SINGLE-SOURCE SHORTEST PATH ALGORITHMS

SINGLE-SOURCE SHORTEST PATH ALGORITHMS

- The shortest path problem is about finding a path between two vertices in a graph such that the total sum of the edges weights is minimum.
- ***single-source shortest-paths problem*****:** given a graph $G = (V, E)$, we want to find a shortest path from a given ***source*** vertex $s \in V$ to every vertex $v \in V$.

- ***Single-source shortest-paths algorithms:***
  - Bellman Ford's Algorithm
  - Dijkstra's Algorithm

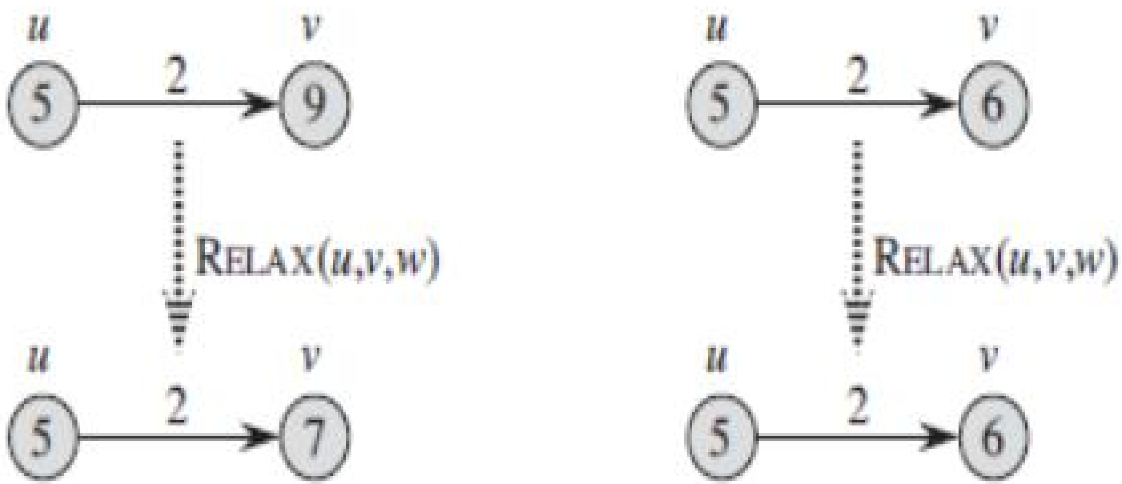
We initialize the shortest-path estimates and predecessors by the following procedure.

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
      v.d = ∞
      v.π = NIL
s.d = 0
```

- The process of ***relaxing*** an edge (u, v) consists of testing whether we can improve the shortest path to v found so far by going through u and, if so, updating v.*d* and v.π.
- A relaxation step may decrease the value of the shortest-path estimate v.*d* and update *v*'s predecessor field v.π

The following code performs a relaxation step on edge (u, v).

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```



(a) (b)

Dijkstra's Algorithm

- Dijkstra Algorithm is a very famous greedy algorithm.
- It is used for solving the single source shortest path problem.
- It computes the shortest path from one particular source node to all other remaining nodes of the graph.
- Dijkstra algorithm works only for connected graphs.
- Dijkstra algorithm works only for those graphs that do not contain any negative weight edge.
- Dijkstra algorithm works for directed as well as undirected graphs.

Algorithm Steps:

- Set all vertices distances = infinity except for the source vertex, set the source distance = 0.
- Initialize parent of each node to be NIL.
- Push the source vertex in a min-priority queue as the comparison in the min-priority queue will be according to vertices distances.
- Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).
- Update the distances of the connected vertices to the popped vertex in case of "v.d > u.d + w(u,v)", then push the vertex with the new distance to the priority queue.
- If the popped vertex is visited before, just continue without using it.
- Apply the same algorithm again until the priority queue is empty.

