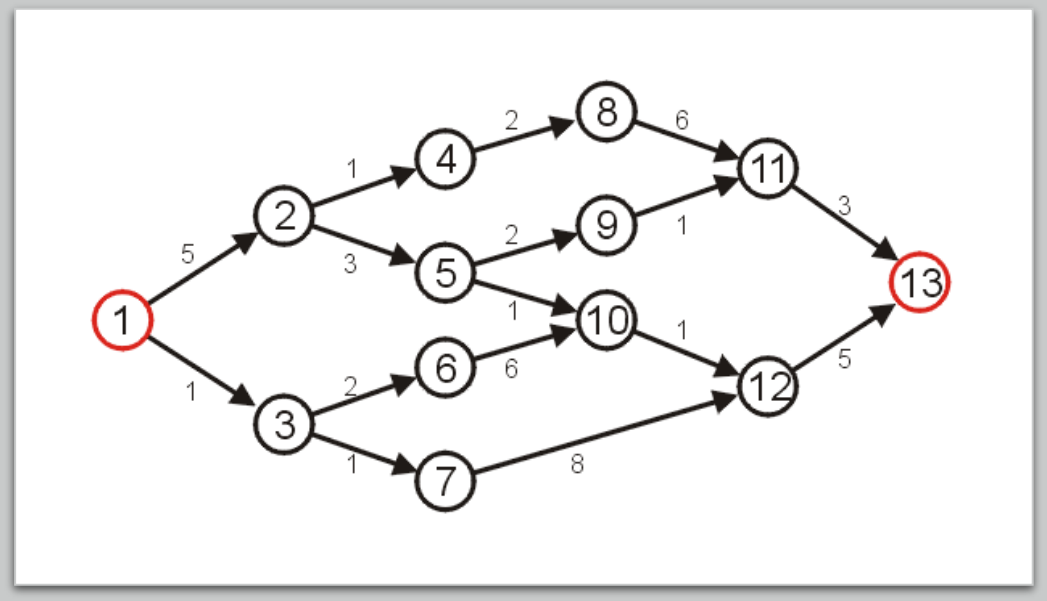
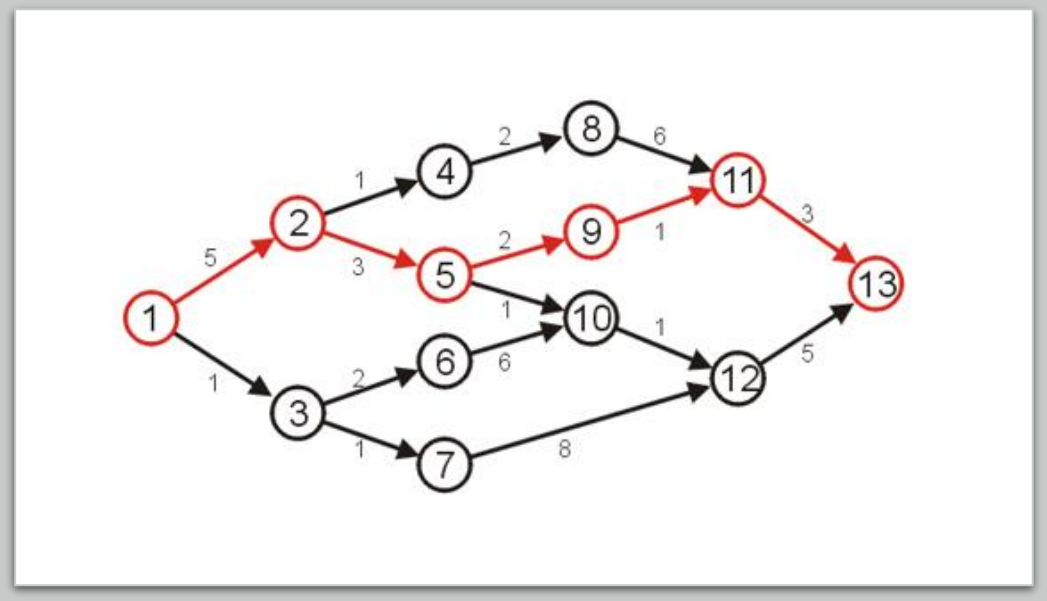
Dijkstra's algorithm

Shortest Path

- Given the graph below, suppose we wish to find the shortest path from vertex 1 to vertex 13

