①

Greedy Approach: Single Source Shortest Path Problem:

Given a directed graph $G(V, E)$ and weighting function cost for the edges of G and source vertex v_0.

The problem is to determine the shortest path from v_0 to all the remaining vertices of G. It is assumed that all the weights are positive.

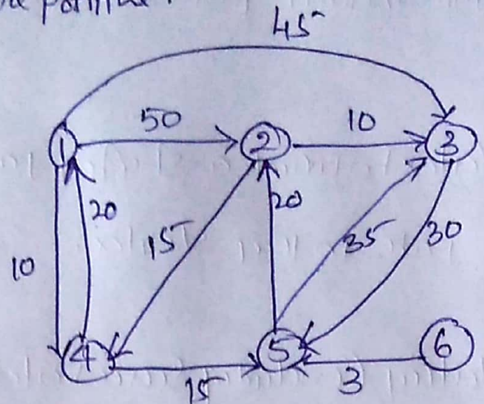


Fig: Graph

From the given directed graph $G(V, E)$ and source vertex 1,

$|V| = 6$, $|E| = 11$, and the numbers on the edges are the weight of graph.

Find the shortest path from source vertex 1 to all remaining vertices of the graph G.

The length of path = $\sum$(the edges in the path)

path	length
1 4	10
1, 4, 5	10+15=25
1, 4, 5, 2	10+15+20=45
1, 3	45
1, 6	∞ (because there is no path from source vertex 1 to vertex 6)

The Greedy Method to generate the shortest path from source vertex v_0 to the remaining vertices is to generate these paths in nondecreasing order of path length.

Scanned by CamScanner

⇒ First, shortest path from source vertex v_0 to nearest vertex is generated. Then shortest path to the second nearest vertex is generated, then shortest path to the third nearest vertex to be generated and so on.

In order to generate the shortest path in this order, we need to able to determine

1) the next vertex to which a shortest path must be generated
ii) A shortest path to this vertex.

Let S = set of vertices (including source vertex v_0) to which the shortest paths have already been generated.

For the vertices w which are not in S, let

dist[w] = length of shortest path from vertex v_0 (source vertex) going through only those vertices that are in S and ending at w. We observe the following

1) if the next shortest path is to vertex u, then path begins at v_0, ends at u and goes through only those vertices that are in S.

ii) The destination of next path generated must be that of vertex u which has the minimum distance dist[u], among all the vertices not in S. In case there are several vertices not in S with the same dist[ ], then any of these may be selected.

iii) when u is selected as nearest vertex to V_0, then vertex u becomes a member of S. At this point the length of shortest path starting at V_0, going through vertices only in S, and ending at a vertex W not in S may decrease i.e. dist[w] may change.

3

if dist[w] does change, then it must be due to shorter path starting at v_0 and going to u and then to w. The intermediate vertices on the v_0 to u path and u to w path are all in S. Also the u to w path can be chosen so as not to contain any intermediate vertices.

⇒ the length of path from v_0 to u to w

= dist[u] + cost (u, w)

Time complexity: $O(n\ e)$ — n → number of vertices; e → number of edges.

Algorithm shortest path(v, cost, dist, n)

// n is the number of vertices in directed graph G(V, E)

// cost is the cost adjacency matrix of G(V, E) represented through cost[1 .. n, 1 .. n]

// dist[j], $1 \le j \le n$ is set to the length of shortest path from vertex v to vertex j in digraph G(V, E)

// V is the source vertex of digraph G(V, E)

```
{   for i ← 1 to n do
    {
        S[i] ← false, dist[i] = cost[v, i]   ; // initialize
    }
    S[v] ← True ;   dist[v] ← 0.   // put v in S.

    for num ← 2 to n-1 do
    {                  // determine n-1 paths from v. choose u from
                       // among those vertices not in S such that
                       // dist[u] is minimum
        S[u] ← True;   // put u in S.
        for (each w adjacent to u with S[w] ← false) do
            // update distance
            if (dist[w] > (dist[u] + cost[u, w])) then
                dist[w] ← dist[u] + cost[u, w];
    }}
```

Scanned by CamScanner

(4) eg: Find single source shortest path starting from vertex 5 in the given graph.

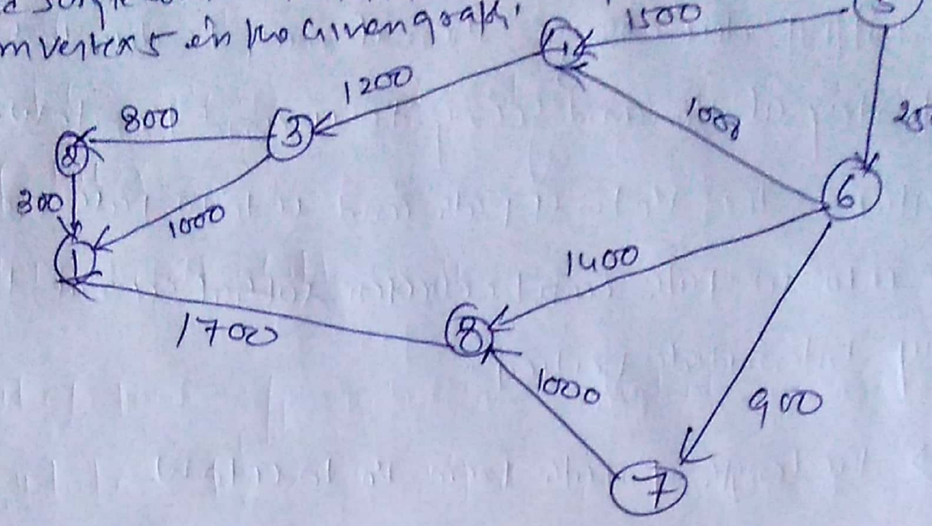


Fig: Diagram G(V, E)

