DATA STRUCTURES & ALGORITHMS

Shortest Path Problem

Instructor: Engr. Laraib Siddiqui

Shortest Path Problem

Shortest path problem is a problem of finding the shortest path(s) between vertices of a given graph. It is a path between two vertices is a path that has the least cost as compared to all other existing paths.

Shortest path algorithms are a family of algorithms used for solving the shortest path problem.

Applications

Shortest path algorithms have a wide range of applications such as in-

- Google Maps
- Road Networks

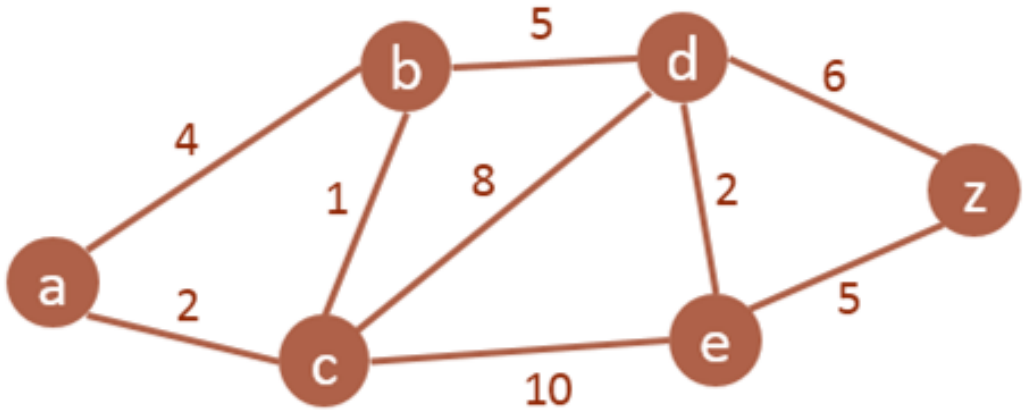
Dijkstra’s Algorithm

Dijkstra's algorithm has many variants but the most common one is to find the shortest paths from the source vertex to all other vertices in the graph.

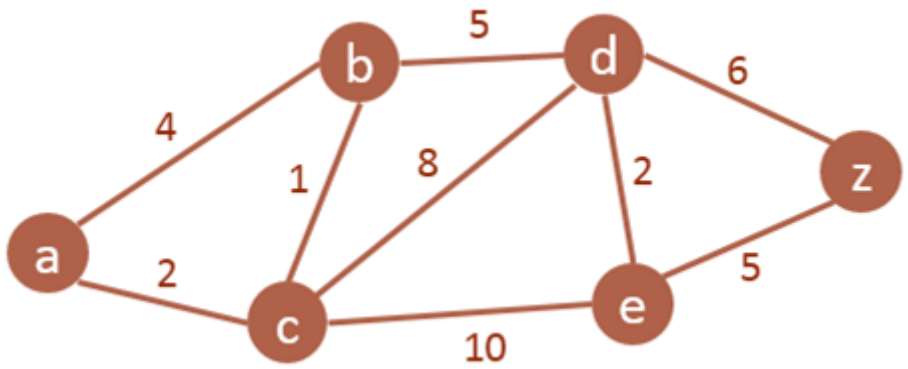
Algorithm

1. Mark your selected initial node with a current distance of 0 and the rest with infinity.
2. Set the non-visited node with the smallest current distance as the current node C.
3. For each neighbor N of your current node C: add the current distance of C with the weight of the edge connecting C-N. If it's smaller than the current distance of N, set it as the new current distance of N.
4. Mark the current node C as visited.
5. If there are non-visited nodes, go to step 2.