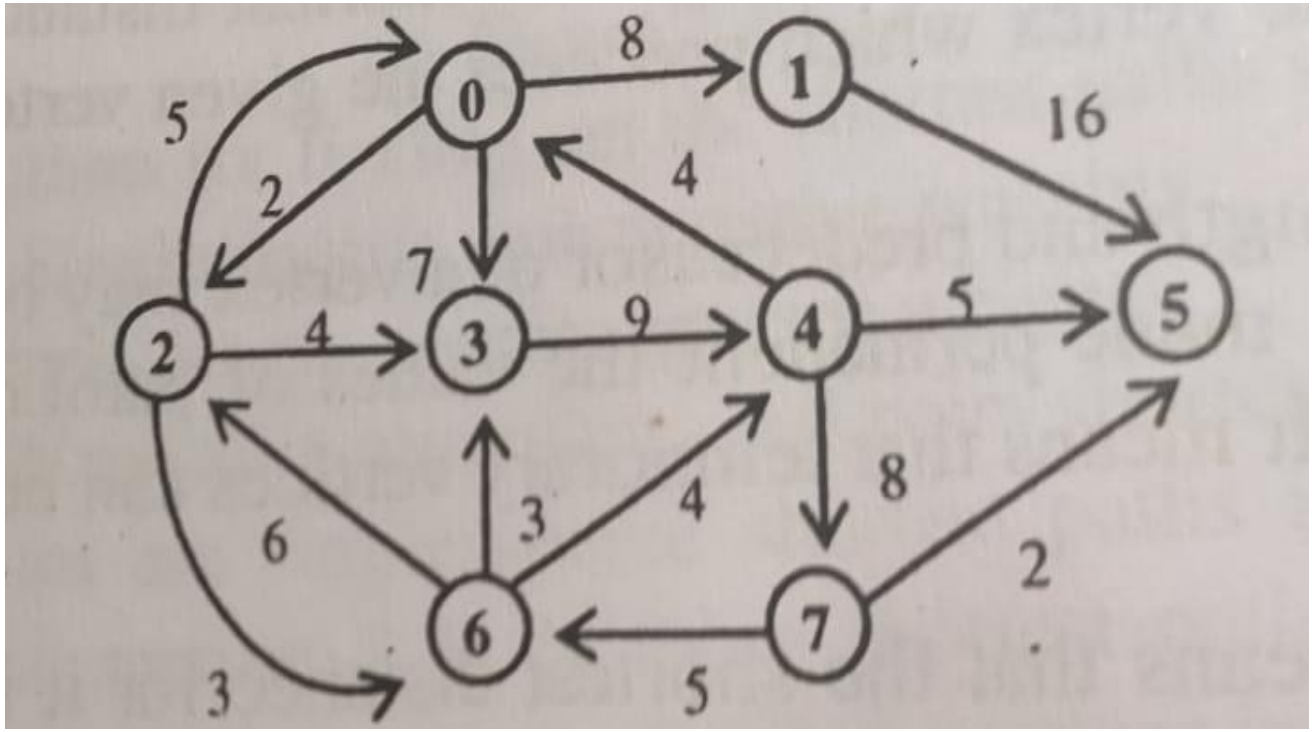
- After some consideration, we may determine that the shortest path is as follows, with length 14
- Other paths exists, but they are longer





Single-Source Shortest Path Problem

Single-Source Shortest Path Problem - The problem of finding shortest paths from a source vertex *v* to all other vertices in the graph.



Dijkstra's algorithm

Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory.

Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Approach: Greedy

Input: Weighted graph G={E,V} and source vertex $v \in V$, such that all edge weights are nonnegative

Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices

Dijkstra's algorithm

A. Initialize the pathlength of vertices to infinity(9999) and predecessor of all vertices to NIL(-1). Make the status of all vertices temporary.

B. Make the pathlength of source vertex equal to 0

C. From all the temporary vertices in the graph, find out the vertex that has minimum value of pathlength, make it permanent and now this is our current vertex.(if there are many such vertices, then any one can be taken)

D. Examine all the temporary vertices adjacent to the current vertex. The value of pathlength is recalculated for all these temporary successors of current and relabelling is done if required.

Dijkstra's algorithm

Suppose s is the source vertex, current is the current vertex and v is a temporary vertex adjacent to current.

If pathlength(current) + weight(current,v) < pathlength(v)

It means that path from s to v via current is smaller than the path currently assigned to v. We need to update the previous path.

pathlength(v) = pathlength(current) + weight(current,v)

