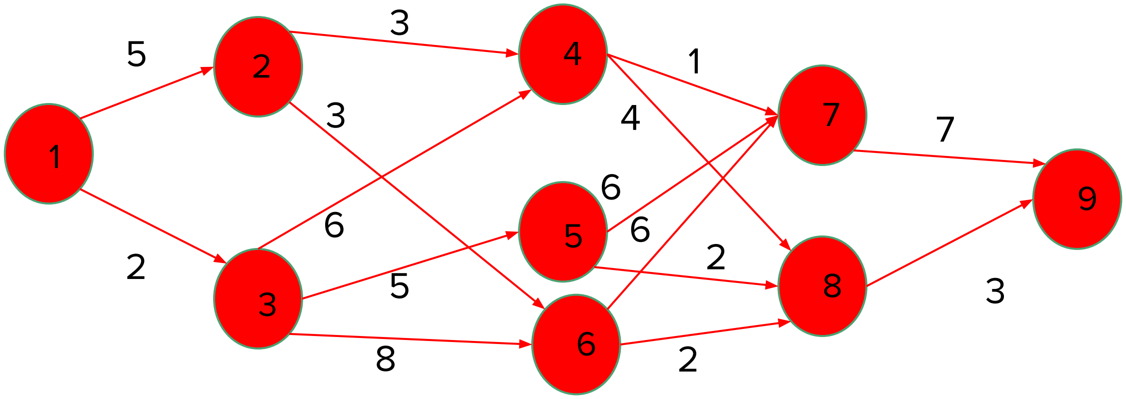


- Cost[2,3]= min{C (3,4)+ cost [3,4], C (3,6)+cost[3,6], C(3,5)+cost[3,5]}
  6+7=13
  5+5=10
  8+5=13

Vertex	1	2	3	4	5	6	7	8	9
Cost		8	10	7	5	5	7	3	0
Dest.		6	5	8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