Dijkstra's Algorithm

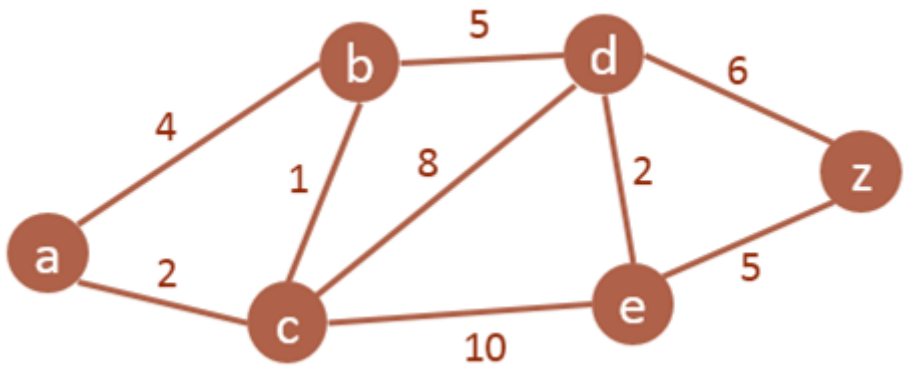


Dijkstra's Algorithm



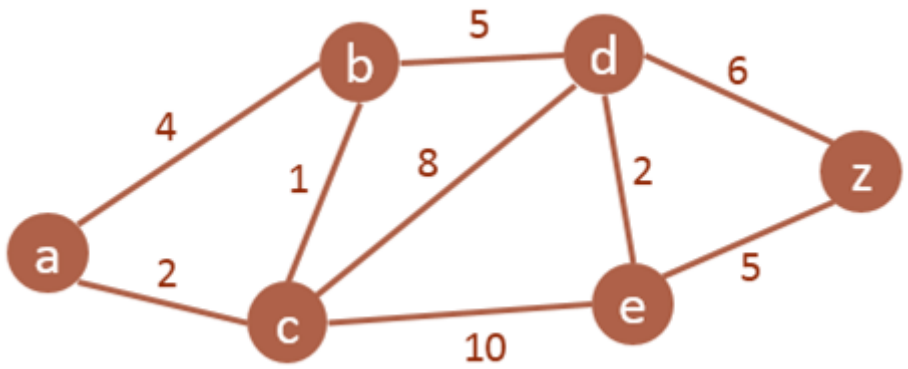
Node	Status	Shortest distance from A	Previous Node
a	CN	o	
b		∞	
c		∞	
d		∞	
e		∞	
z		∞	

Dijkstra's Algorithm



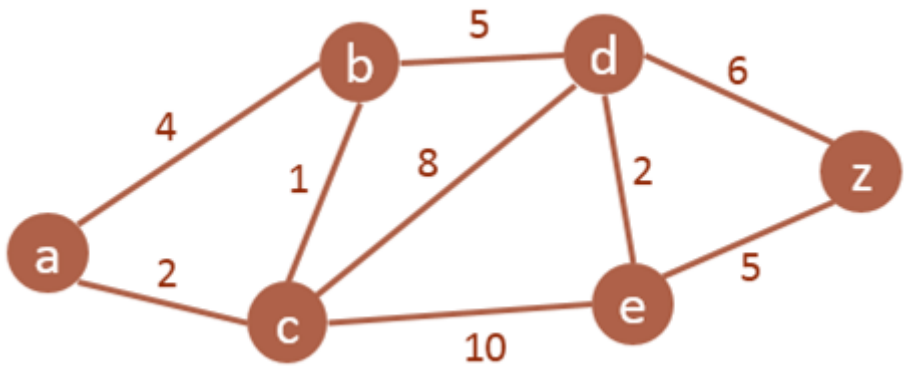
Node	Status	Shortest distance from A	Previous Node
a	CN	o	
b		∞, 4	A
c		∞, 2	A
d			
e			
z			

Dijkstra's Algorithm



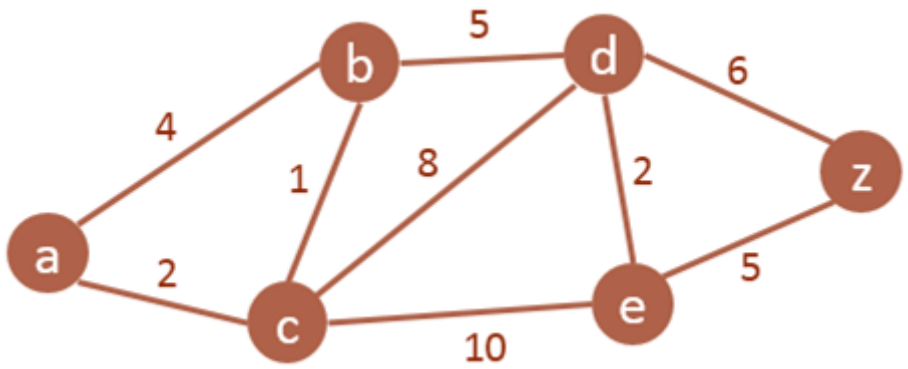
Node	Status	Shortest distance from A	Previous Node
a	VN	o	
b		~~∞~~, ~~4~~, 2+1 = 3	~~A~~, C
c	CN	~~∞~~, ~~=~~2	A
d		~~∞~~, 2+8=10	C
e		~~∞~~, 2+10=12	C
z			

