Data Structures

CSCI 2270-202: REC 11

Sanskar Katiyar

Logistics

Office Hours (This week)

Today: 5 pm – 7 pm

Friday: 3 pm – 5 pm

Assignment 7 Concerns

Interview grading opportunity

No notes on Assignment 8, 9

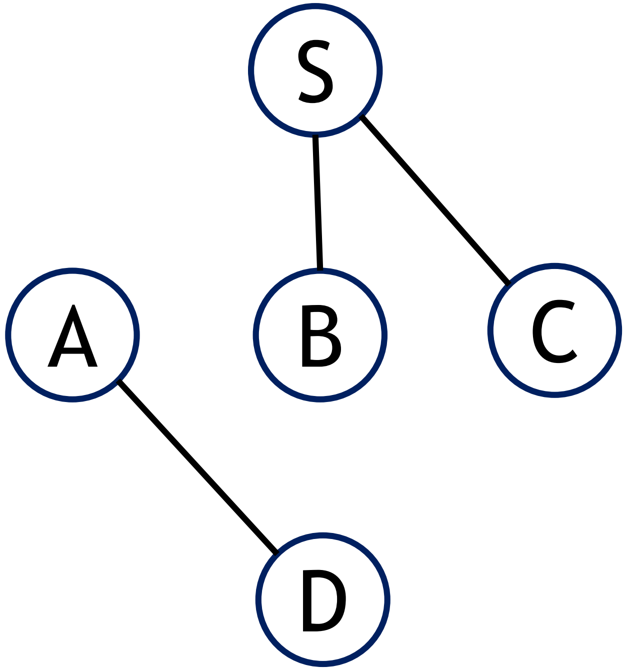
Most stuff covered in recitation, Check Github

Recitation Outline

1. Implementing Graph: Add/Remove, Display
2. BFS, DFS: Review
3. Dijkstra’s Shortest Path Algorithm
4. Exercise

BFS, DFS: Review

DFS: Pseudocode (Recursive)



```
DFS(G, u) {
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)
}
```

```
for each u ∈ G
    u.visited = false
for each u ∈ G
    if u.visited == false
        DFS(G, u)
```

Initialize all nodes as unvisited

Loop: If there is more than one component

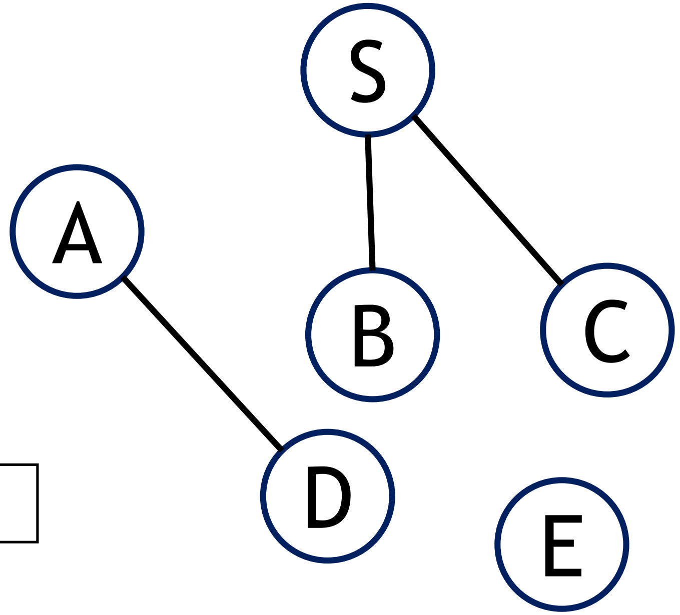
BFS: Pseudocode (Iterative)

```
BFS(G, u) {
 Q = Queue()
 Q.enqueue(u)
 u.visited = true
 while !Q.isEmpty()
  v = Q.peek(); Q.dequeue();

  for each w ∈ G[v]
   if w.visited == false
    Q.enqueue(w)
    w.visited = true
}
```

DFS: Finding Number of Components

```
DFS(G, u) {
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)
}
```



```
for each u ∈ G
    u.visited = false
for each u ∈ G
    if u.visited == false
        DFS(G, u)
```

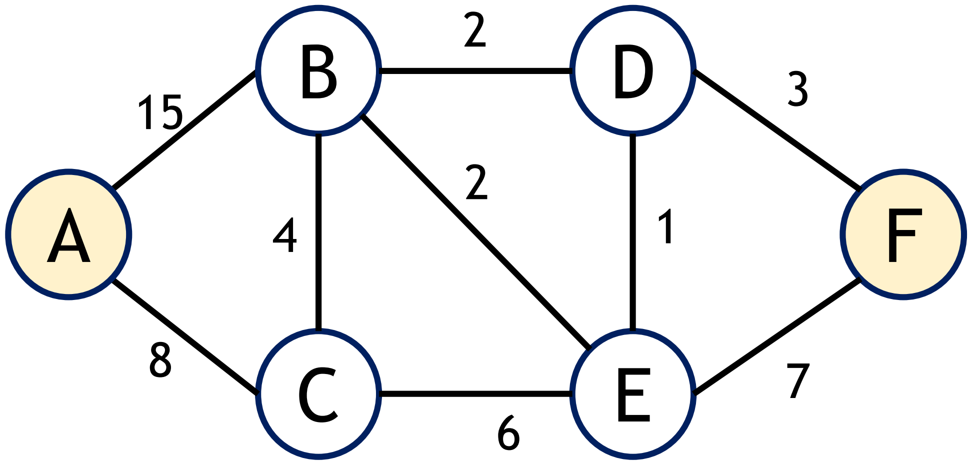
#components++;

Initialize all nodes as unvisited

Loop: If there is more than one component

Dijkstra’s Shortest Path

Breadth First Search: Shortest Path



Assume Edge weights are distances between two vertices

Shortest Path from A->F

A -> B -> D -> F (20)

A -> C -> E -> F (21)

A -> B -> E -> F (24)

A -> C -> B -> D -> F (17)

BFS: Shortest Path

Path with the smallest number of edges (agnostic to edge weight)

Dijkstra’s Shortest Path Algorithm

Edsger W. Dijkstra (in 1956)

Greedy Algorithm

Single-source shortest paths to all *reachable* vertices

Applicable to Weighted Graphs

Applications: Network Routing Protocols, Path Planning (Potential Field Methods), AI

Dijkstra’s Shortest Path Algorithm

Disclaimer: The following implementation is different from what is covered in lectures and the textbook.

The idea is to demonstrate a different implementation approach for the same algorithm.

Please ensure that you understand the implementation from the lecture before going forward with this.

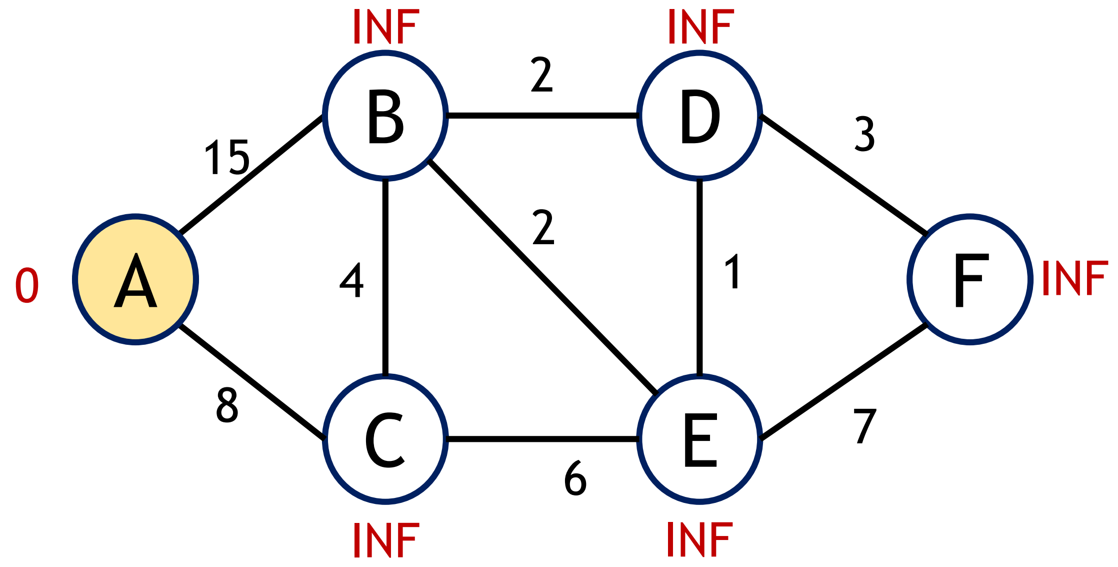
Dijkstra’s Shortest Path: Steps

1. From the source vertex, visit the vertex u with the least known distance
2. Once at the vertex u, check each of u’s neighbors
3. Calculate the distance for the neighbors by summing the cost of the edges leading from the source vertex
4. If the distance is less than a known distance, update the shortest distance for that vertex; mark the node from which the edge emanates as the predecessor

Dijkstra’s Shortest Path: Example

	A	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

-	A	B	C	D	E	F



Initialize the distance of all nodes from A as INF; Populate Queue/Array with all vertices

Source’s distance to itself is 0

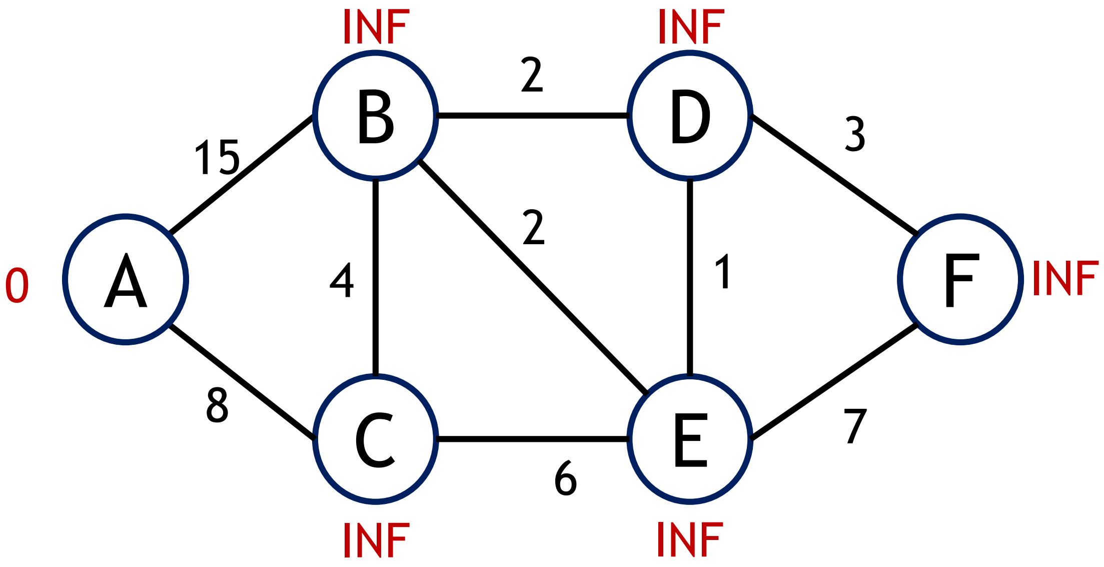
Why INF?