The cost Adjacency matrix of above graph.

$$
Cost[i,j] = \begin{array}{c|cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
1 & 0 & - & - & - & - & - & - & - \\
2 & 800 & 0 & - & - & - & - & - & - \\
3 & 1000 & 800 & 0 & - & - & - & - & - \\
4 & - & - & 1200 & 0 & - & - & - & - \\
5 & - & - & - & 1500 & 0 & 250 & - & - \\
6 & - & - & - & 1000 & - & 0 & 900 & 1400 \\
7 & - & - & - & - & - & - & 0 & 1000 \\
8 & 1700 & - & - & - & - & - & - & 0
\end{array}
$$

Iteration	S	Vertex selected	Distance 1	2	3	4	5	6	7	8
Initial	{-}	–	∞	∞	∞	1500	0	250	∞	∞
1	{5}	6	∞	∞	∞	1250	0	250	1150	1650
2	{5,6}	7	∞	∞	∞	1250	0	250	1150	1650
3	{5,6,7}	4	∞	∞	2450	1250	0	250	1150	1650
4	{5,6,7,4}	8	3350	∞	2450	1250	0	250	1150	1650
5	{5,6,7,4,8}	3	3350	3250	2450	1250	0	250	1150	1650
6	{5,6,7,4,8,3}	2	3350	3250	2450	1250	0	250	1150	1650
	{5,6,7,4,8,3,2}									

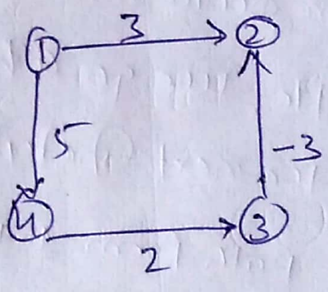
Fig: Action of shortest path starting from vertex 5

From the above example, we find a shortest path from vertex 5 to all remaining vertices of the given graph G(V, E) i.e. shortest path length from vertex 5 to all remaining vertices are as follows:

shortest path	length
5 6	250
5 6 7	1150
5, 6, 4	1250
5, 6, 7, 8	1650
5, 6, 4, 3	2450
5, 6, 4, 3, 2	3250
5, 6, ~~7~~, 8, 1	3350

⇒ Greedy approach gives an optimal solution for single source shortest path problem (when we assume all weights of edges are positive)

Let us consider an example →



if we find a single source shortest path from vertex 1 to all remaining vertices:

S	distance 1	2	3	4
{1}	0	3	∞	5
{1, 2}	0	3	∞	5
{1, 2, 4}	0	3	7	5
{1, 2, 4, 3}	0	3	7	5

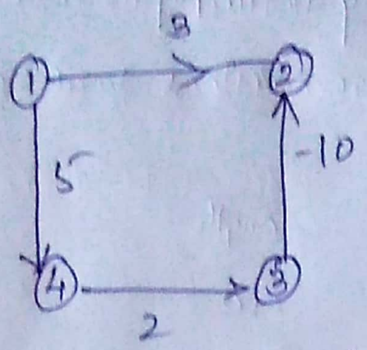
when we go through one more step ahead then we also find the same distance

dist[2] > dist[3] + cost[3, 2]

3 > 7 − 3 = 4

⇒ ~~cost~~ dist[2] is remain no change.

⑥ Let us consider the same example with small changes



Find the single source shortest path problem from source vertex 1 to all remaining vertices.

Iteration	S	distance 1	2	3	4
1	{1}	0	3	∞	5
2	{1,2}	0	3	∞	5
3	{1,2,4}	0	3	7	5
	{1,2,4,3}	0	-3	7	5

since
$dist[4] < dist[2] + c(2,4)$
$5 < 3 + \infty$
$\Rightarrow dist[4]$ not changed

As per dijkstra Algo this step is not needed but in this step giving shorter distance.

As per the dijkstra Algorithm after $(n-1)$th iteration we find a shortest path from source vertex to all remaining vertices of G. but in above example, we go ahead one more step and consider vertex 3 is the next shortest distance vertex.

$\Rightarrow$ the distance of path from 1 to 2 through 3

$= \underline{1 \to 4 \to 3 \to 2} = 5 + 2 - 10 = -3$

which is less than the previous distance of vertex 2

$\Rightarrow$ We can say that dijkstra algorithm may or may not work for negative weight edge graph.