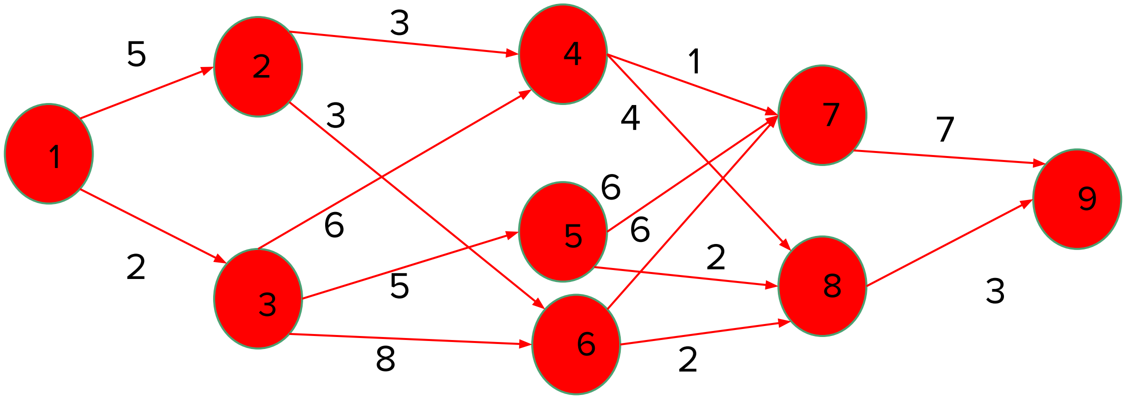


- Cost[1,1]= min{C (1,2)+ cost [2,2], C (2,3)+cost[2,3]}
  5+8=13
  2+10=12

Vertex	1	2	3	4	5	6	7	8	9
Cost		8	10	7	5	5	7	3	0
Dest.		6	5	8	8	8	9	9	9

Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

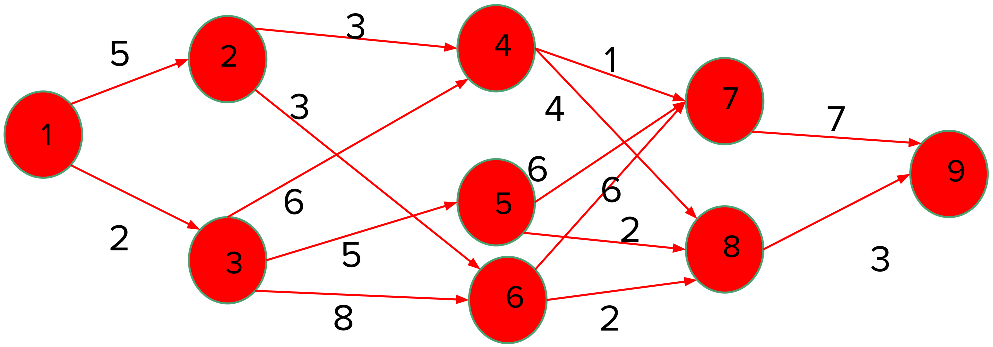


- Cost[1,1]= min{C (1,2)+ cost [2,2], C (2,3)+cost[2,3]}
  5+8=13
  2+10=12

Vertex	1	2	3	4	5	6	7	8	9
Cost	12	8	10	7	5	5	7	3	0
Dest.	3	6	5	8	8	8	9	9	9

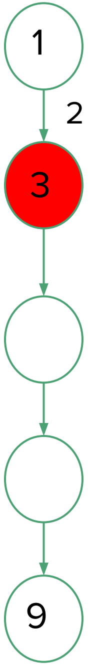
Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



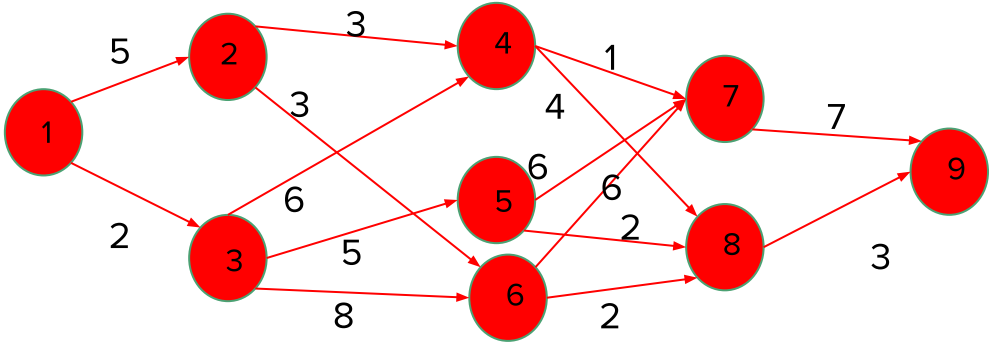
Vertex	1	2	3	4	5	6	7	8	9
Cost	12	8	10	7	5	5	7	3	0
Dest.	3	6	5	8	8	8	9	9	9

Tracking the shortest path:



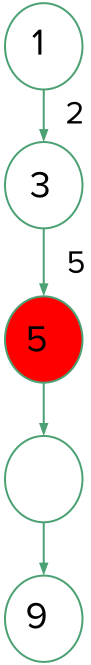
Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