DIJKSTRA(G, w, s)

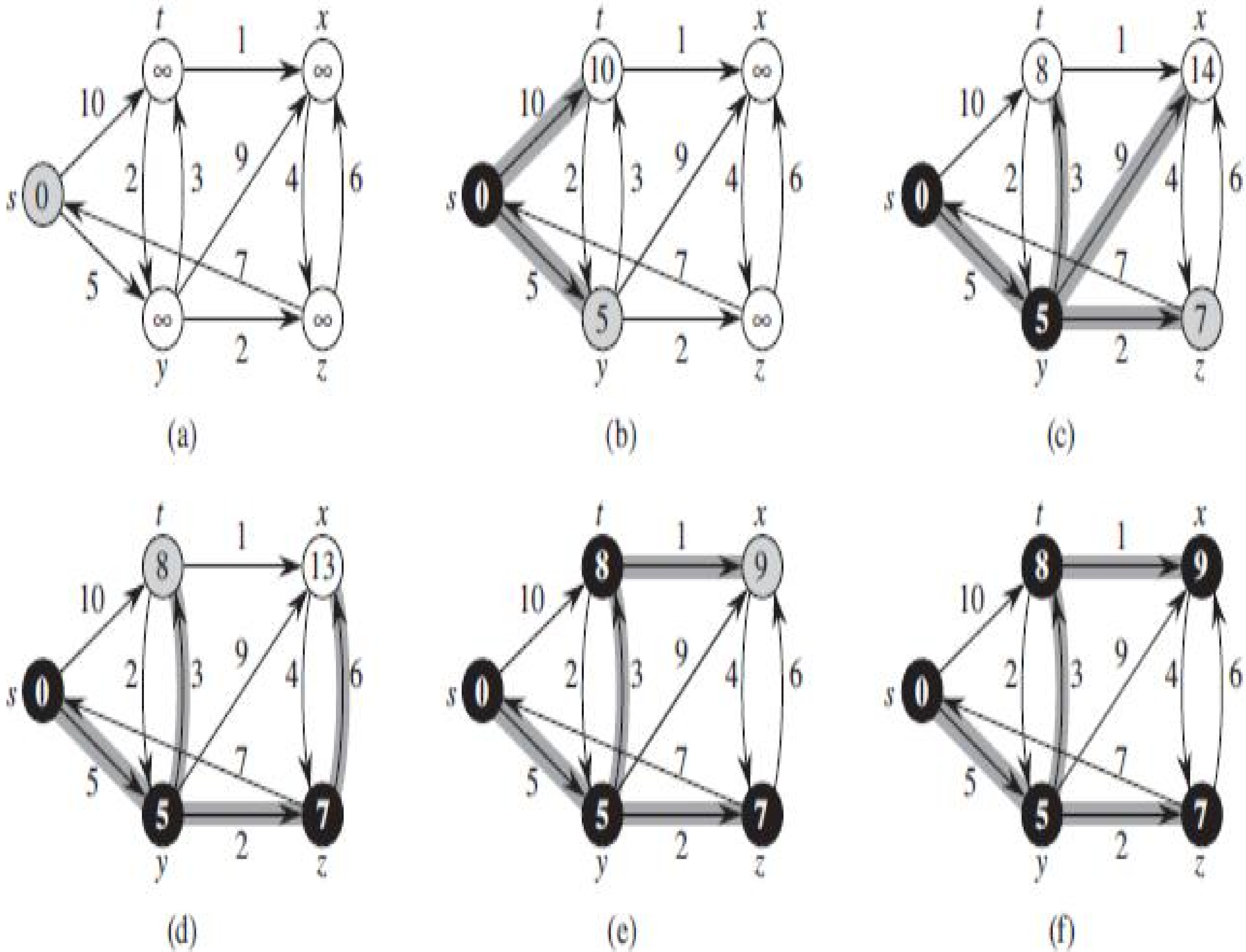
```
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

INITIALIZE-SINGLE-SOURCE(G, s)

```
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```



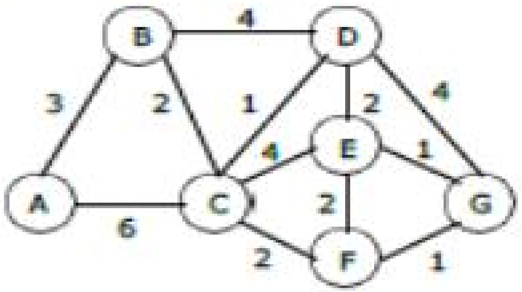
t
x
s
y
z
10
1
2
3
9
4
6
5
7
2
0
∞
(a)
10
5
(b)
8
14
7
(c)
13
(d)
9
(e)
(f)

Time Complexity

- **Time Complexity** of Dijkstra's Algorithm is **$O(V^2)$**
- but with min-priority queue it drops down to **$O(V+E \log V)$.**

Example 1:

Use Dijkstras algorithm to find the shortest path from A to each of the other six vertices in the graph:



Solution:

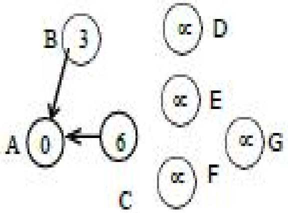
The cost adjacency matrix is $\begin{pmatrix} 0 & 3 & 6 & \infty & \infty & \infty & \infty \\ 3 & 0 & 2 & 4 & \infty & \infty & \infty \\ 6 & 2 & 0 & 1 & 4 & 2 & \infty \\ \infty & 4 & 1 & 0 & 2 & \infty & 4 \\ \infty & \infty & 4 & 2 & 0 & 2 & 1 \\ \infty & \infty & 2 & \infty & 2 & 0 & 1 \\ \infty & \infty & \infty & 4 & 1 & 1 & 0 \end{pmatrix}$