Think about non-reachable nodes, and comparison limits

Dijkstra’s Shortest Path: Example

	A	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

-	A	B	C	D	E	F



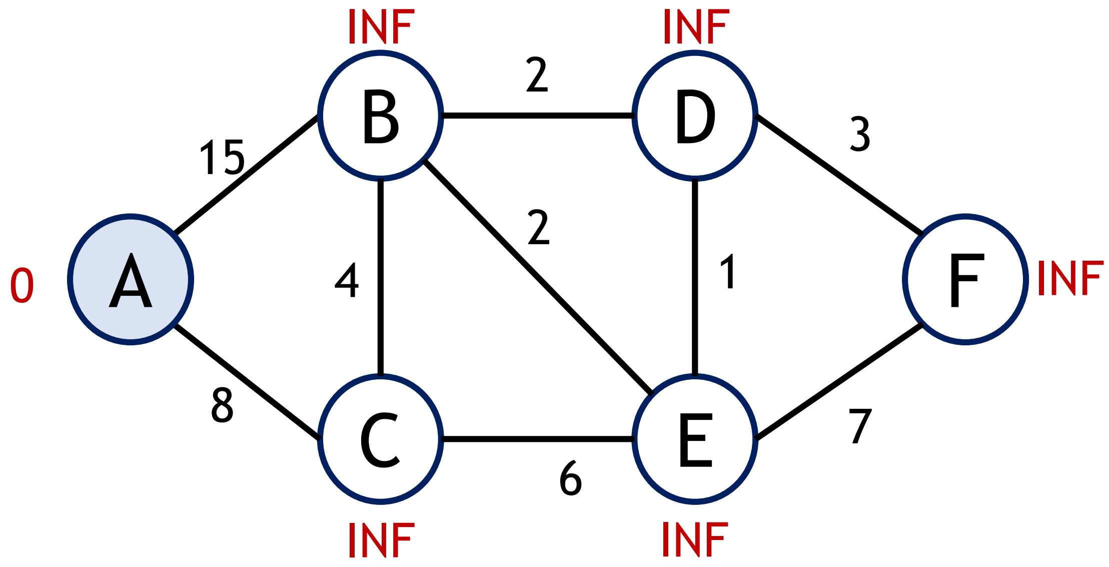
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

-	A	B	C	D	E	F



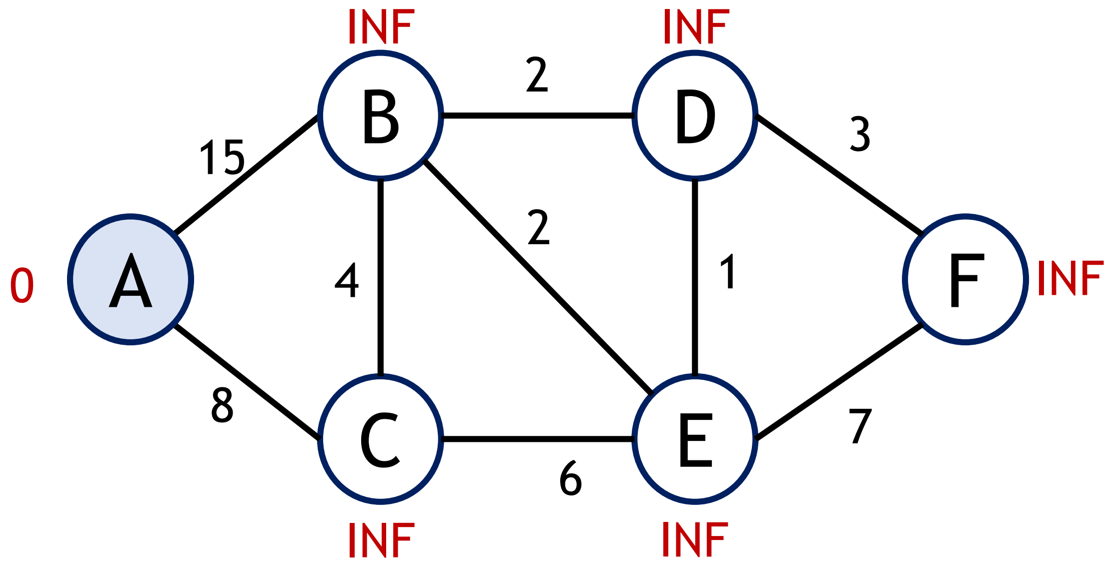
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

A	B	C	D	E	F	



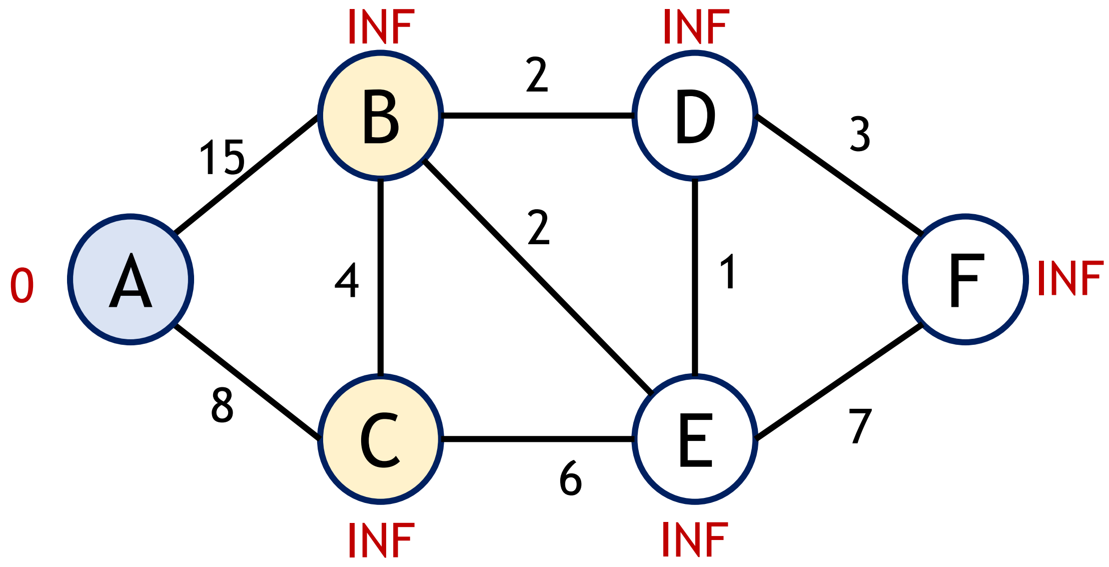
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

A	B	C	D	E	F	



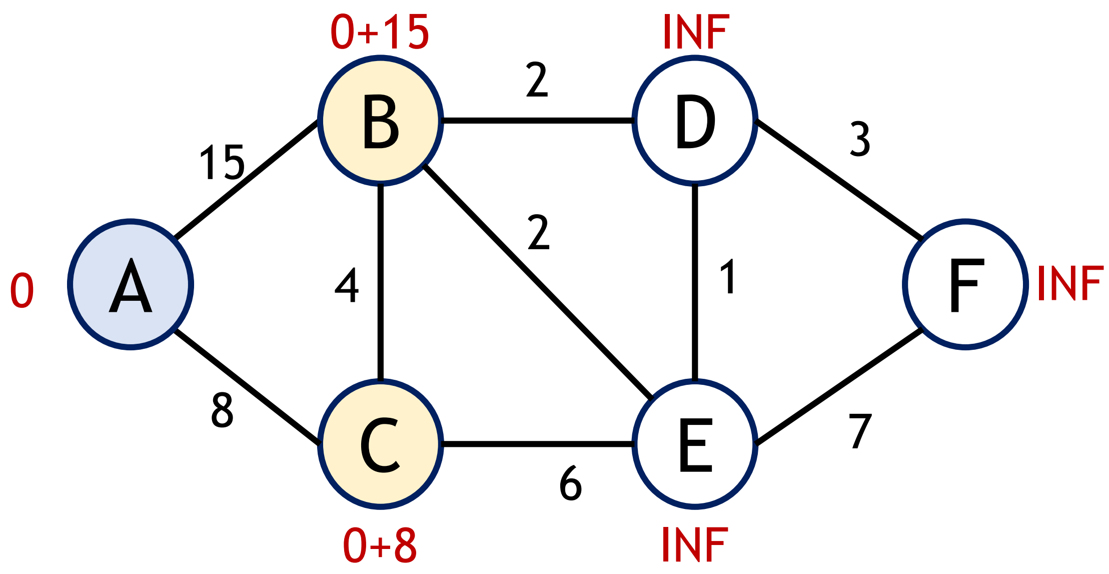
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

A	B	C	D	E	F	



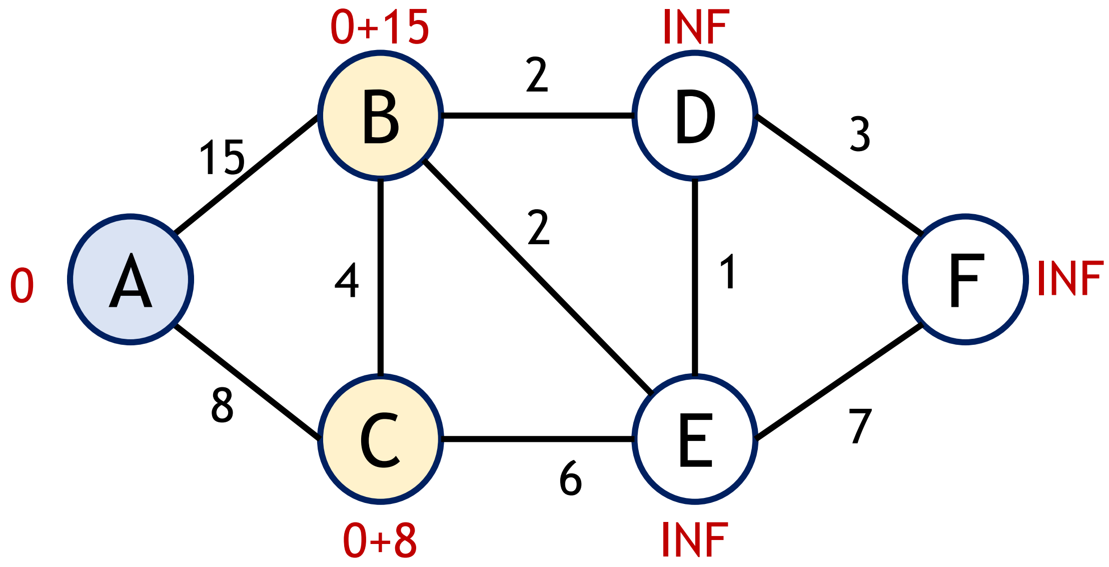
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	INF	INF	INF	INF	INF
PREV	-	-	-	-	-	-

A	B	C	D	E	F	



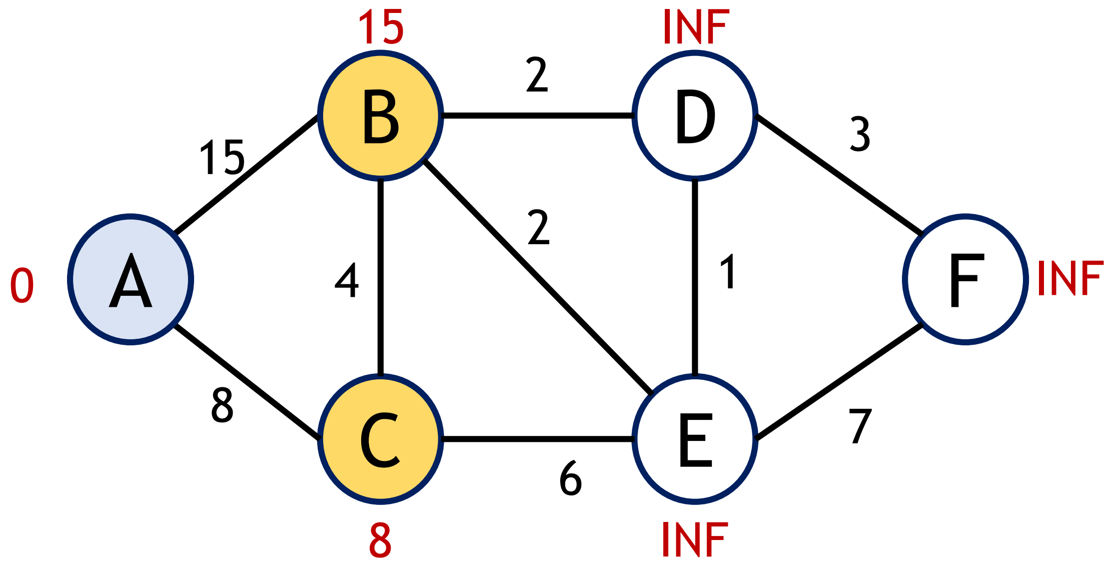
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	15	8	INF	INF	INF
PREV	-	A	A	-	-	-

A	B	C	D	E	F	



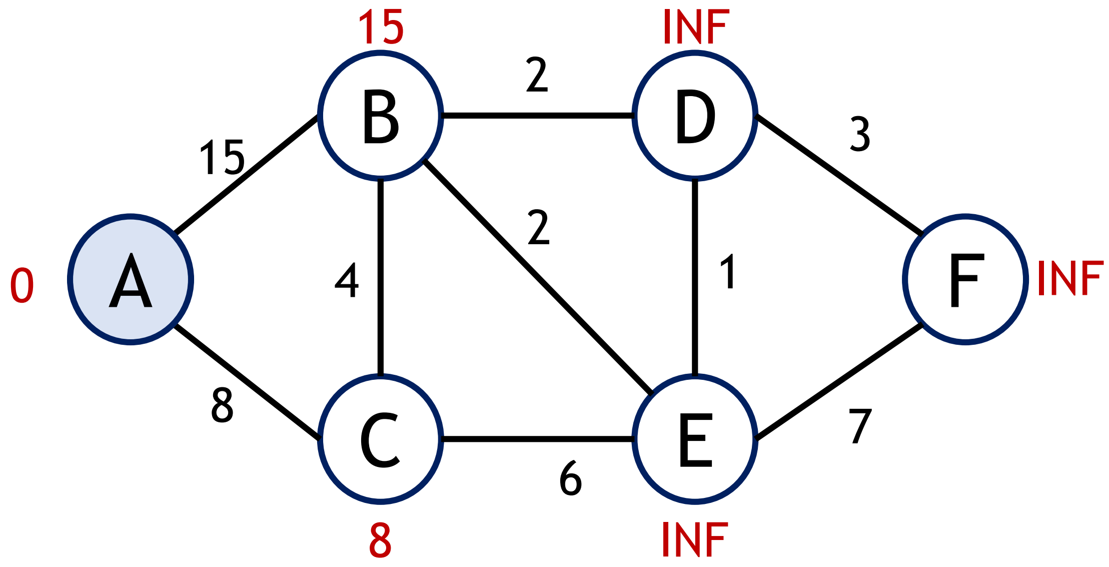
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	15	8	INF	INF	INF
PREV	-	A	A	-	-	-