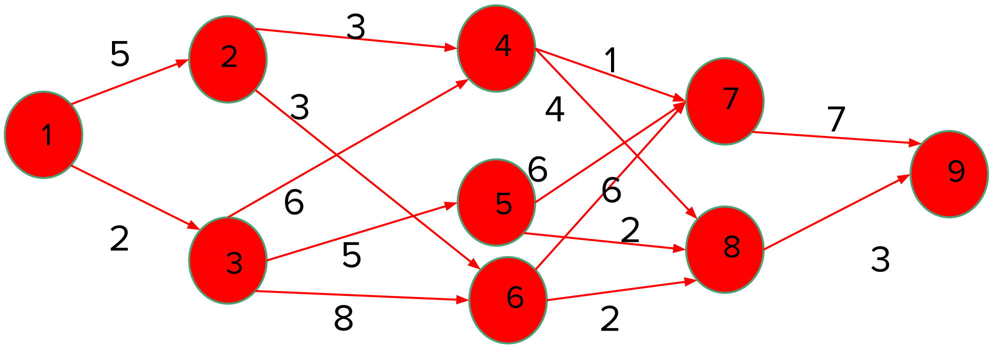
Vertex	1	2	3	4	5	6	7	8	9
Cost	12	8	10	7	5	5	7	3	0
Dest.	3	6	5	8	8	8	9	9	9

Tracking the shortest path:



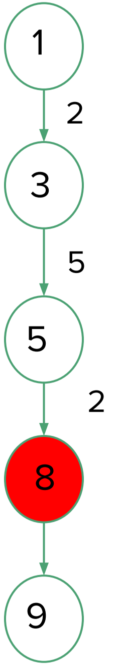
Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



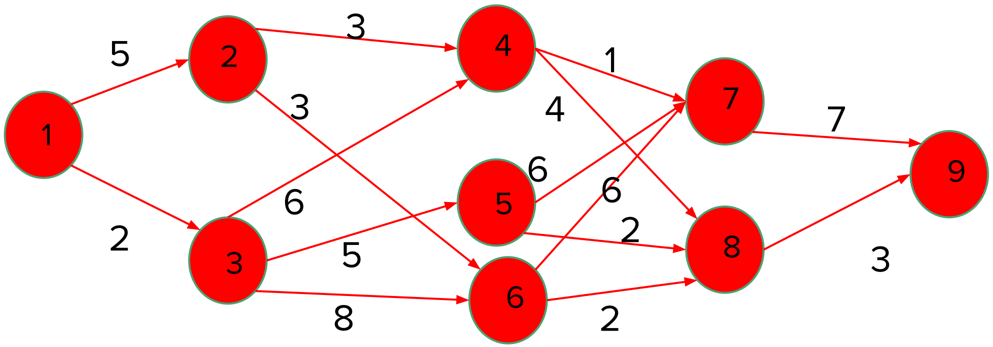
Vertex	1	2	3	4	5	6	7	8	9
Cost	12	8	10	7	5	5	7	3	0
Dest.	3	6	5	8	8	8	9	9	9

Tracking the shortest path:



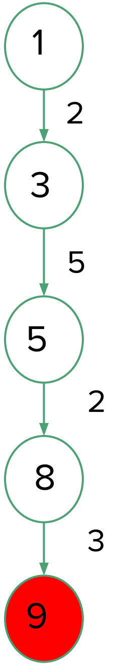
Example 1

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



Vertex	1	2	3	4	5	6	7	8	9
Cost	12	8	10	7	5	5	7	3	0
Dest.	3	6	5	8	8	8	9	9	9

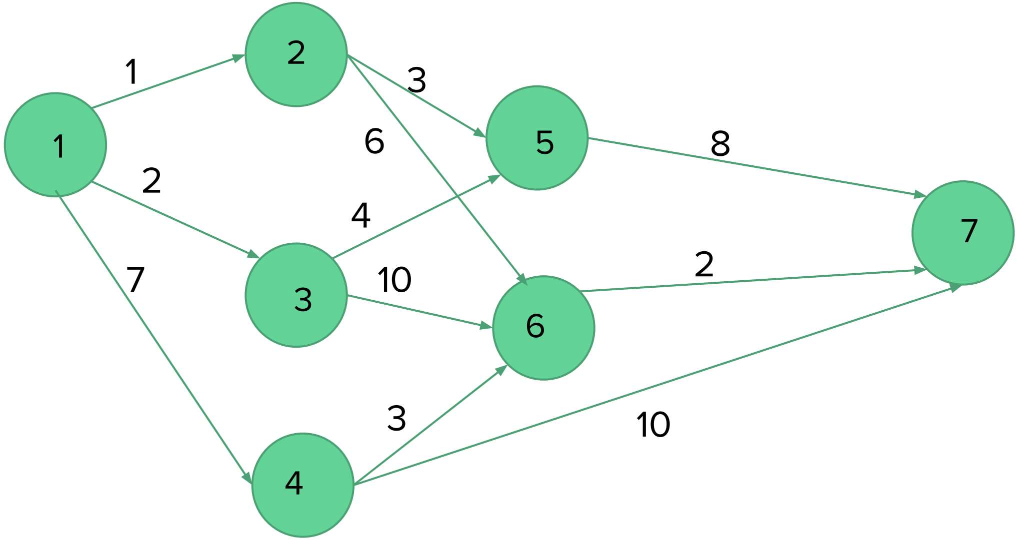
Tracking the shortest path:



=2+5+2+3=
12

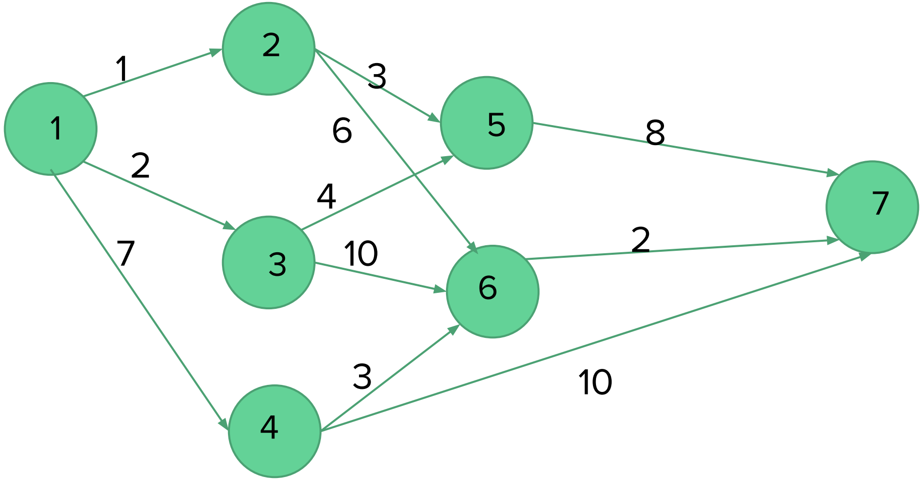
Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



Example 2

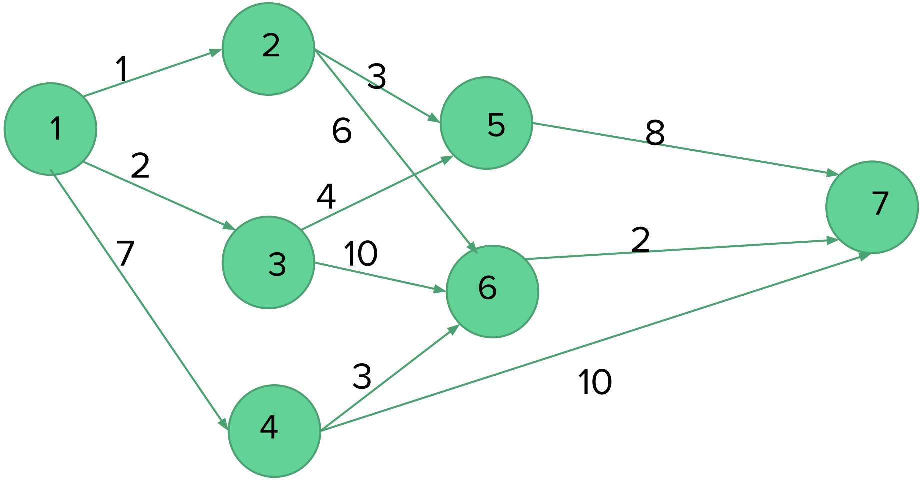
Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}