Dijkstra's Algorithm



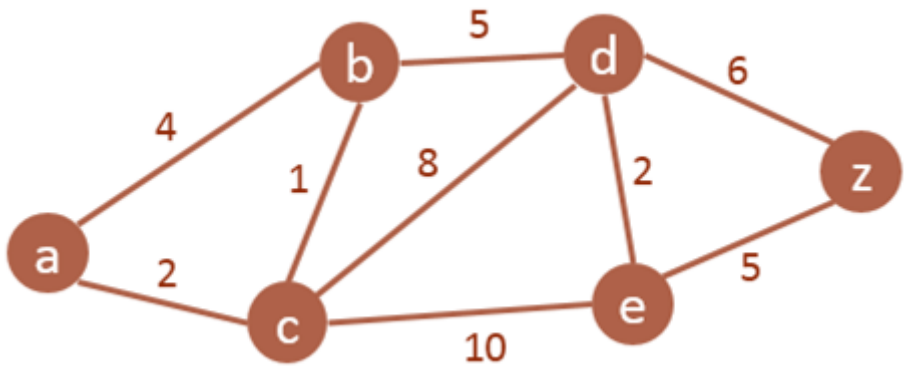
Node	Status	Shortest distance from A	Previous Node
a	VN	o	
b	CN	~~∞, 4~~, 2+1 = 3	~~A~~, C
c	VN	~~∞,~~2	A
d		~~∞,~~ ~~2+8=10~~ , 3+5 = 8	~~C~~ , B
e		~~∞,~~ 2+10=12	C
z			

Dijkstra's Algorithm

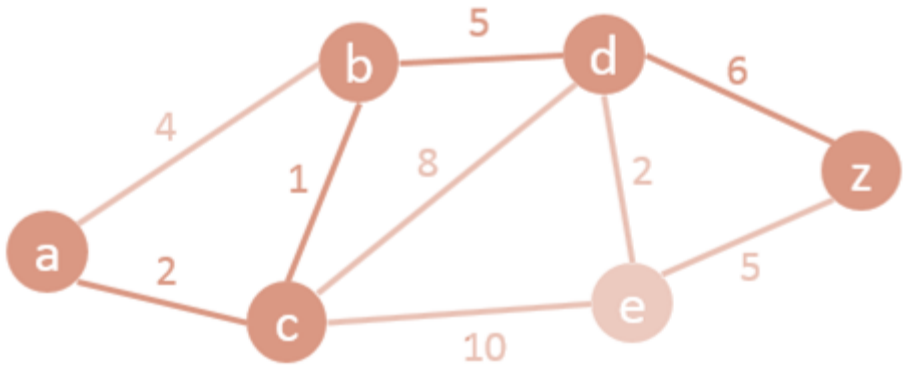


Node	Status	Shortest distance from A	Previous Node
a	VN	o	
b	VN	∞, ~~4~~, 2+1 = 3	~~A~~, C
c	VN	∞, 2	A
d	CN	∞, ~~2+8=10~~ , 3+5=8	~~C~~, B
e		∞, ~~2+10=12~~, 8+2=10	~~C~~, D
z		∞, 8+6=14	D

Dijkstra's Algorithm



Node	Status	Shortest distance from A	Previous Node
a	VN	o	
b	VN	∞, ~~4~~, 2+1 = 3	~~A~~, C
c	VN	∞, 2	A
d	VN	∞, ~~2+8=10~~ , 3+5=8	~~C~~, B
e	VN	∞, ~~2+10=12~~, 8+2=10	~~C~~, D
z	CN	∞, 8+6=14, ~~10+5=15~~	D



Node	Status	Shortest distance from A	Previous Node
a	Visited Node	0	
b	Visited Node	3	C
c	Visited Node	2	A
d	Visited Node	8	B
e	Visited Node	10	D
z	Current Node	14	D

A – C – B – D – Z with a length of 14

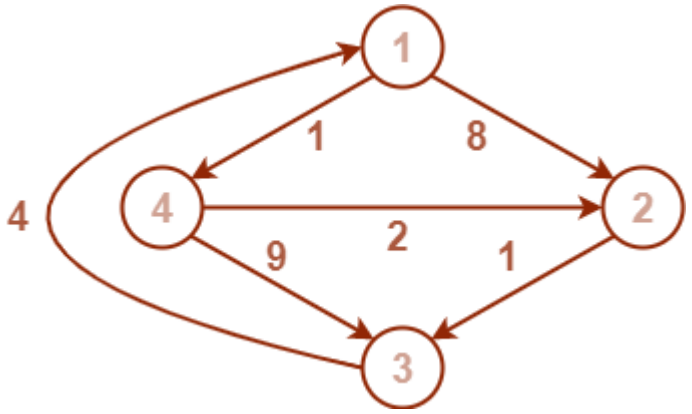
Floyd Warshal's Algorithm

Floyd Warshall's Algorithm is used to find the shortest paths between all pairs of vertices in a graph, where each edge in the graph has a weight which is positive or negative.

Algorithm

- Remove all the self loops and parallel edges (keeping the lowest weight edge) from the graph.
- Write the initial distance matrix.
- It represents the distance between every pair of vertices in the form of given weights.
- For diagonal elements (representing self-loops), distance value = 0.
- For vertices having a direct edge between them, distance value = weight of that edge.
- For vertices having no direct edge between them, distance value = ∞.