predecessor(v) = current

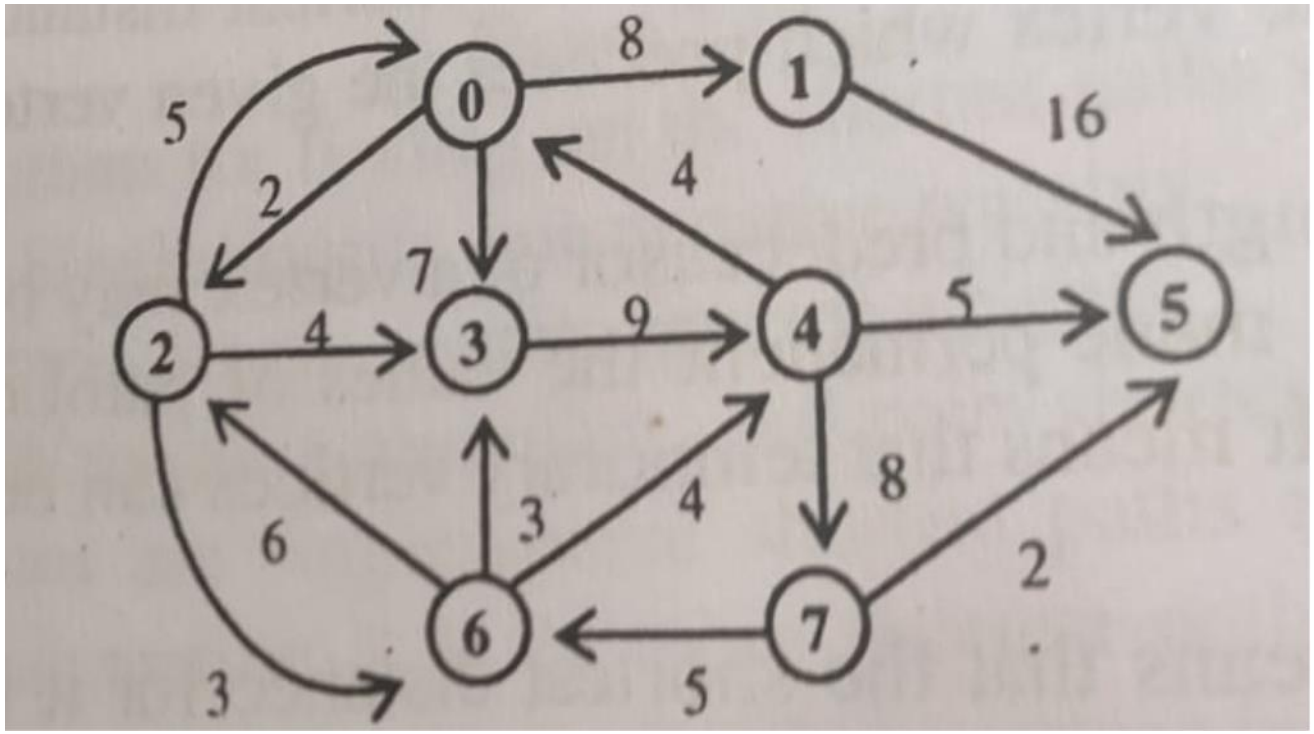
Dijkstra's algorithm

If pathlength(current) + weight(current,v) >= pathlength(v)

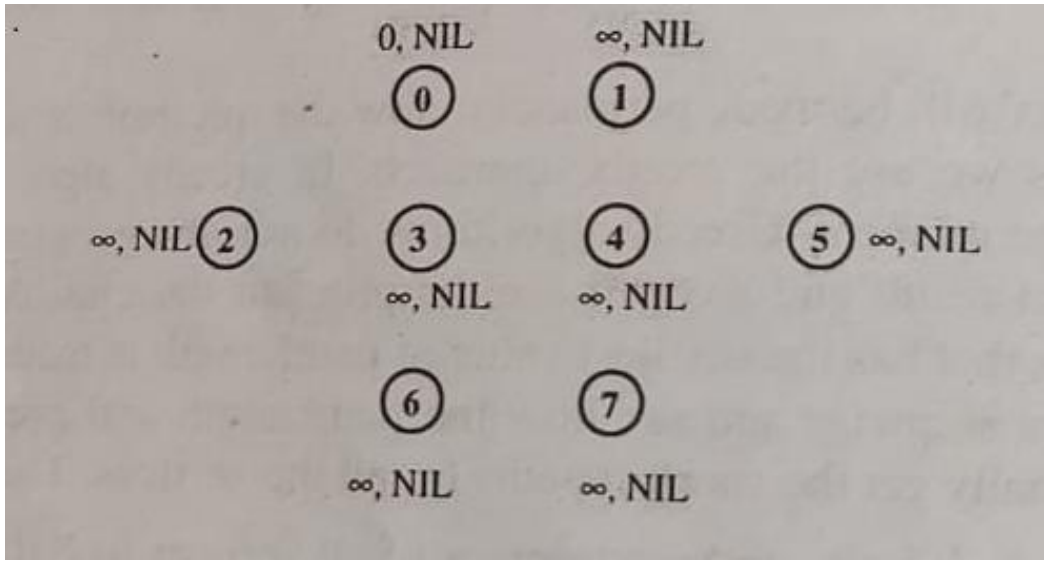
In this case vertex v is not relabelled and the values of pathlength and predecessor for vertex v remains unchanged.