A	B	C	D	E	F	



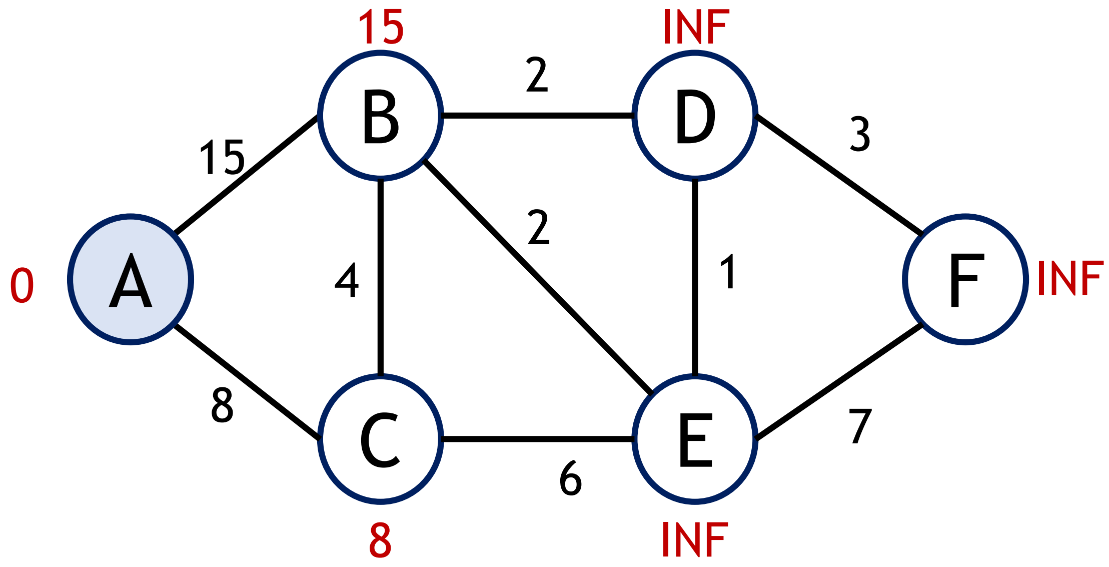
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C	D	E	F
DIST	0	15	8	INF	INF	INF
PREV	-	A	A	-	-	-

A	B	C	D	E	F	



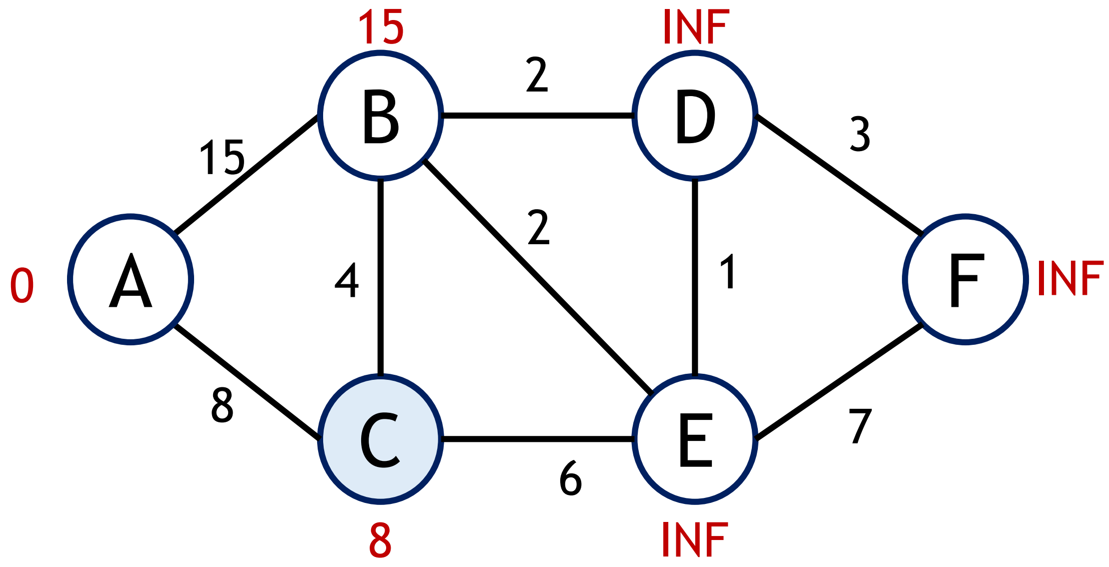
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C*	D	E	F
DIST	0	15	8	INF	INF	INF
PREV	-	A	A	-	-	-

C	B	D	E	F		



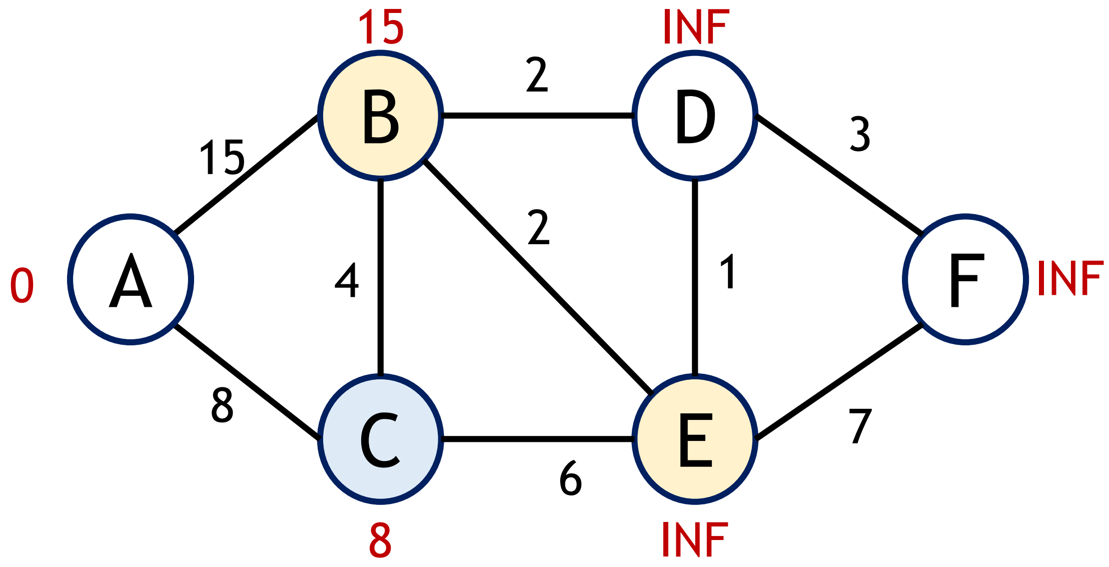
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C*	D	E	F
DIST	0	15	8	INF	INF	INF
PREV	-	A	A	-	-	-

C	B	D	E	F		



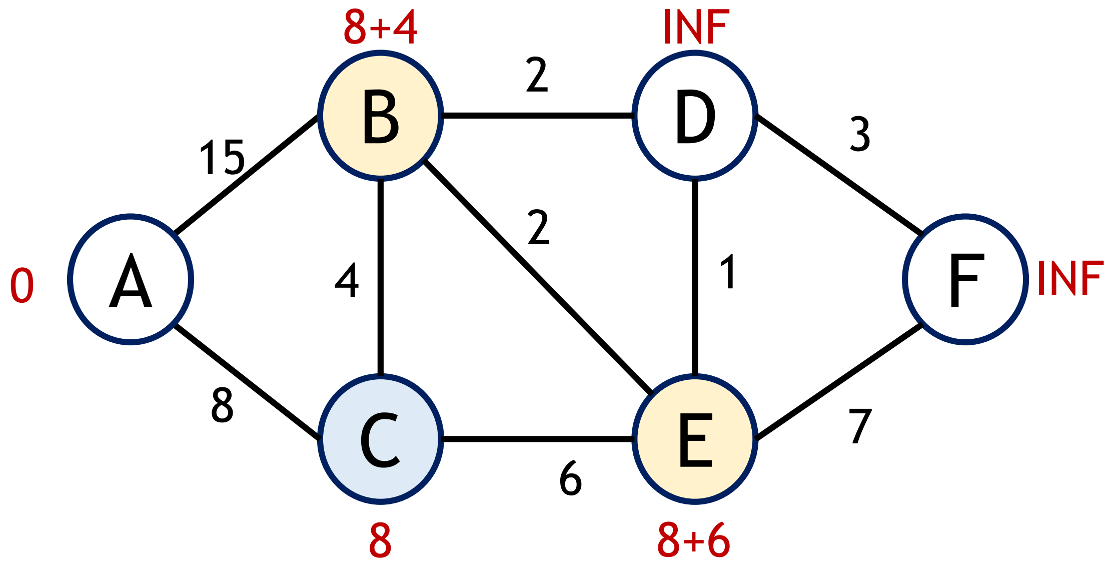
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C*	D	E	F
DIST	0	15	8	INF	INF	INF
PREV	-	A	A	-	-	-

C	B	D	E	F		



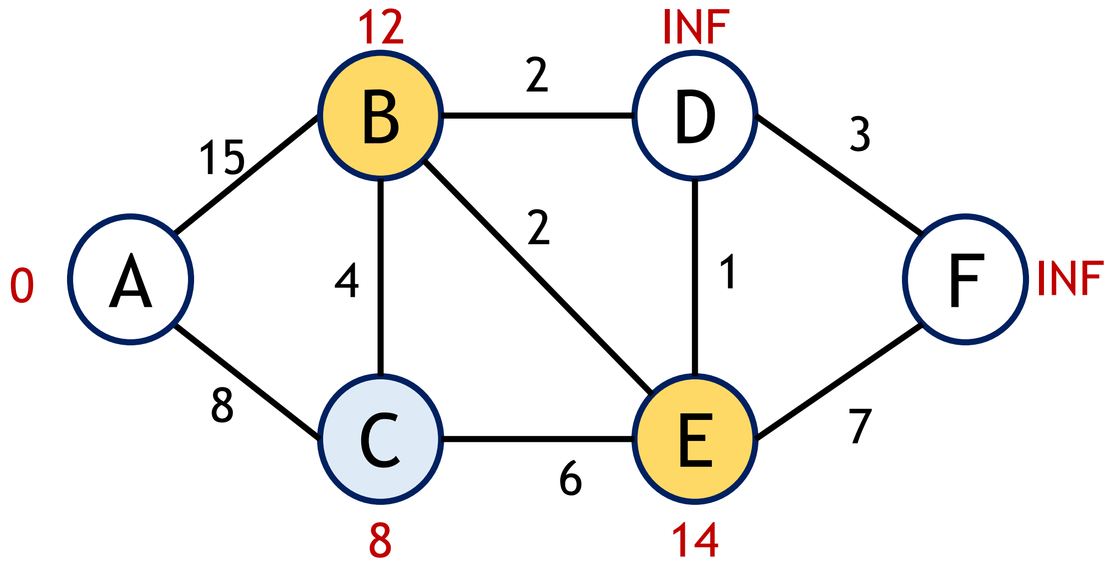
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C*	D	E	F
DIST	0	12	8	INF	14	INF
PREV	-	C	A	-	C	-

C	B	D	E	F		



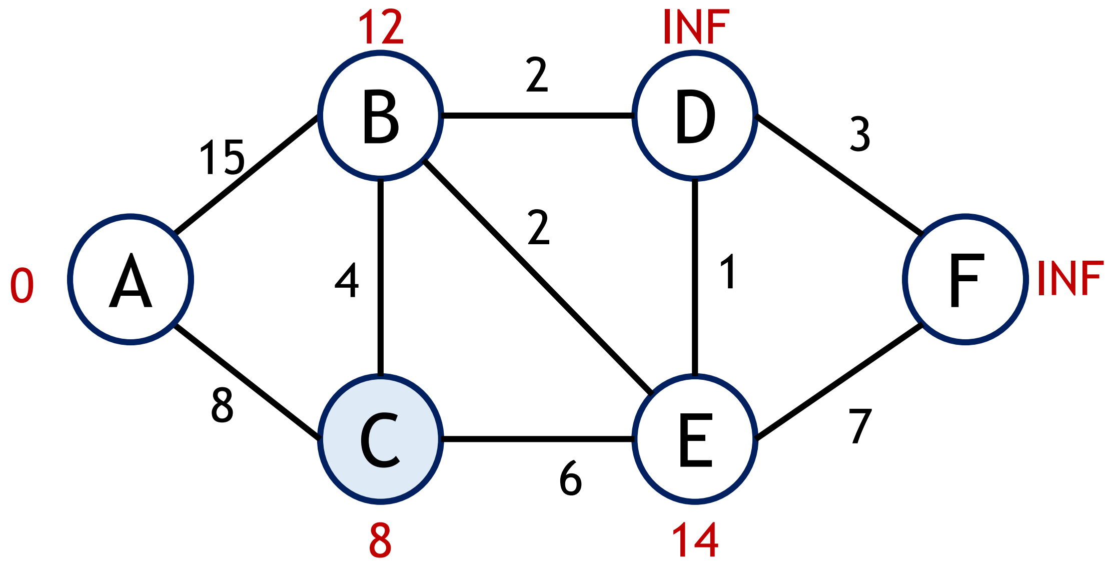
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B	C*	D	E	F
DIST	0	12	8	INF	14	INF
PREV	-	C	A	-	C	-