The problem is solved by considering the following information:

- Status[v] will be either '0', meaning that the shortest path from v to v_0 has definitely been found; or '1', meaning that it hasn't.
- Dist[v] will be a number, representing the length of the shortest path from v to v_0 found so far.
- Next[v] will be the first vertex on the way to v_0 along the shortest path found so far from v to v_0

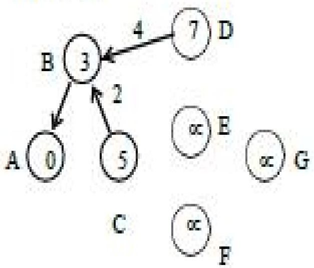
The progress of Dijkstra's algorithm on the graph shown above is as follows:

Step 1:



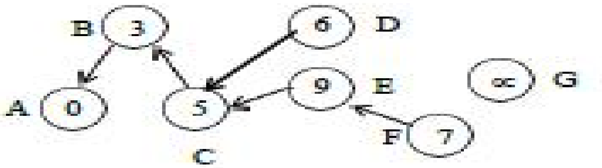
Vertex	A	B	C	D	E	F	G
Status	0	1	1	1	1	1	1
Dist.	0	3	6	∝	∝	∝	∝
Next	*	A	A	A	A	A	A

Step 2:



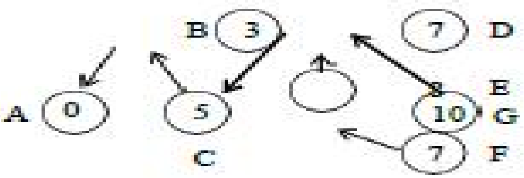
Vertex	A	B	C	D	E	F	G
Status	0	0	1	1	1	1	1
Dist.	0	3	5	7	∝	∝	∝
Next	*	A	B	B	A	A	A

Step 3:



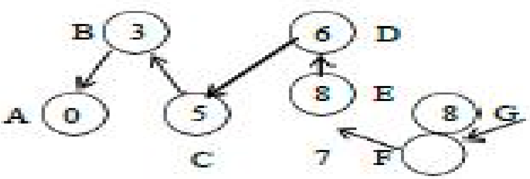
Vertex	A	B	C	D	E	F	G
Status	0	0	0	1	1	1	1
Dist.	0	3	5	6	9	7	∝
Next	*	A	B	C	C	C	A

Step 4:



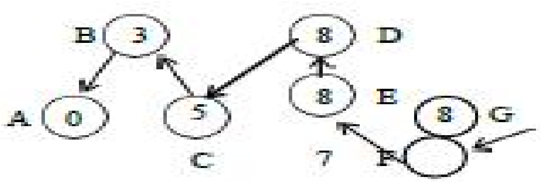
Vertex	A	B	C	D	E	F	G