Vertex	1	2	3	4	5	6	7
Cost							0
Dest.							7

Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

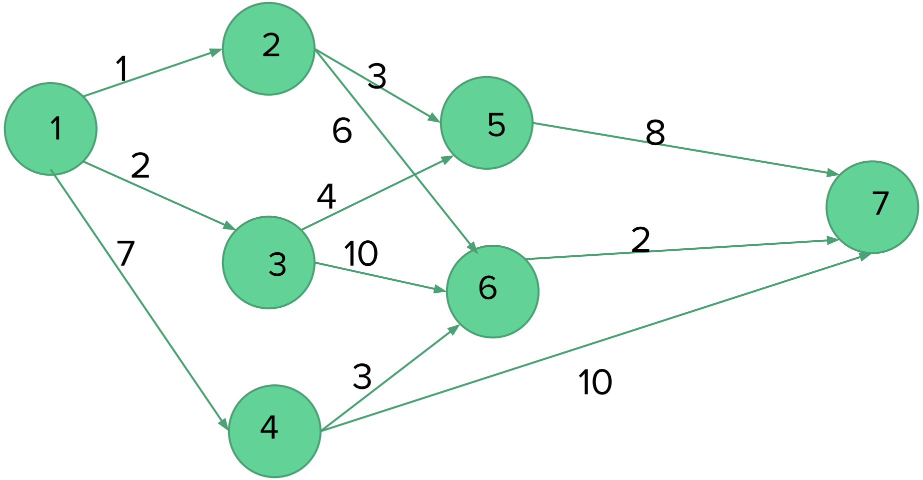


- Cost[3,5]= min{C (5,7)+ cost [4,7]= 8+0=8
- Cost[3,6]= min{C(6,4)+cost[4,7]=2+0=2

Vertex	1	2	3	4	5	6	7
Cost					8	2	0
Dest.					7	7	7

Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

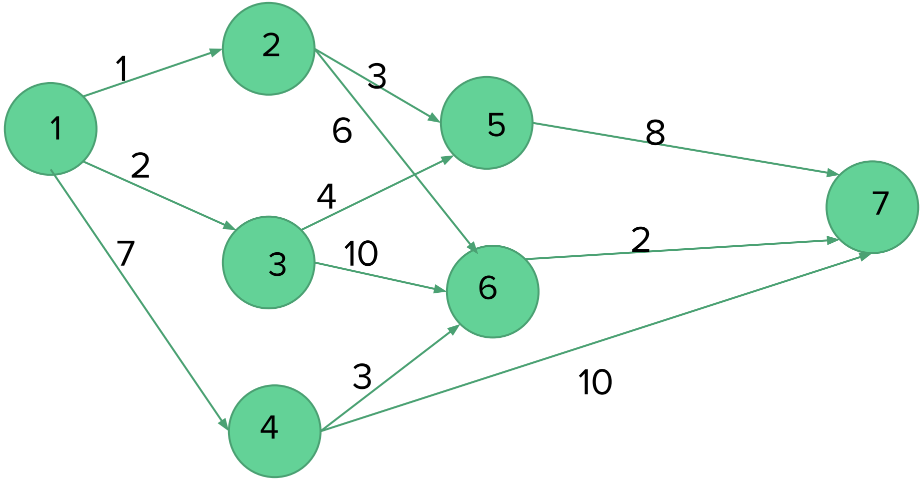


- Cost[2,2]= min{C (2,5)+ cost [3,5], C(2,6)+cost[3,6]
- Cost[2,3]= min{C(3,5)+cost[3,5], C(3,6)+cost[3,6]
- Cost[2,4]=min{C(4,6)+ cost[3,6], C(4,7)+ cost [4,7]}