E. Repeat steps C and D until there is no temporary vertex left in the graph

Dijkstra's algorithm on the graph below

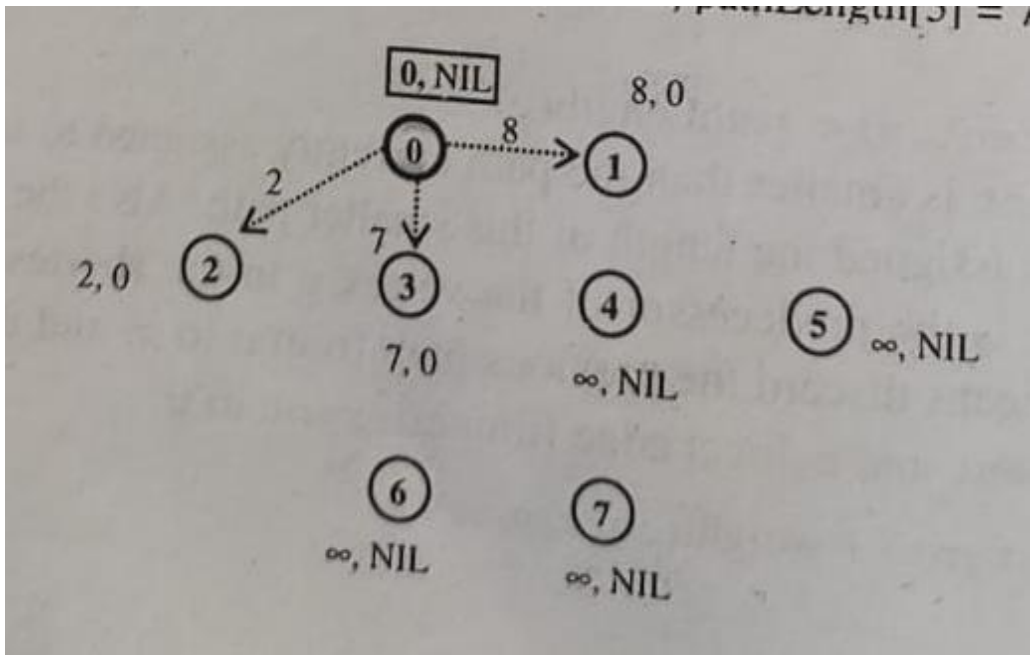


1



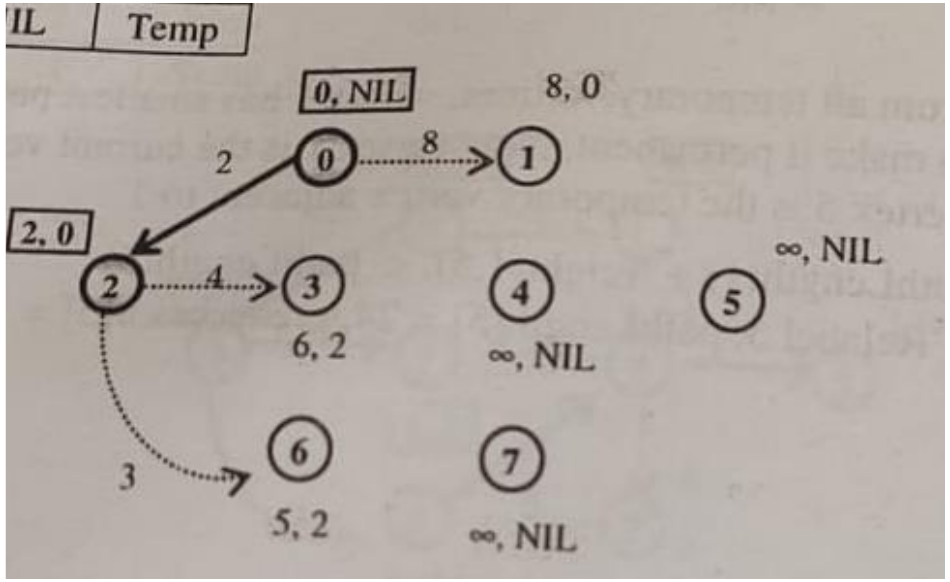
Vertex	path Length	prede-cessor	status
0	0	NIL	Temp
1	∞	NIL	Temp
2	∞	NIL	Temp
3	∞	NIL	Temp
4	∞	NIL	Temp
5	∞	NIL	Temp
6	∞	NIL	Temp
7	∞	NIL	Temp

2



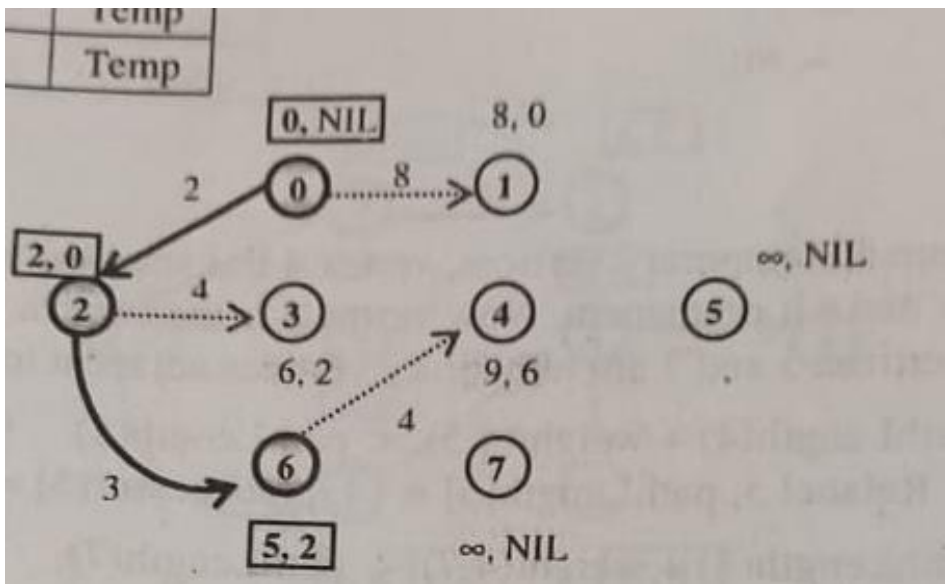
Vertex	path Length	prede-cessor	status
0	0	NIL	Perm
1	8	0	Temp
2	2	0	Temp
3	7	0	Temp
4	∞	NIL	Temp
5	∞	NIL	Temp
6	∞	NIL	Temp
7	∞	NIL	Temp