Status	0	0	0	0	1	1	1
Dist.	0	3	5	6	8	7	10
Next	*	A	B	C	D	C	D

Step 5:



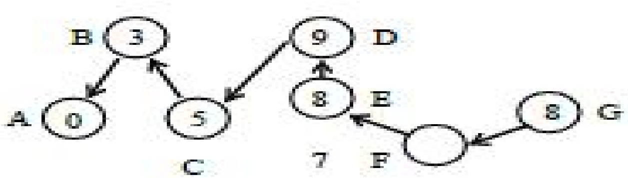
Vertex	A	B	C	D	E	F	G
Status	0	0	0	0	1	0	1
Dist.	0	3	5	6	8	7	8
Next	*	A	B	C	D	C	F

Step 6:



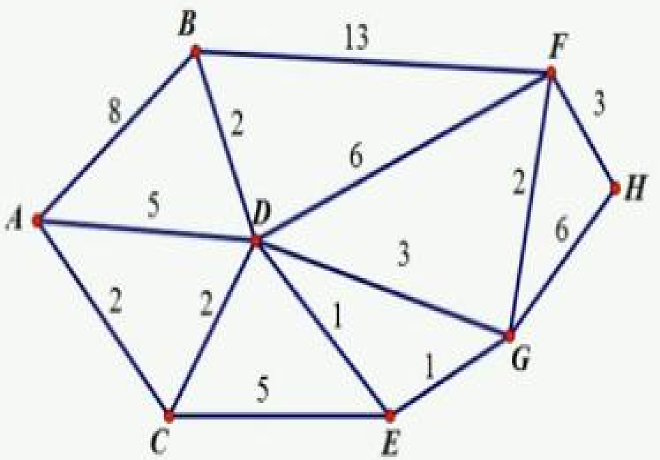
Vertex	A	B	C	D	E	F	G
Status	0	0	0	0	0	0	1
Dist.	0	3	5	6	8	7	8
Next	*	A	B	C	D	C	F

Step 7:



Vertex	A	B	C	D	E	F	G
Status	0	0	0	0	0	0	0
Dist.	0	3	5	6	8	7	8
Next	*	A	B	C	D	C	F

Example 2:

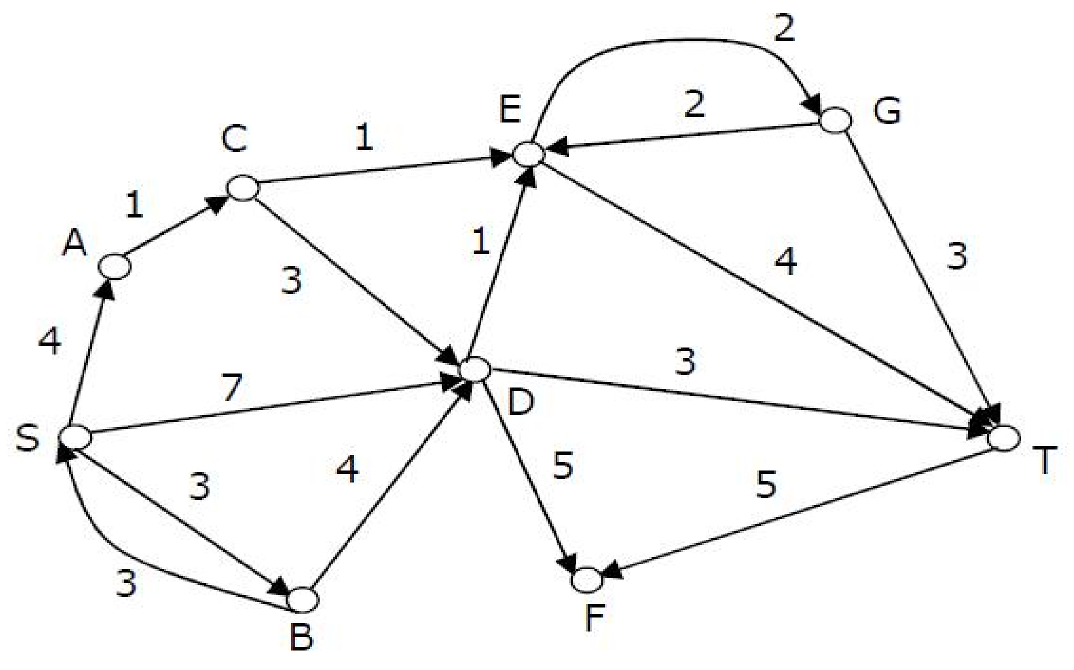


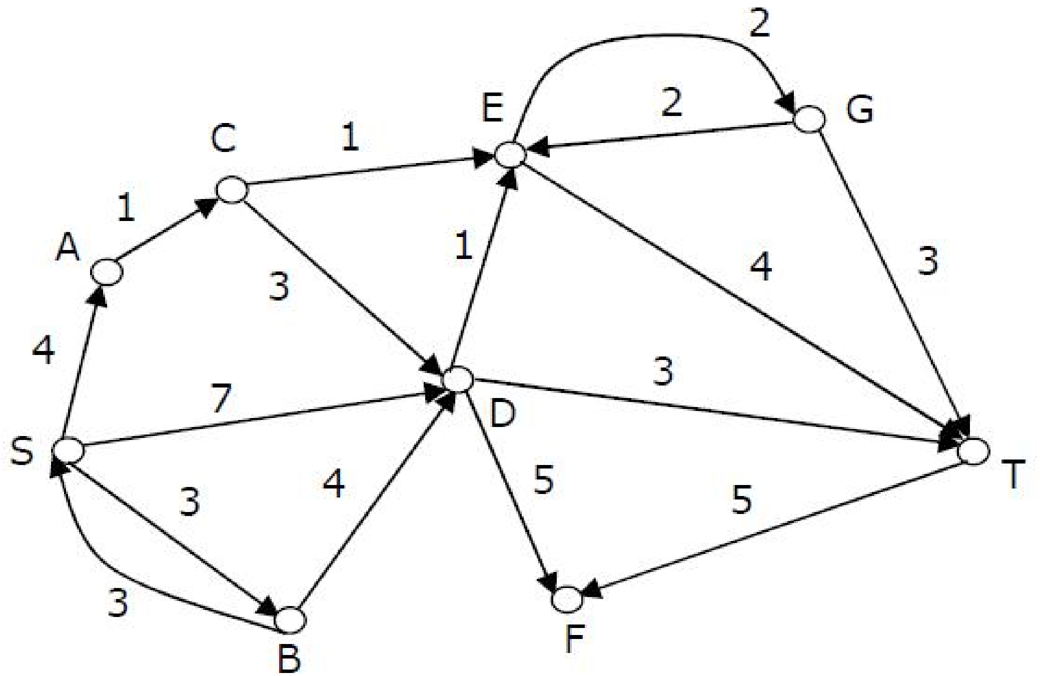
V	A	B	C	D	E	F	G	H
A	$\boxed{0_A}$	8_A	2_A	5_A	∞	∞	∞	∞
C		8_A	$\boxed{2_A}$	4_C	7_C	∞	∞	∞
D		6_D		$\boxed{4_C}$	5_D	10_D	7_D	∞
E		6_D			$\boxed{5_D}$	10_D	6_E	∞
B		$\boxed{6_D}$				10_D	6_E	∞
G						8_G	$\boxed{6_E}$	12_G
F						$\boxed{8_G}$		11_F
H								$\boxed{11_F}$

In a weighted graph, assume that the shortest path from a source 's' to a destination 't' is correctly calculated using a shortest path algorithm. Is the following statement true? If we increase weight of every edge by 1, the shortest path always remains same. Justify your answer with proper example. [May 2019- 3 marks]

- The shortest path may change. The reason is, there may be different number of edges in different paths from s to t. For example, let shortest path between a and c be of weight 8 and has 4 edges. Let there be another path with 2 edges and total weight 9. The weight of the shortest path is increased by 4*1 and becomes 8 + 4=12. Weight of the other path is increased by 2*1 and becomes 9 + 2=11. So the shortest path changes to the other path with weight as 11.
- *False – 1 Mark, Justification with example – 2 Marks.*

- **Find the shortest path from s to all other vertices in the following graph using Dijkstra's Algorithm.[May 2019- 3 marks]**





	A	B	C	D	E	F	G	T
B	4	**3**	∞	7	∞	∞	∞	∞
A	**4**		∞	7	∞	∞	∞	∞
C			**5**	7	∞	∞	∞	∞
E				7	**6**	∞	∞	∞
D				**7**		∞	8	10
G						12	**8**	10
T						12		**10**
F						**12**		

S-B =4, S-A =3, S-C=5, S-E=6, S-D=7, S-G=8, S-T=10, S-F =12