Vertex	1	2	3	4	5	6	7
Cost					8	2	0
Dest.					7	7	7

Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

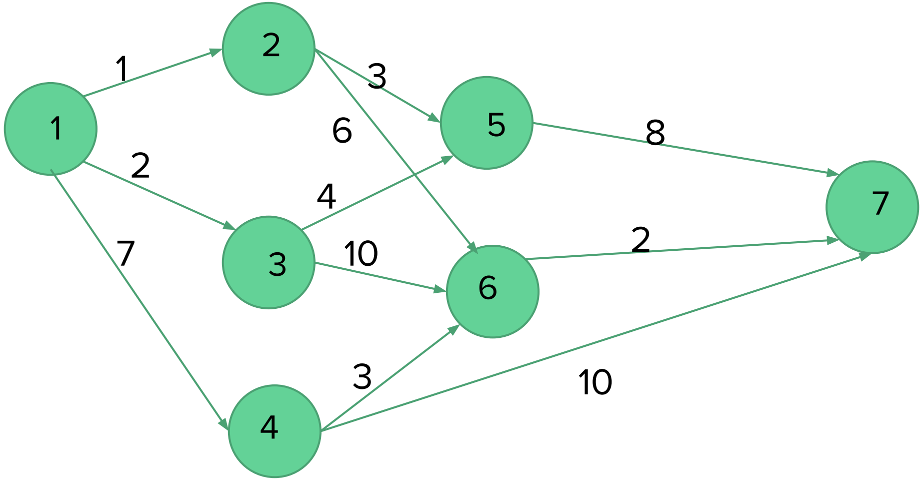


- Cost[2,2]= min{C (2,5)+ cost [3,5], C(2,6)+cost[3,6]
- Cost[2,3]= min{C(3,5)+cost[3,5], C(3,6)+cost[3,6]
- Cost[2,4]=min{C(4,6)+ cost[3,6], C(4,7)+ cost [4,7]}