C	B	D	E	F		



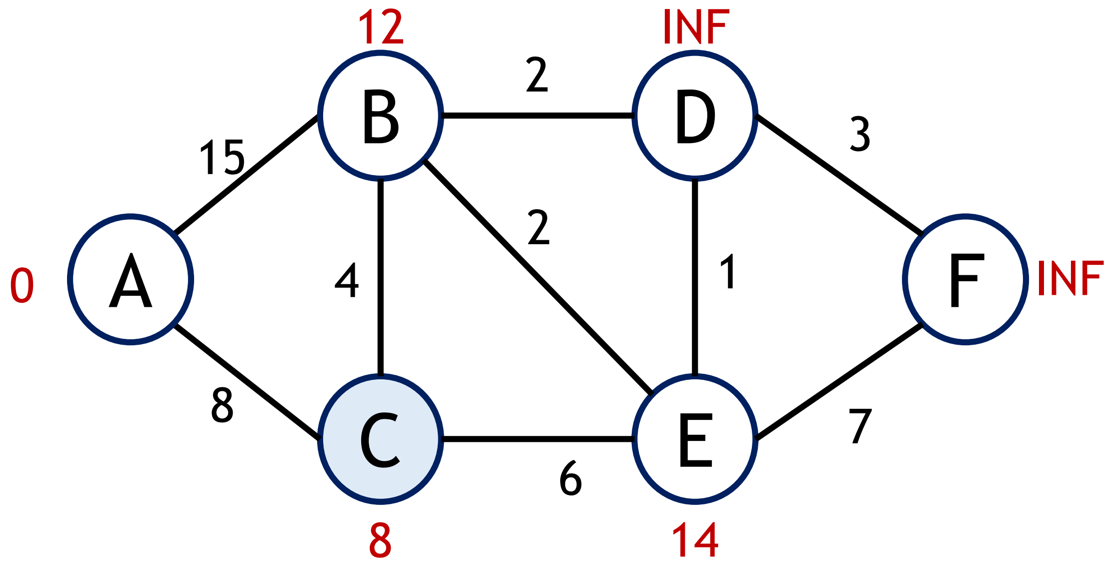
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	INF	14	INF
PREV	-	C	A	-	C	-

C	B	D	E	F		



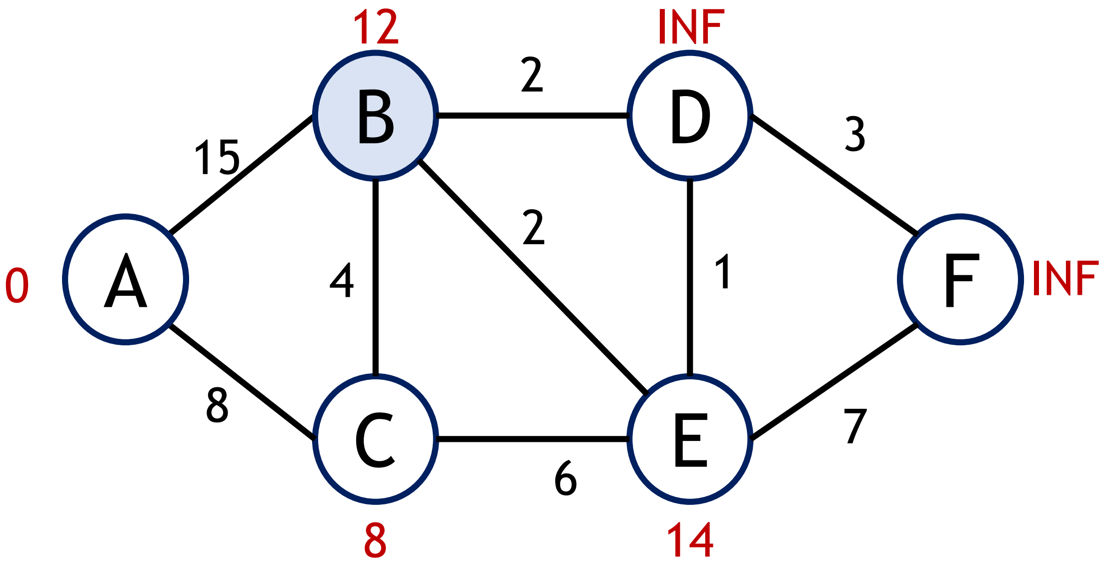
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	INF	14	INF
PREV	-	C	A	-	C	-

B	D	E	F			



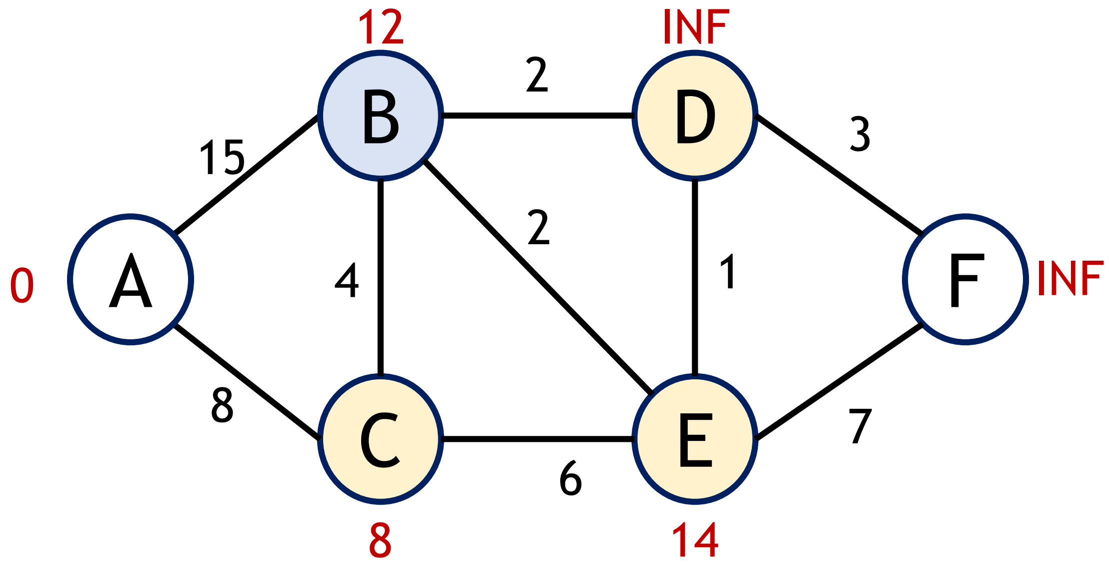
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	INF	14	INF
PREV	-	C	A	-	C	-

B	D	E	F			



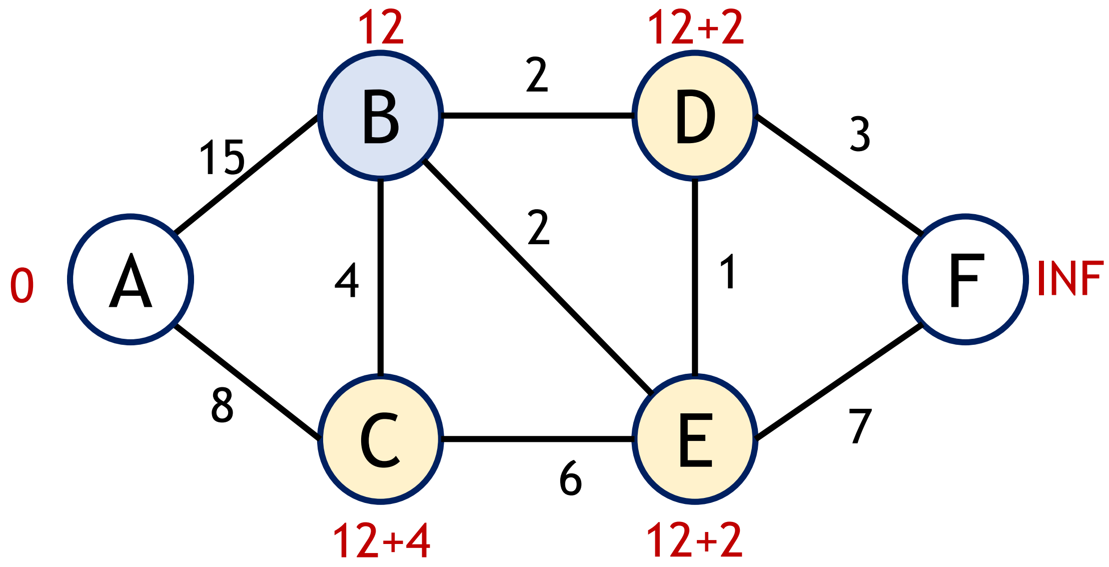
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	INF	14	INF
PREV	-	C	A	-	C	-

B	D	E	F			



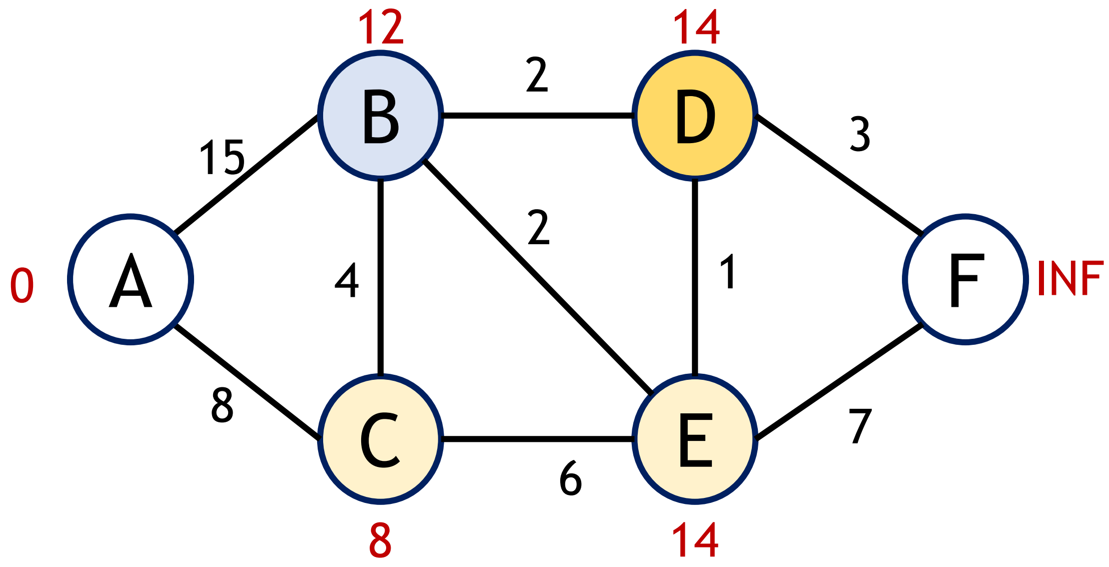
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	**14**	14	INF
PREV	-	C	A	**B**	C	-

B	D	E	F			



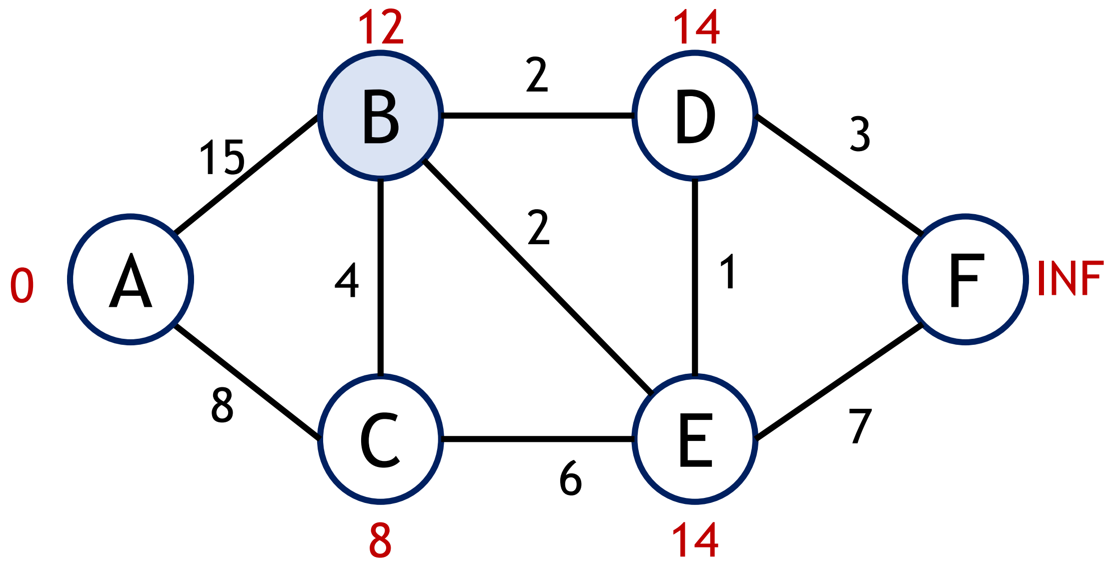
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	14	14	INF
PREV	-	C	A	B	C	-