3

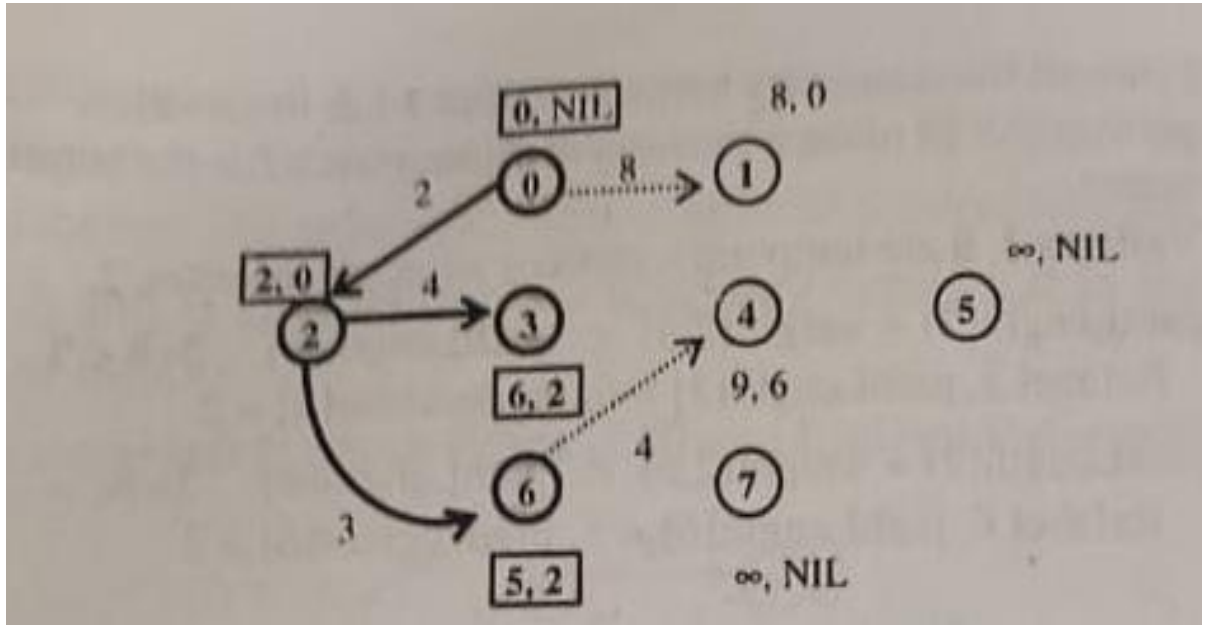


Vertex	path Length	prede -cessor	status
0	0	NIL	Perm
1	8	0	Temp
2	2	0	Perm
3	6	2	Temp
4	∞	NIL	Temp
5	∞	NIL	Temp
6	5	2	Temp
7	∞	NIL	Temp