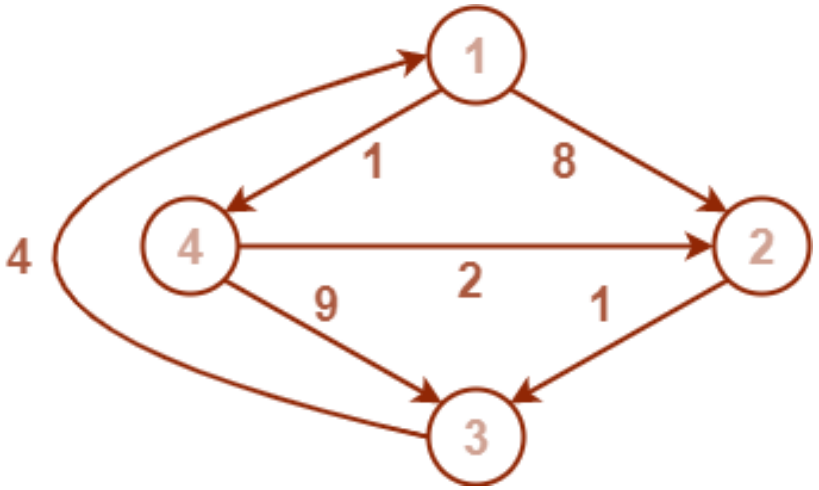
Floyd Warshal's Algorithm



$D_0 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	∞	0	∞
4	∞	2	9	0

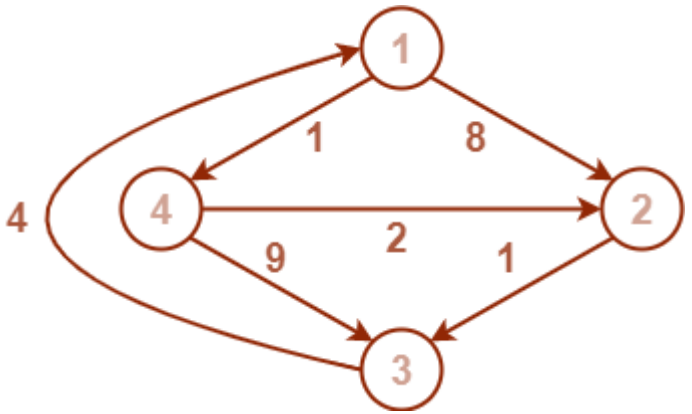
Floyd Warshal's Algorithm



$D_1 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	9	0

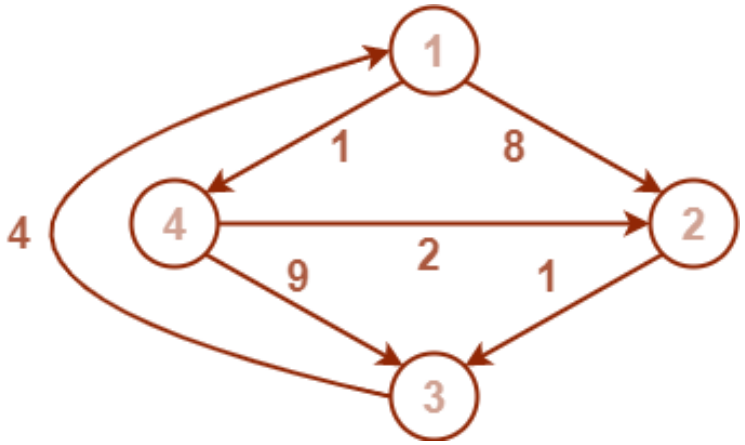
Floyd Warshal's Algorithm



$D_2 =$

	1	2	3	4
1	0	8	9	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	3	0

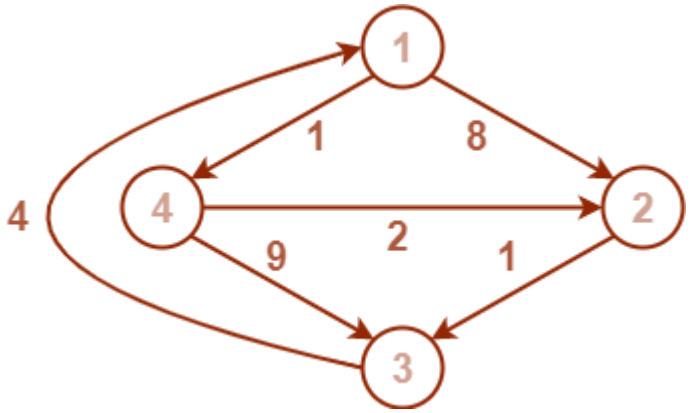
Floyd Warshal's Algorithm



$D_3 =$

	1	2	3	4
1	0	8	9	1
2	5	0	1	6
3	4	12	0	5
4	7	2	3	0

Floyd Warshal's Algorithm



$$D_4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{array}$$