B	D	E	F			



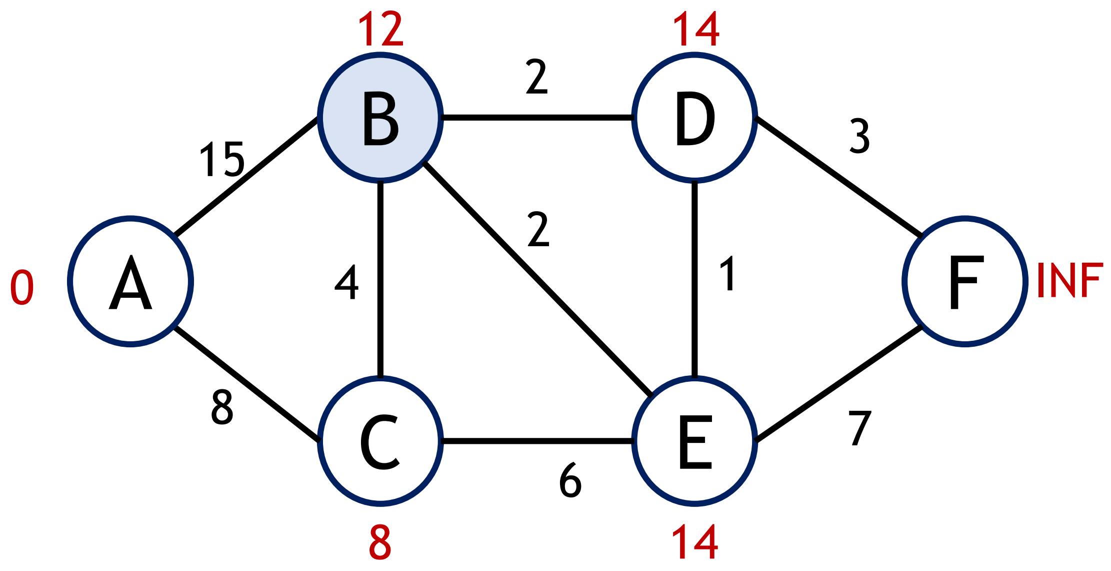
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E	F
DIST	0	12	8	14	14	INF
PREV	-	C	A	B	C	-

B	D	E	F			



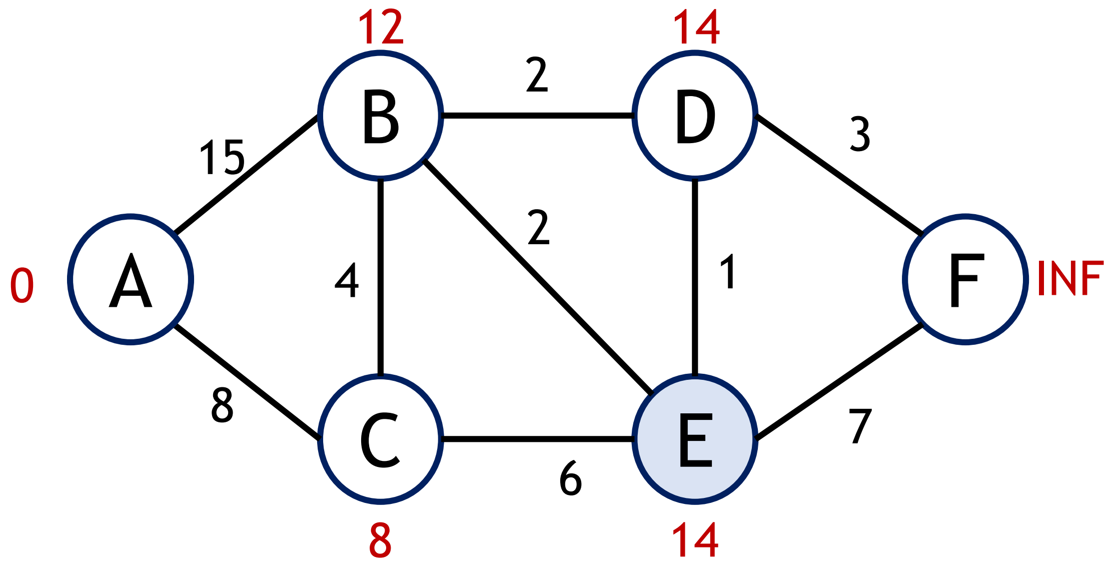
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E*	F
DIST	0	12	8	14	14	INF
PREV	-	C	A	B	C	-

E	D	F				



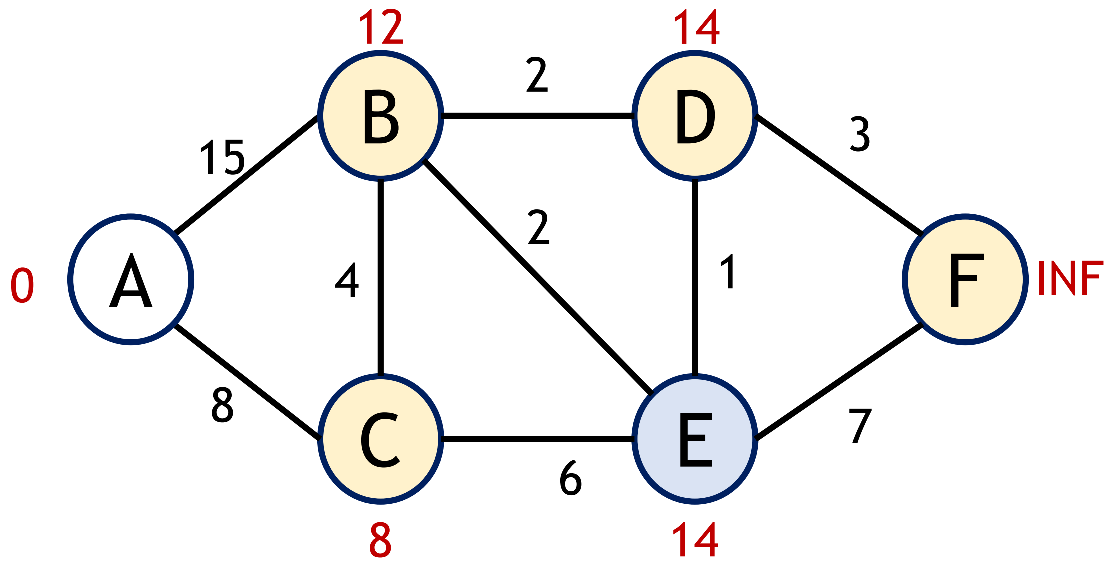
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E*	F
DIST	0	12	8	14	14	INF
PREV	-	C	A	B	C	-

E	D	F				



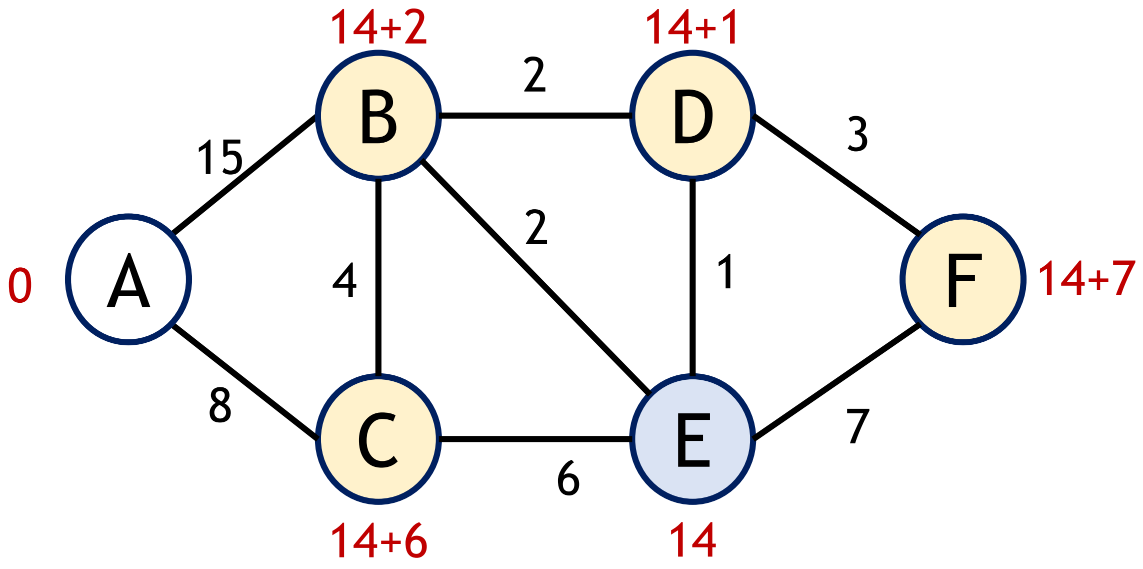
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E*	F
DIST	0	12	8	14	14	INF
PREV	-	C	A	B	C	-

E	D	F				



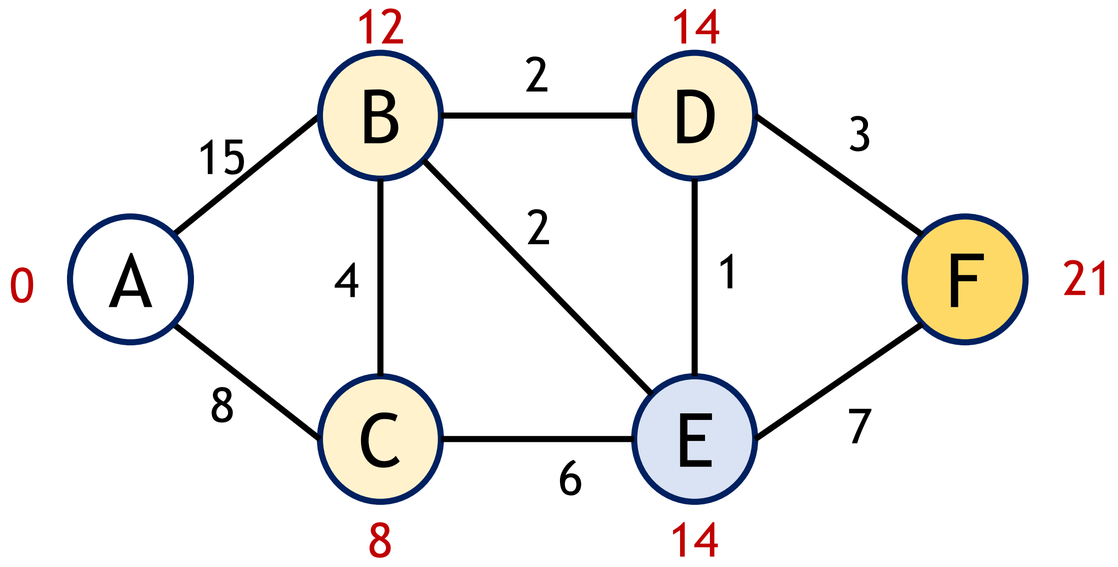
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E*	F
DIST	0	12	8	14	14	**21**
PREV	-	C	A	B	C	**E**

E	D	F				



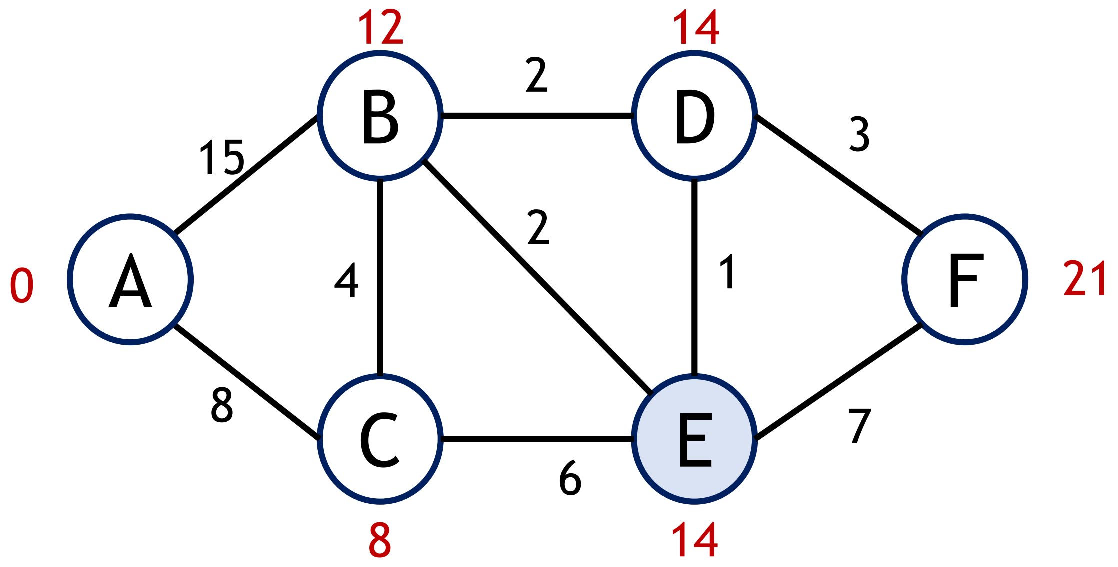
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E*	F
DIST	0	12	8	14	14	**21**
PREV	-	C	A	B	C	**E**

