Procedure DIJKSTRA(G, s)

Inputs: G as described above, and a starting vertex, *s*.

Results: For each non-source vertex *v* in V, *shortest[v]* is the weight of the shortest path from *s to* v and *pred[v]* is the vertex preceding *v* on some shortest path.

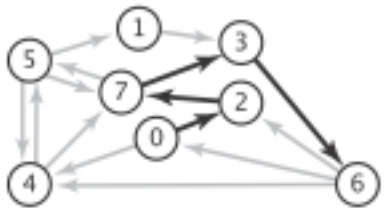
```
for each vertex v in V:
   shortest[v] = ∞
   pred[v] = null

shortest[s] = 0, pred[s] = null

for each vertex v in V:
   insert v in FRONTIER

while FRONTIER is not empty:
   u = remove from FRONTIER the vertex with shortest value
   for each vertex v adjacent to u:
      RELAX(u, v)
```

Procedure RELAX(*u, v*)
Inputs: *u, v* are vertices in V such that there is an edge between them
Result: Updates the values in *shortest[v]* and *pred[v]* if possible

```
if shortest[u] + weight(u, v) < shortest[v]
   shortest[v] = shortest[u] + weight(u, v)
   pred[v] = u
```

edge-weighted digraph

```
4->5  0.35
5->4  0.35
4->7  0.37
5->7  0.28
7->5  0.28
5->1  0.32
0->4  0.38
0->2  0.26
7->3  0.39
1->3  0.29
2->7  0.34
6->2  0.40
3->6  0.52
6->0  0.58
6->4  0.93
```



shortest path from 0 to 6

```
0->2  0.26
2->7  0.34
7->3  0.39
3->6  0.52
```

An edge-weighted digraph and a shortest path

shortest	
0	
1	
2	
3	
4	
5	
6	
7	

pred	
0	
1	
2	
3	
4	
5	
6	
7	

FRONTIER

_____