4

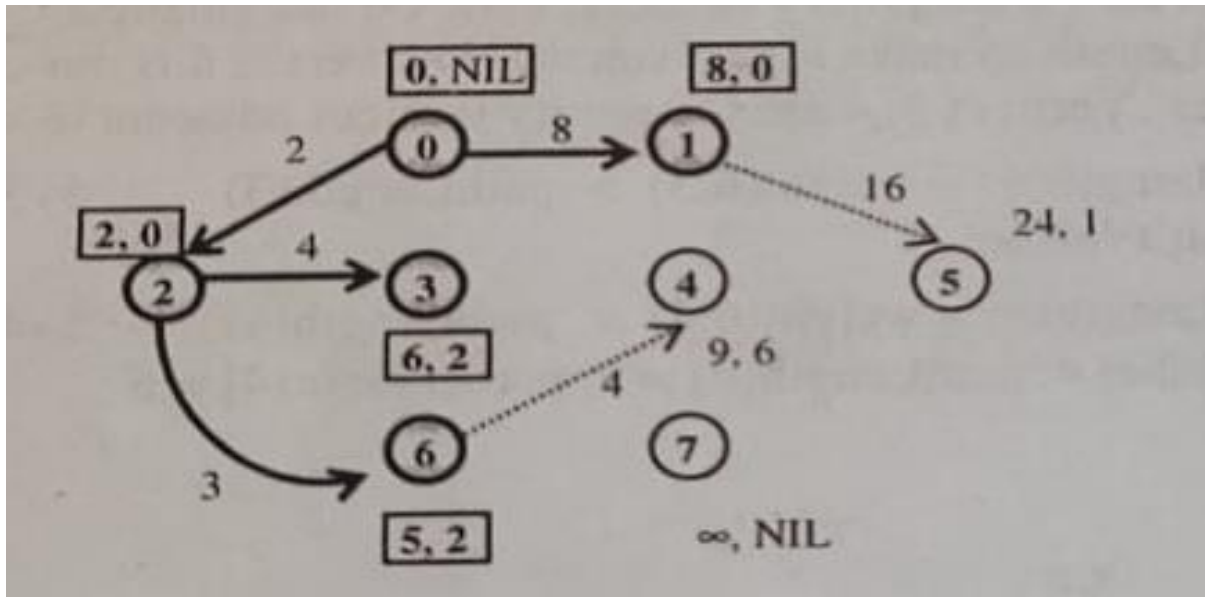


Vertex	path Length	prede-cessor	status
0	0	NIL	Perm
1	8	0	Temp
2	2	0	Perm
3	6	2	Temp
4	9	6	Temp
5	∞	NIL	Temp
6	5	2	Perm
7	∞	NIL	Temp



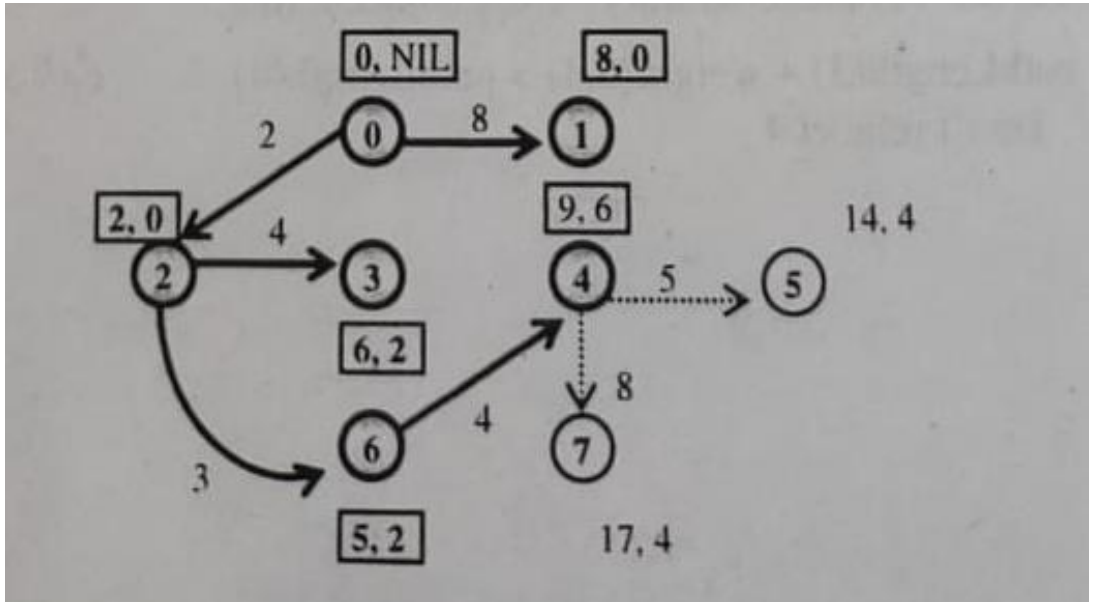
5

Vertex	path Length	prede -cessor	status
0	0	NIL	Perm
1	8	0	Temp
2	2	0	Perm
3	6	2	Perm
4	9	6	Temp
5	∞	NIL	Temp
6	5	2	Perm
7	∞	NIL	Temp