E	D	F				



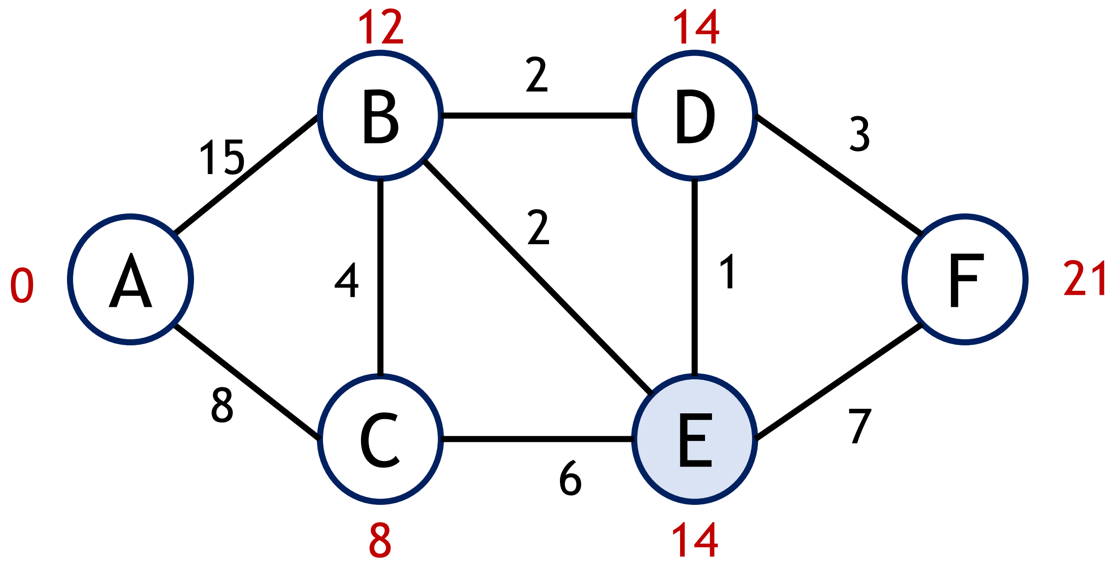
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D	E*	F
DIST	0	12	8	14	14	21
PREV	-	C	A	B	C	E

E	D	F				



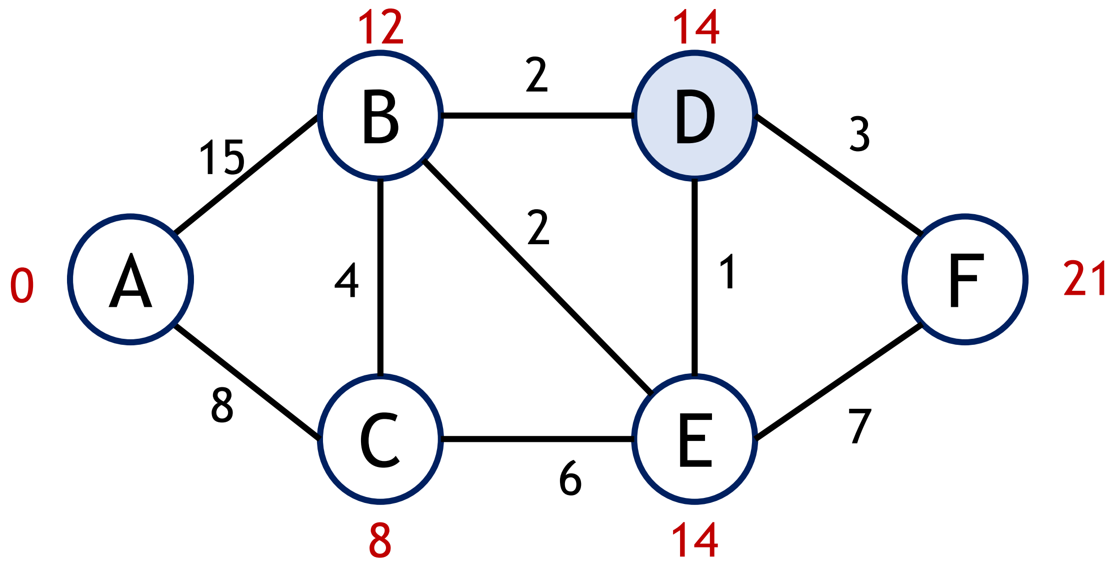
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F
DIST	0	12	8	14	14	**21**
PREV	-	C	A	B	C	**E**

D	F					



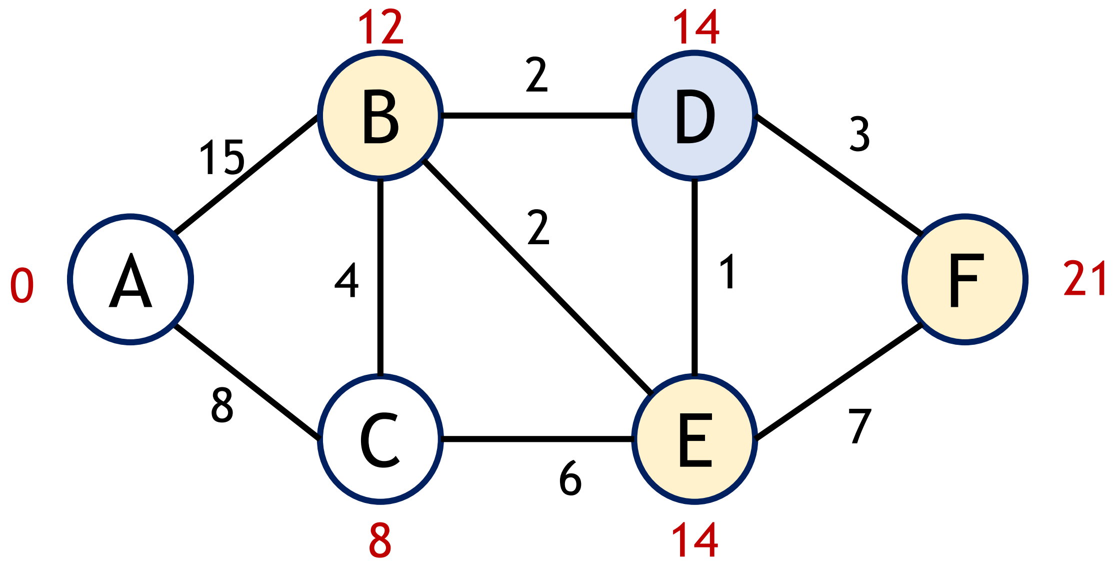
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F
DIST	0	12	8	14	14	21
PREV	-	C	A	B	C	E

D	F					



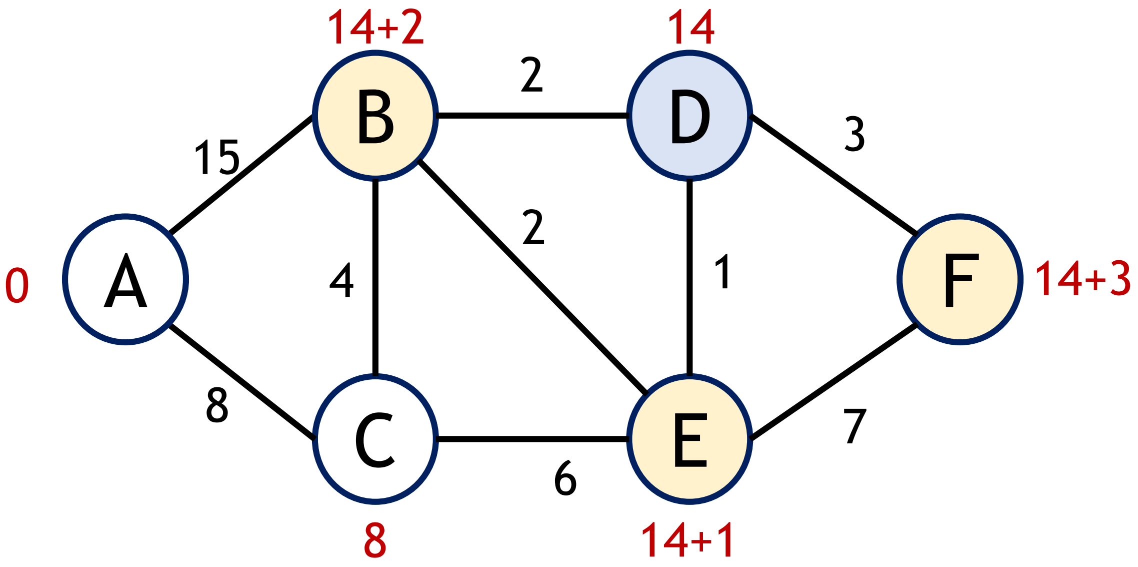
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F
DIST	0	12	8	14	14	21
PREV	-	C	A	B	C	E

D	F					



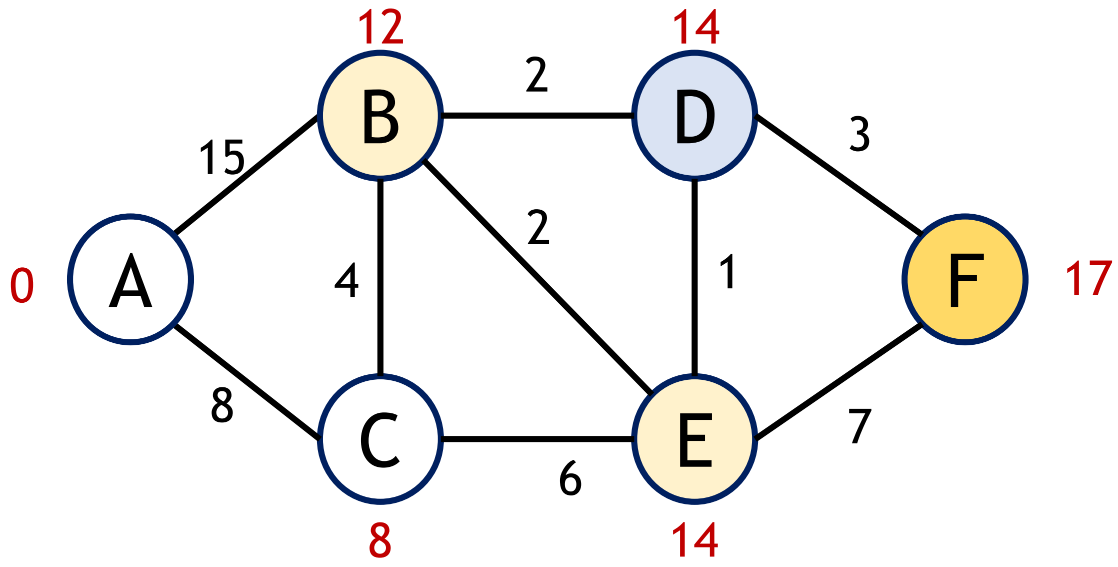
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F
DIST	0	12	8	14	14	**17**
PREV	-	C	A	B	C	**D**

