Tutorial 11 Solution.

1. Modify Dijkstra's algorithm to compute the number of shortest paths from s to every vertex t.

    a. Add a new variable nos(v)=0, for all v. nos(s)=1

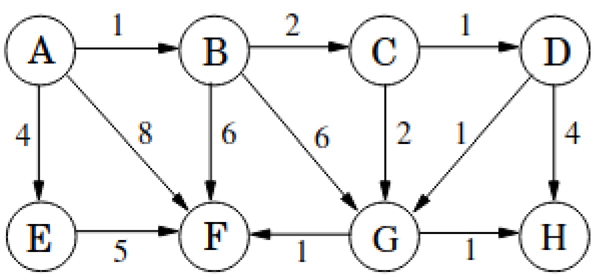
    b. Add the line under the following if d(v)<d(u)+l(u,v).

        nos(v)=nos(u);

    c. add the following code:

```
If d(v)==d(u)+l(u,v)
    nos(v)=nos(v)+nos(u);
```

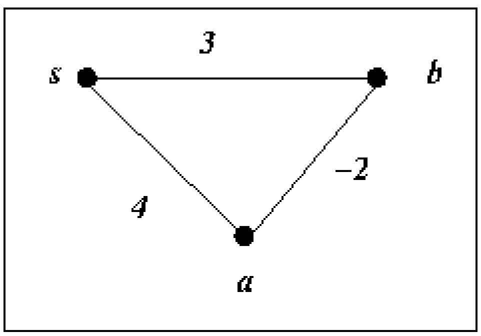
2. Compute the shortest path for all vertices starting from A. Do this in tabular form.



A	B	C	D	E	F	G	H	Queue	u=delmin	d(u)
0	infty	infty	infty	infty	infty	infty	infty	A	A	0
	1	infty	infty	4	8	infty	infty	B,E,F	B	1
		3	infty	4	7	7	infty	C,E,F,G	C	3
			4	4	7	5	infty	D,E,F,	D	4

								G		
				4	7	5	8	E,F,G, H	E	4
					7	5	8	F,G,H	G	5
					6		6	F,H	F	6
							6	H	H	6

3. Show an example of a graph with negative edge weights and show how Dijkstra's algorithm may fail. Suppose that the minimum negative edge weight is -d. Suppose that we create a new graph G' with weights w', where G' has the same edges and vertices as G, but w'(e)=w(e)+d. In other words, we have added d to every edge weight so that all edges in the new graph has edge weights non-negative. Let us run Dijkstra on this graph. Will it return the shortest paths for G?



The fact about Dijkstra is that once a vertex leaves the queue, its distance from s is never updated. Start Dijkstra from s and see that d(b)=3 is what Dijkstra will produce. The correct answer is 2. Also note that adding a constant will give wrong answers. The simple reason being that the cost of a path will change depending on the number of edges in the path. Adding 3 to the graph above does not identify the correct path.

4. Look at the following graph from Tutorial 9 with red edges and blue edges. Our task was to find the path from s to every vertex t, with the fewest red edges. Run any modified bfs of your choice and Dijkstra and compare the sequence of vertices visited by BFS and by Dijkstra.

5. You are given a time table for a city. The city consists of n stops V={v1,v2,...,vn}. It runs m services s1,s2,...,sm. Each service is a sequence of vertices and timings. For example, the schedule for service K7 is given below. Now, you are at stop A at 8:00am and you would like to reach stop B at the earliest possible time. Assume that buses may

be delayed by at most 45 seconds. Model the above problem as a shortest path problem. The answer should be a travel plan.

Service : K7			
H15	Convocation Hall	Market Gate	H15
7:15am	7:20am	7:30	7:40

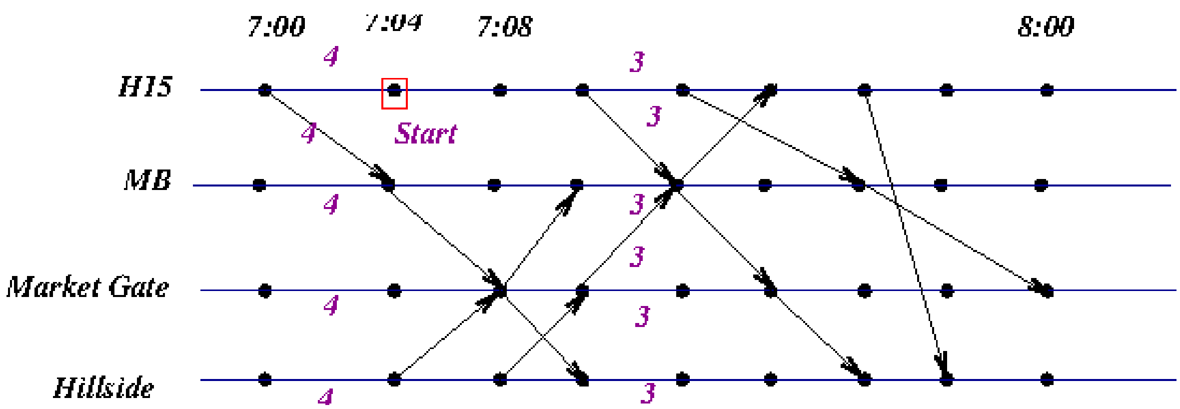


Figure 4.8 Dijkstra's shortest-path algorithm.

```
procedure dijkstra(G, l, s)
Input:    Graph G = (V, E), directed or undirected;
          positive edge lengths {l_e : e ∈ E}; vertex s ∈ V
Output:   For all vertices u reachable from s, dist(u) is set
          to the distance from s to u.

for all u ∈ V:
   dist(u) = ∞
   prev(u) = nil
dist(s) = 0

H = makequeue(V)  (using dist-values as keys)
while H is not empty:
   u = deletemin(H)
   for all edges (u, v) ∈ E:
      if dist(v) > dist(u) + l(u, v):
         dist(v) = dist(u) + l(u, v)
         prev(v) = u
         decreasekey(H, v)
```