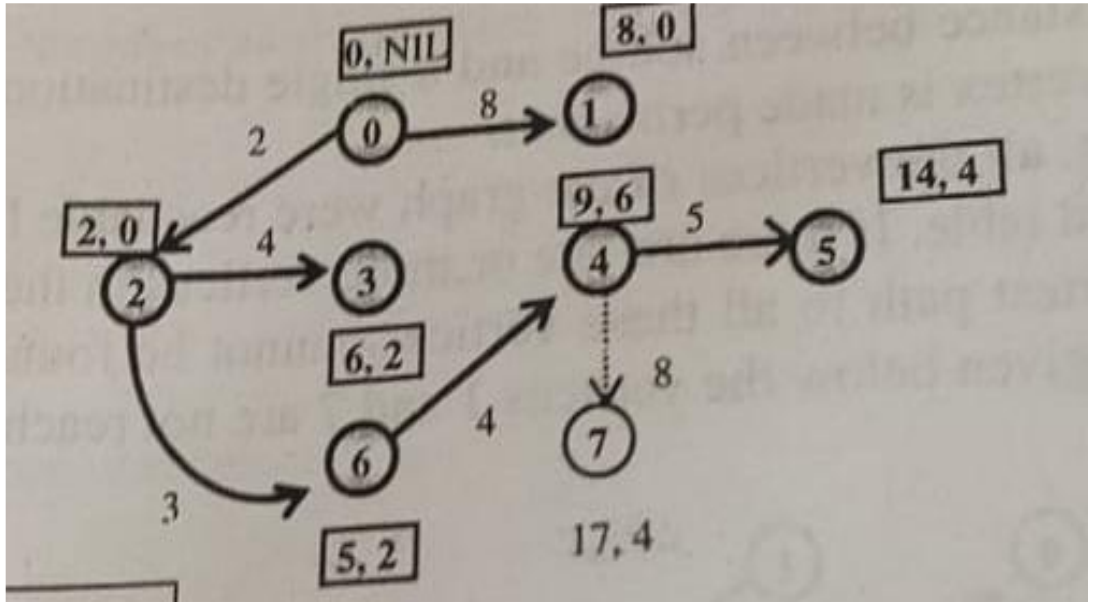


6

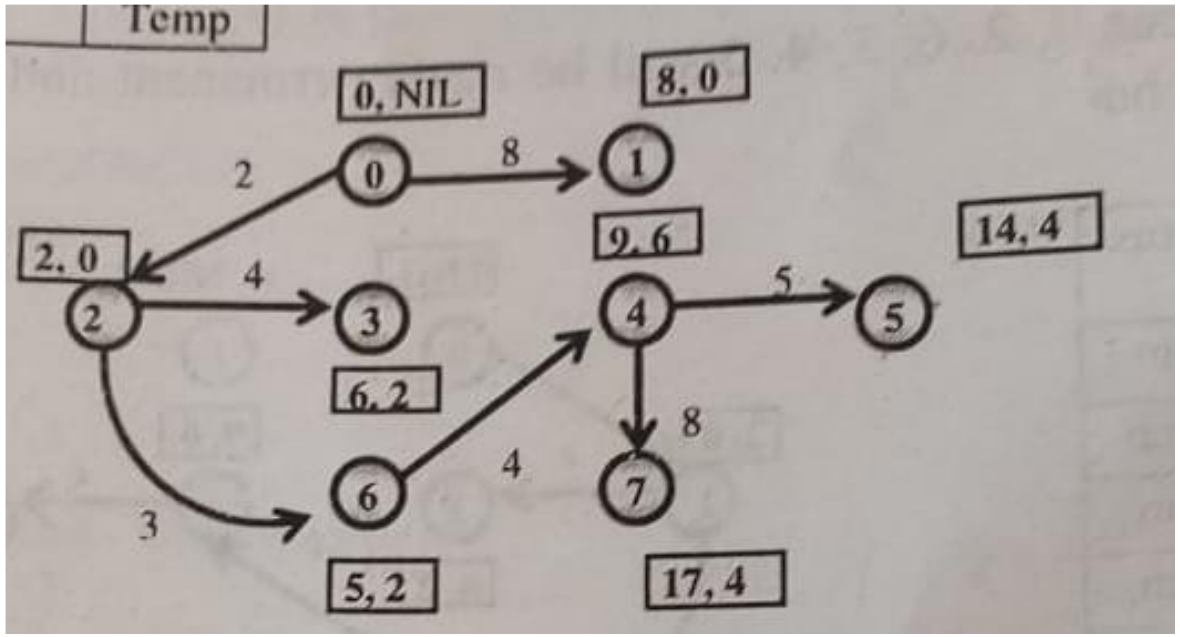
Vertex	path Length	prede-cessor	status
0	0	NIL	Perm
1	8	0	Perm
2	2	0	Perm
3	6	2	Perm
4	9	6	Temp
5	24	1	Temp
6	5	2	Perm
7	∞	NIL	Temp



Vertex	path Length	prede -cessor	status
0	0	NIL	Perm
1	8	0	Perm
2	2	0	Perm
3	6	2	Perm
4	9	6	Perm
5	14	4	Temp
6	5	2	Perm
7	17	4	Temp



Vertex	path Length	prede -cessor	status
0	0	NIL	Perm
1	8	0	Perm
2	2	0	Perm
3	6	2	Perm
4	9	6	Perm
5	14	4	Perm
6	5	2	Perm
7	17	4	Temp