D	F					



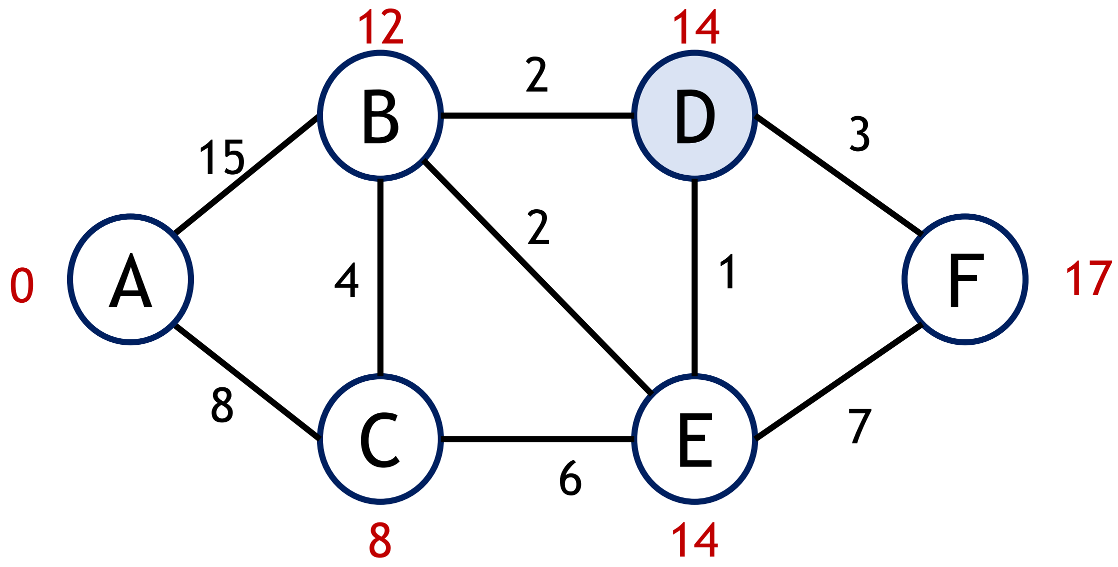
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

D	F					



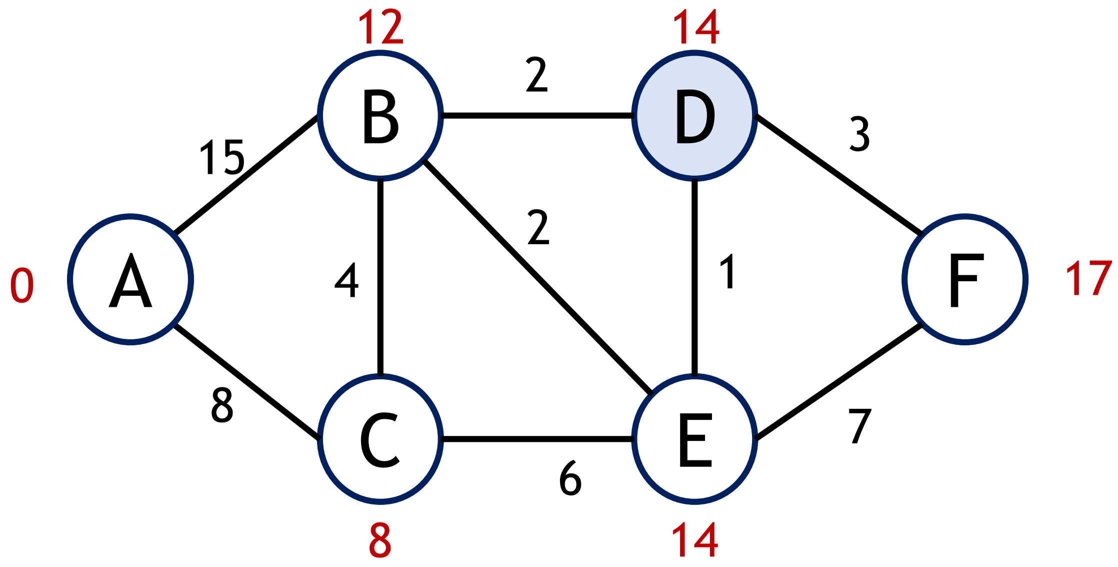
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

D	F					



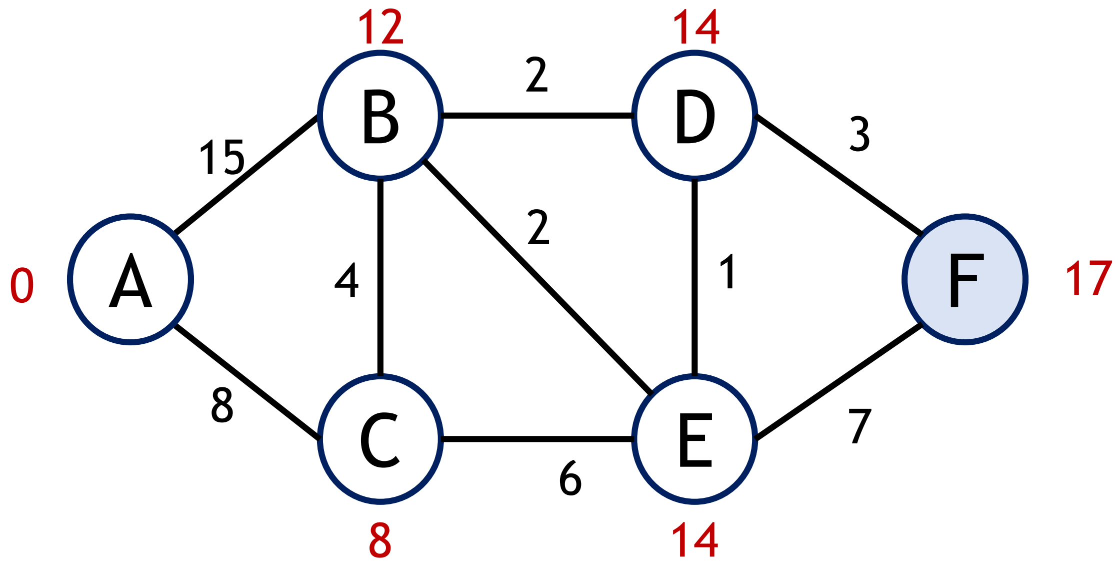
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

F						



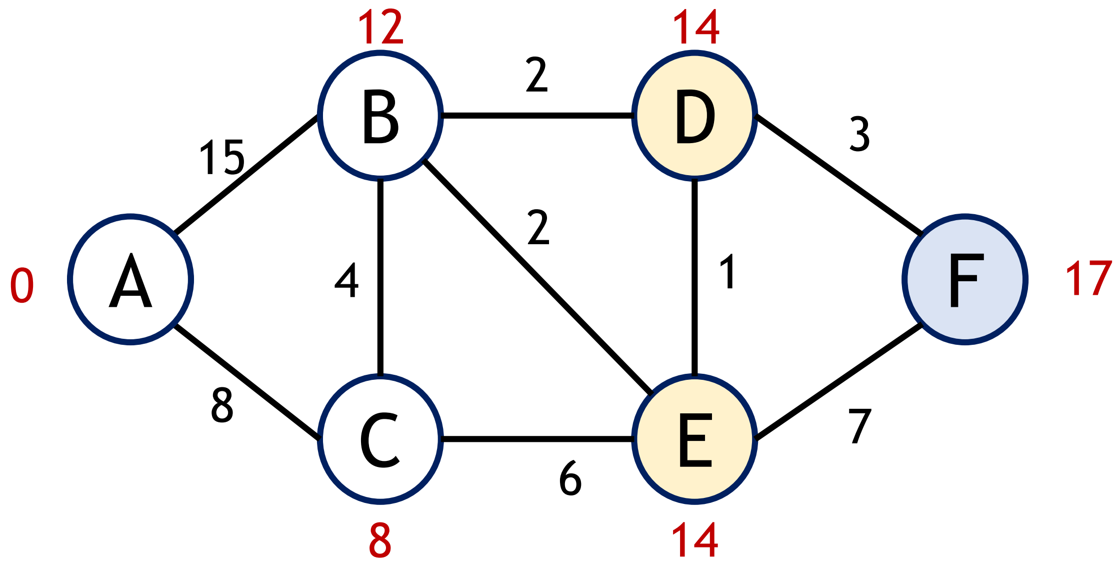
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

F						



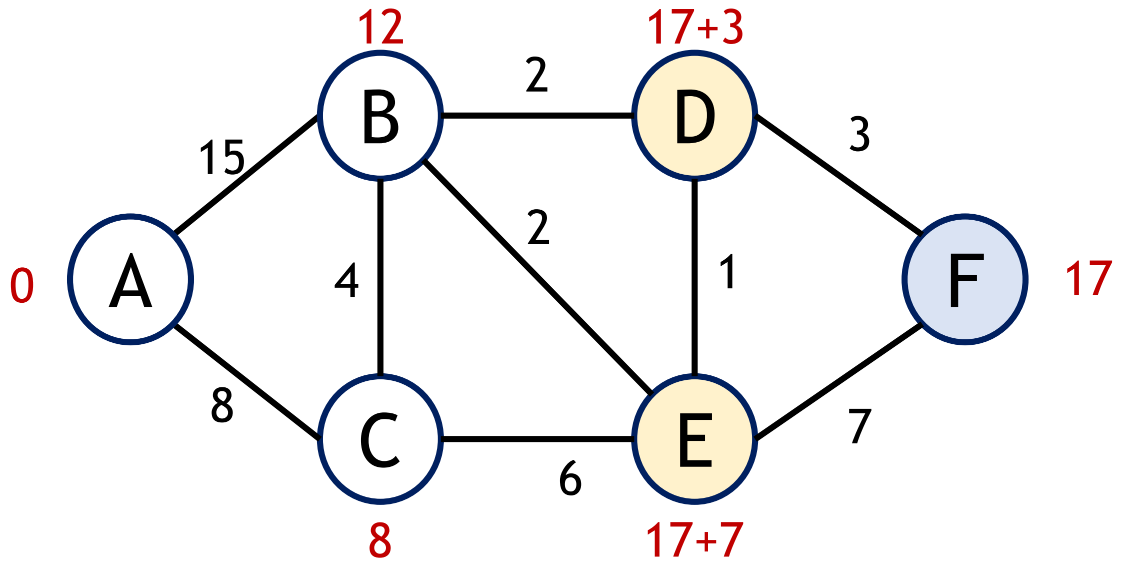
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

F						



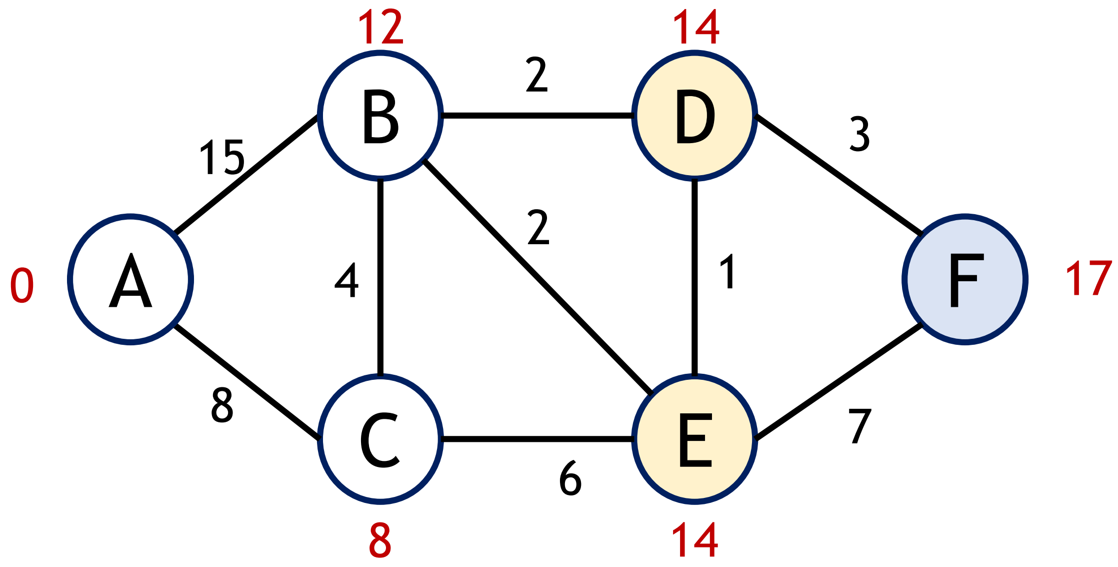
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

