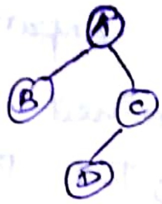
✓ Tree is a special case of graph having no loops, no circuits and no self loops.

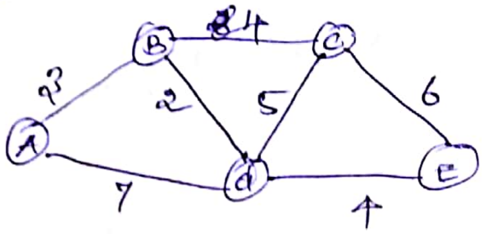


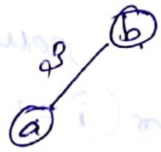
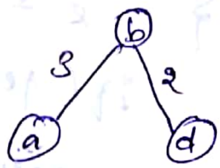
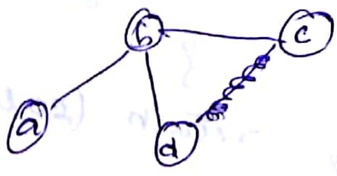
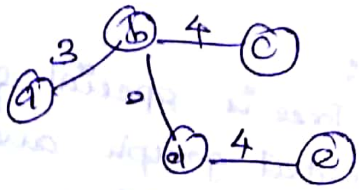
Dijkstra's Algorithm:

Dijkstra's algorithm considers the single source shortest paths problem. For a given vertex called the source in a weighted connected graph, find the shortest paths to all its other vertices.

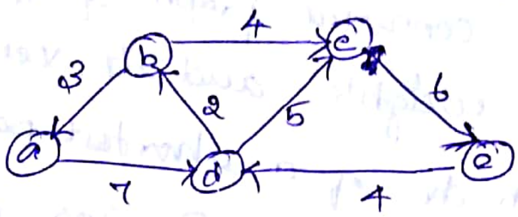
Note: It will not find the single shortest path that starts at the source and visits all the other vertices.

(1) Apply Dijkstra's algorithm.



Tree vertices	Remaining vertices	Illustration
a(−,0)	b(a,3), c(−,∞) d(a,7), e(−,∞)	
b(a,3)	c(b,3+4), d(b,3+2) e(−,∞)	
d(b,5)	c(b,7), e(d,5+4)	
c(b,7)	e(d,9)	
e(d,9)		

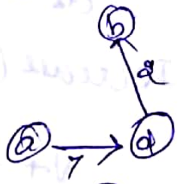
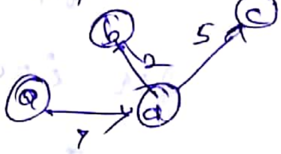
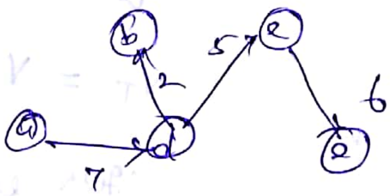
② Solve the single source shortest path problem with vertex a as the source



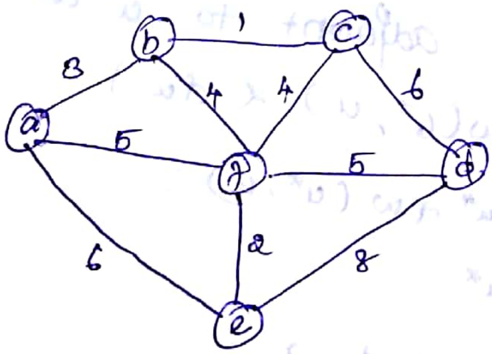
(a) Solve the SSSP problem for directed weighted graph.

(b) Find a shortest path between two given vertices of a weighted graph or digraph

(c) Find the shortest paths to a given vertex from each other vertex of a graph. (single-destination shortest path problem)

Tree vertices	Remaining vertices	Illustration
a(-, 0)	b(-, ∞), c(-, ∞) d(a, 7), e(-, ∞)	
d(a, 7)	b(d, 7+2), c(d, 7+5) e(~~d, ∞~~).	
b(d, 9)	c(d, 12), e(-, ∞)	
c(d, 12)	e(c, 12+6)	

③



Algorithm Dijkstra (G, s)

// D.alg for single source shortest paths

// i/p : A weighted connected graph $G = \{V, E\}$ with non-negative weights and its vertex s.

// o/p : The length d_v of a shortest path from s to v and its penultimate vertex P_v for every vertex v in V

{

Initialize (Q) // Initialize priority Q. to empty

for every vertex v in V do

$d_v = \infty$;

P_v = null

Insert (Q, v, d_v) // Initialize vertex priority in the priority Queue.

$d_s = 0$;

Decrease (Q, s, d_s) // update priority of s with d_s

$V_T = \phi$

for i = 0 to |V| - 1 do

u^* = DeleteMin (Q) // delete the min priority element

$V_T = V_T \cup \{u^*\}$

for every vertex u in $V - V_T$ that is adjacent to u^* do

if ($d_{u^*} + w(u^*, u) < d_u$)

$d_u = d_{u^*} + w(u^*, u)$;

$P_u = u^*$

Decrease (Q, u, d_u)

}

Algorithm Analysis:

Time complexity:

$T(n) = O(|E| \log |V|)$

$S(n) = O(|E| + |V|)$