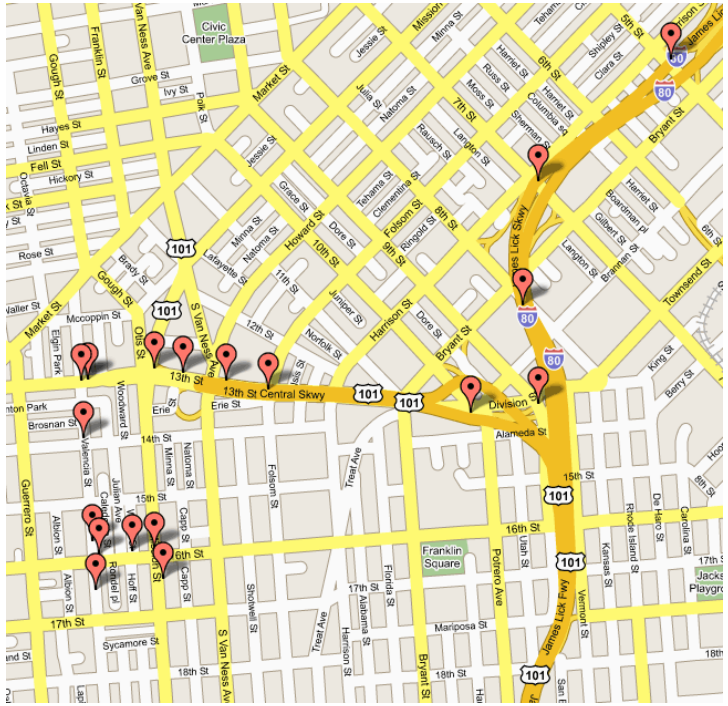
Vertex	path Length	prede-cessor	status
0	0	NIL	Perm
1	8	0	Perm
2	2	0	Perm
3	6	2	Perm
4	9	6	Perm
5	14	4	Perm
6	5	2	Perm
7	17	4	Perm

Dijkstra's Algorithm - Why use it?

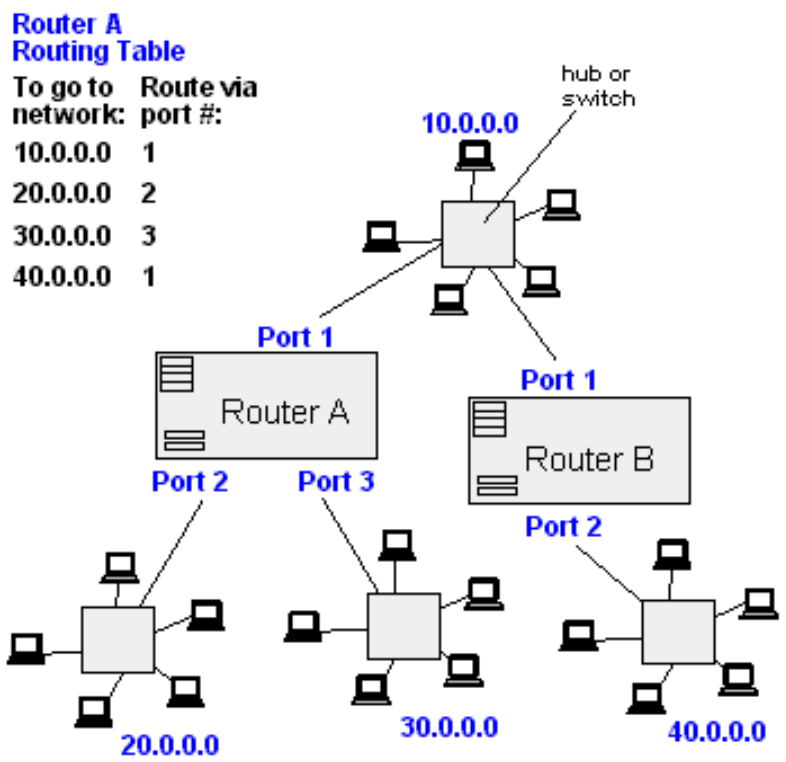
- As mentioned, Dijkstra's algorithm calculates the shortest path to every vertex.
- However, it is about as computationally expensive to calculate the shortest path from vertex *u* to every vertex using Dijkstra's as it is to calculate the shortest path to some particular vertex *v*.
- Therefore, anytime we want to know the optimal path to some other vertex from a determined origin, we can use Dijkstra's algorithm.

Applications of Dijkstra's Algorithm

- Traffic Information Systems are most prominent use
- Mapping (Map Quest, Google Maps)
- Routing Systems



From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.



Bellman Ford algorithm

Bellman Ford Algorithm

- The bellman ford algorithm works even when there are negative weight edges in the graph. It does not work if there is some cycle in the graph whose total weight is negative.
- In Dijkstra algorithm, we make a vertex Permanent at each step.
- In Bellman Ford algorithm the shortest distance is finalized at the end of the algorithm
- Here we drop the concept of making vertices permanent.
- Dijkstra’s algorithm is called as label setting algorithm and Bellman Ford algorithm is called as label correcting algorithm

Bellman Ford algorithm

A. Initialize the pathlength of all vertices to infinity, predecessor to NIL
B. Make the pathlength of source vertex equal to 0 and insert it into queue
C. Delete vertex from queue and make it current vertex
D. Examine all the vertices adjacent to the current. Check the condition of minimum weight for these vertices and do the relabelling if required, as in Dijkstra's algorithm.
E. Each label that is relabelled is inserted into queue provided it is not already present in the queue.
F. Repeat steps C, D, E till queue becomes empty.