F						



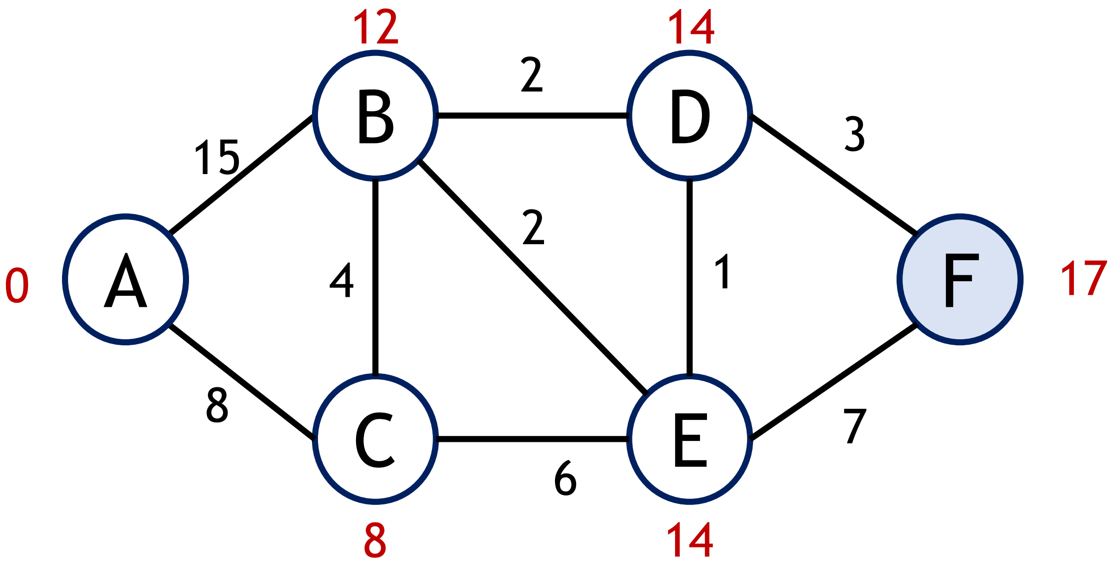
```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

F						

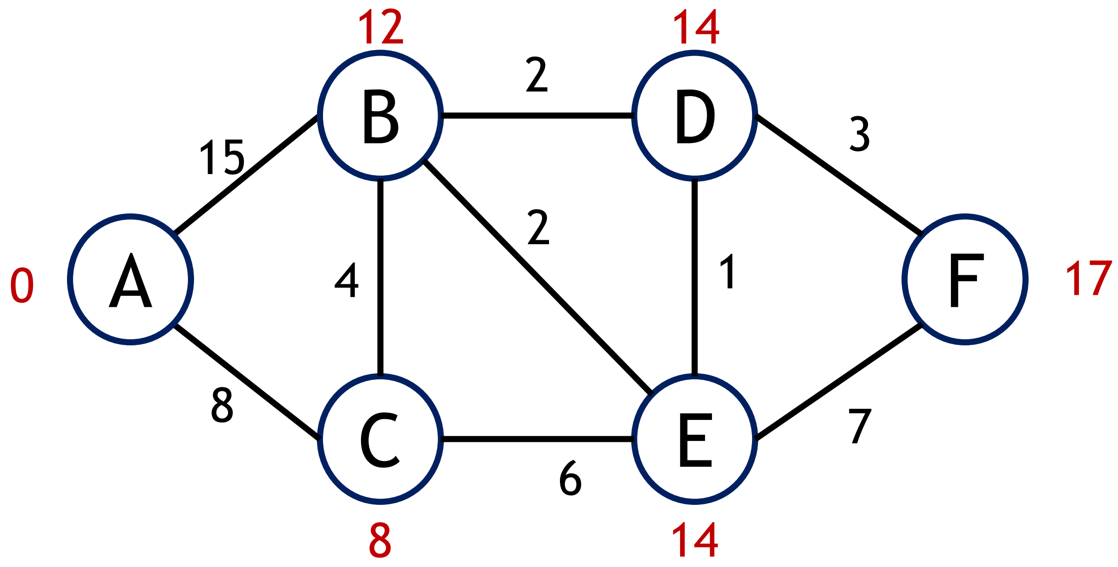


```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D

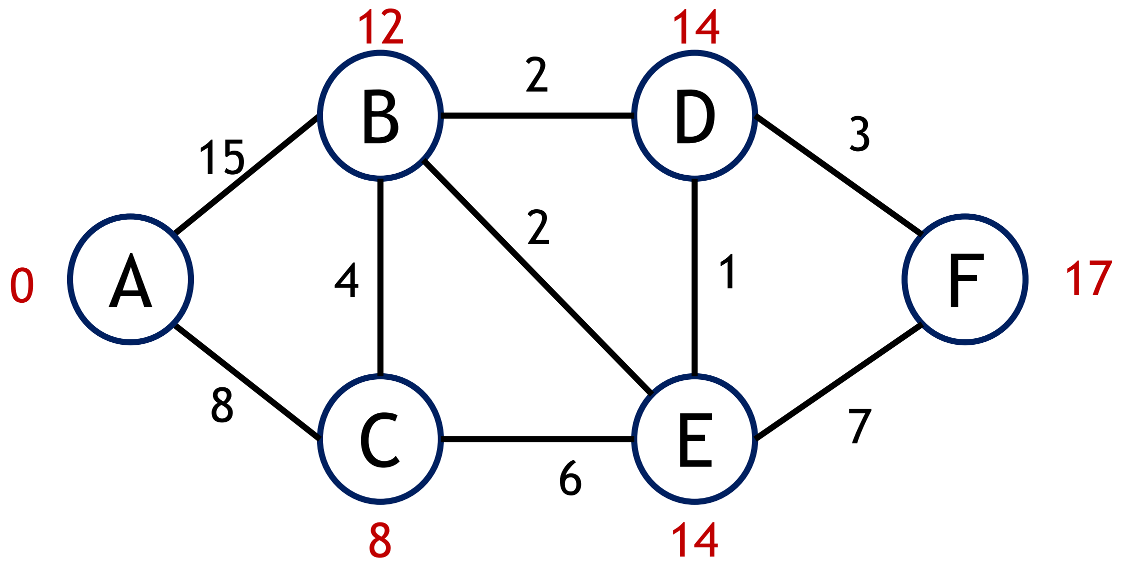


```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Dijkstra’s Shortest Path: Example

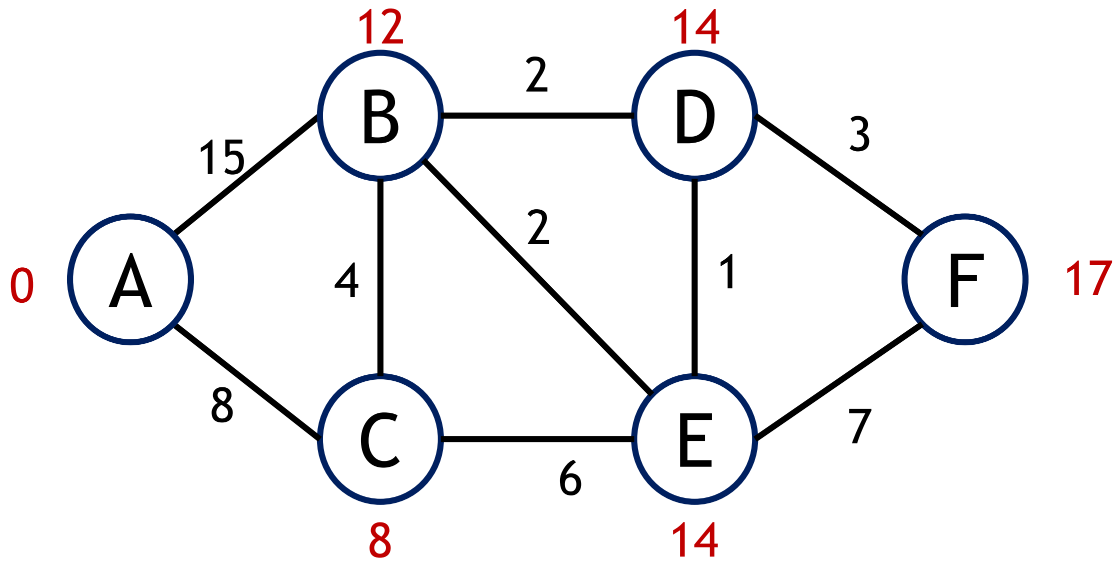
	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D



To	Path	d
A	-	0
B	A -> C -> B	12
C	A -> C	8
D	A -> C -> B -> D	14
E	A -> C -> E	14
F	A -> C -> B -> D -> F	17

Dijkstra’s Shortest Path: Example

	A*	B*	C*	D*	E*	F*
DIST	0	12	8	14	14	17
PREV	-	C	A	B	C	D



Trace Path from A -> F

Start at F

PREV(F) = D

PREV(D) = B

PREV(B) = C

PREV(C) = A

A, C, B, D, F

Dijkstra’s Shortest Path: Implementation

```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

```
findmin(Q, DIST) {
    ARG = 0; MIN = INF;
    for(i=0; i<Q.size(); i++) {
        if(DIST(Q[i]) < MIN) {
            MIN = DIST(Q[i]);
            ARG = i;
        }
    }
return Q[i];
}
```

O(|Q|) ≈ O(|V|)

Dijkstra’s Shortest Path: Complexity

iter # : `findmin()`, neighbors

```
1: |V| reps, |V|-1 nbrs
2: |V| - 1 reps, |V|-1 nbrs
3: |V| - 2 reps, |V|-1 nbrs
.
.
i: |V| - i – 1 reps, |V|-1 nbrs
.
.
|V|: 1 reps, |V|-1 nbrs
```

$$\boldsymbol{O}(\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{i}=|\boldsymbol{V}|}[|\boldsymbol{V}|-\boldsymbol{i}\ -\mathbf{1}]+[|\boldsymbol{V}|-\mathbf{1}])$$

$$O(\sum_{i=1}^{i=|V|}2|V|-i\ -2)$$

$$= O(2\sum_{i=1}^{i=|V|}|V|\ -2\sum_{i=1}^{i=|V|}1-\sum_{i=1}^{i=|V|}i)$$

$$= O\left(2|V|^2\ -2|V|-\frac{|V|(|V|+1)}{2}\right)$$

$$= O(|V|^2)$$

Dijkstra’s Shortest Path: Implementation

```
while !Q.isEmpty()
    u = vertex in Q w. min(DIST)
    remove u from Q
    for each v in G[u]
        T = DIST[u] + DIST[u,v]
        if T < DIST[v]
            DIST[v] = T
            PREV[v] = u

return DIST, PREV
```

Utilize Priority Queue, which is implemented using a Binary Heap (Min-Heap)

O(log(|V|))

References

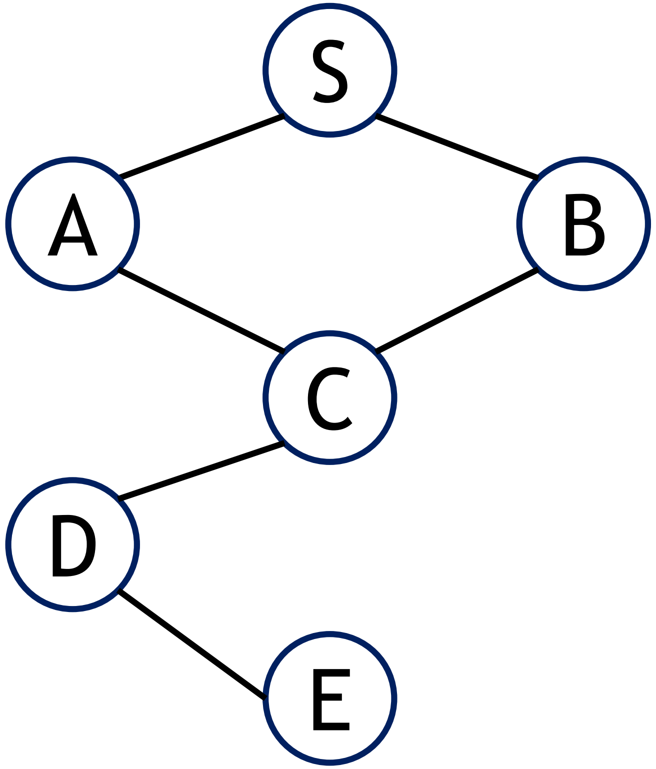
https://medium.com/basecs/finding-the-shortest-path-with-a-little-help-from-dijkstra-613149fbdc8e

https://www.hackerearth.com/practice/notes/dijkstras-algorithm/

Exercise: Identify if an edge is a bridge

What is a bridge?

If removing an edge increases the number of components of the graph, then that edge is a bridge.



Implement following subroutines:

`removeEdge`

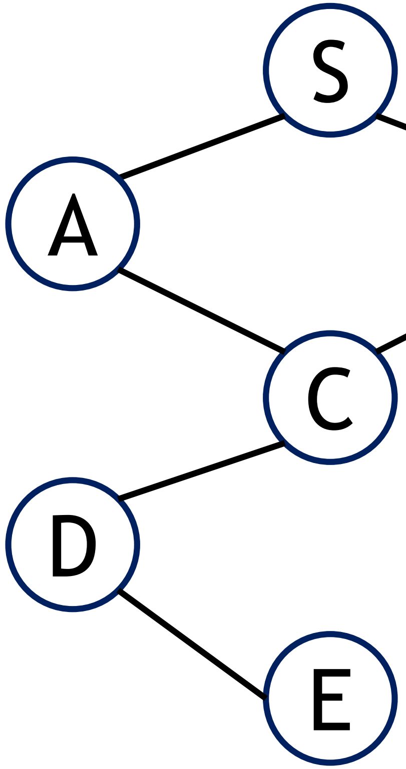
`DFTraversal`

`isBridge`

(1,6) is the only bridge

Exercise: Identify if an edge is a bridge

1. Pick Edge `e` you wish to check
2. Perform DFS/BFS, count components `ci`
3. tmp_edge = e;
4. Remove edge `e`
5. Perform BFS/DFS, count components `cf`
6. Restore Edge tmp_edge
7. if `cf > ci`: return True
8. return False



tmp_edge = (C, D)

ci = 1

cf = 2