Vertex	1	2	3	4	5	6	7
Cost		8	12	5	8	2	0
Dest.		6	5	6	7	7	7

Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

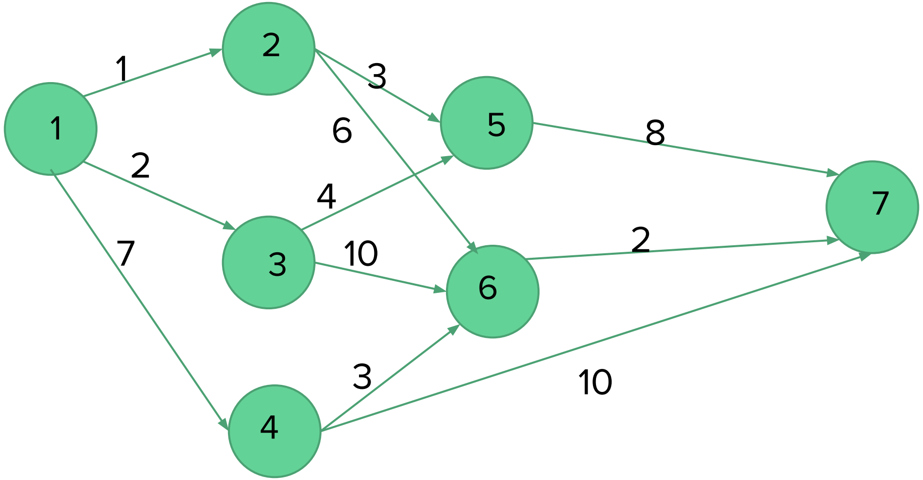


- Cost[1,1]= min{C (1,2)+ cost [2,2], C(1,3)+cost[2,3], C (1,4)+ cost [2,4]}

Vertex	1	2	3	4	5	6	7
Cost		8	12	5	8	2	0
Dest.		6	5	6	7	7	7

Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

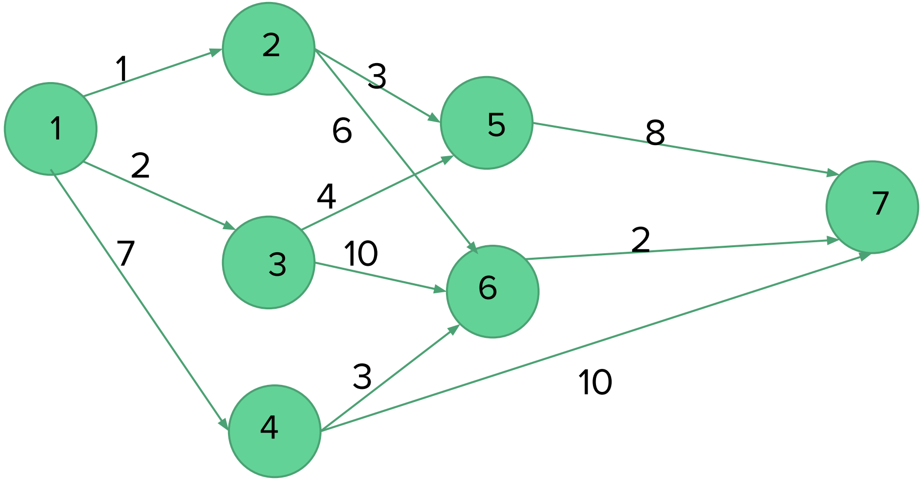


- Cost[1,1]= min{C (1,2)+ cost [2,2], C(1,3)+cost[2,3], C (1,4)+ cost [2,4]}

Vertex	1	2	3	4	5	6	7
Cost	9	8	12	5	8	2	0
Dest.	2	6	5	6	7	7	7

Example 2

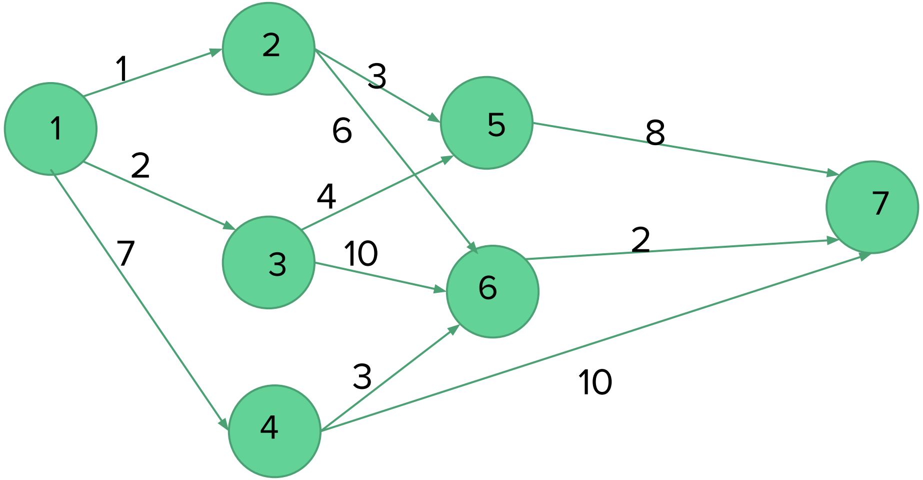
Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}

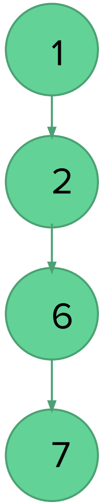


Vertex	1	2	3	4	5	6	7
Cost	9	8	12	5	8	2	0
Dest.	2	6	5	6	7	7	7

Example 2

Formula: cost[i,j]= min{ c[j,r]+ cost[i+1,r]}





Vertex	1	2	3	4	5	6	7
Cost	9	8	12	5	8	2	0
Dest.	2	6	5	6	7	7	7

1+6+2=9(Shortest path)