DIJKSTRAS

Greedy Algorithm Paradigm

- **Characteristics of greedy algorithms:**
  - Make a sequence of choices
  - Each choice is the one that seems best so far, only depends on what's been done so far
  - Choice produces a smaller problem to be solved
- Optimal solution to the big problem contains optimal solutions to subproblems

Designing a Greedy Algorithm

- Cast the problem so that we make a greedy (locally optimal) choice and are left with one subproblem

- Prove there is always a (globally) optimal solution to the original problem that makes the greedy choice

- Show that the choice together with an optimal solution to the subproblem gives an optimal solution to the original problem

Some Greedy Algorithms

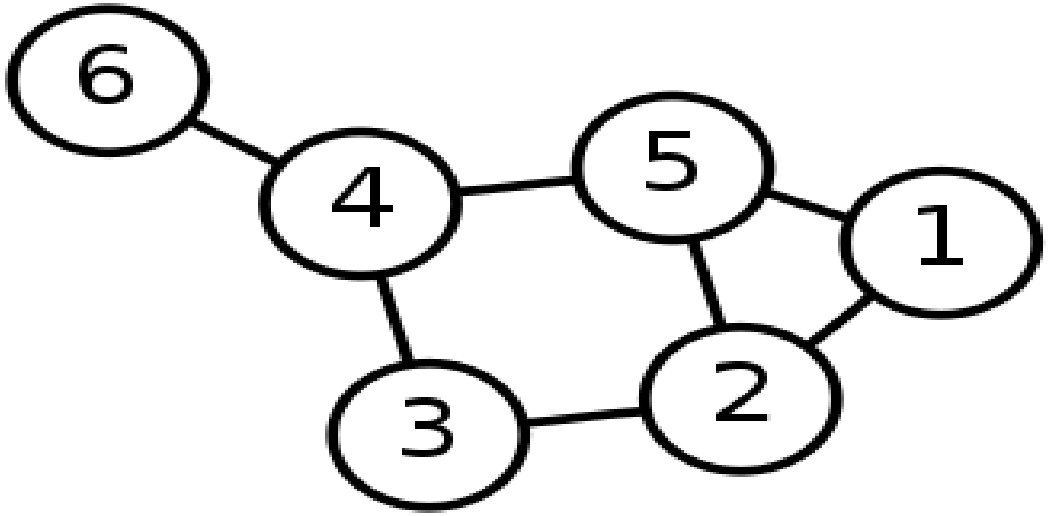
- Huffman codes
- Kruskal's MST algorithm
- Prim's MST algorithm
- Dijkstra's algorithm

DIJKSTRAS

From → A	B	C	D	E	F
A	25	35	∞	∞	∞
B	25	40	∞	∞	∞
C	25	40	∞	70	∞
E	25	40	80	70	∞
D	25	40	80	70	100

Single-Source Shortest Path Problem

<u>Single-Source Shortest Path Problem</u> - 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.

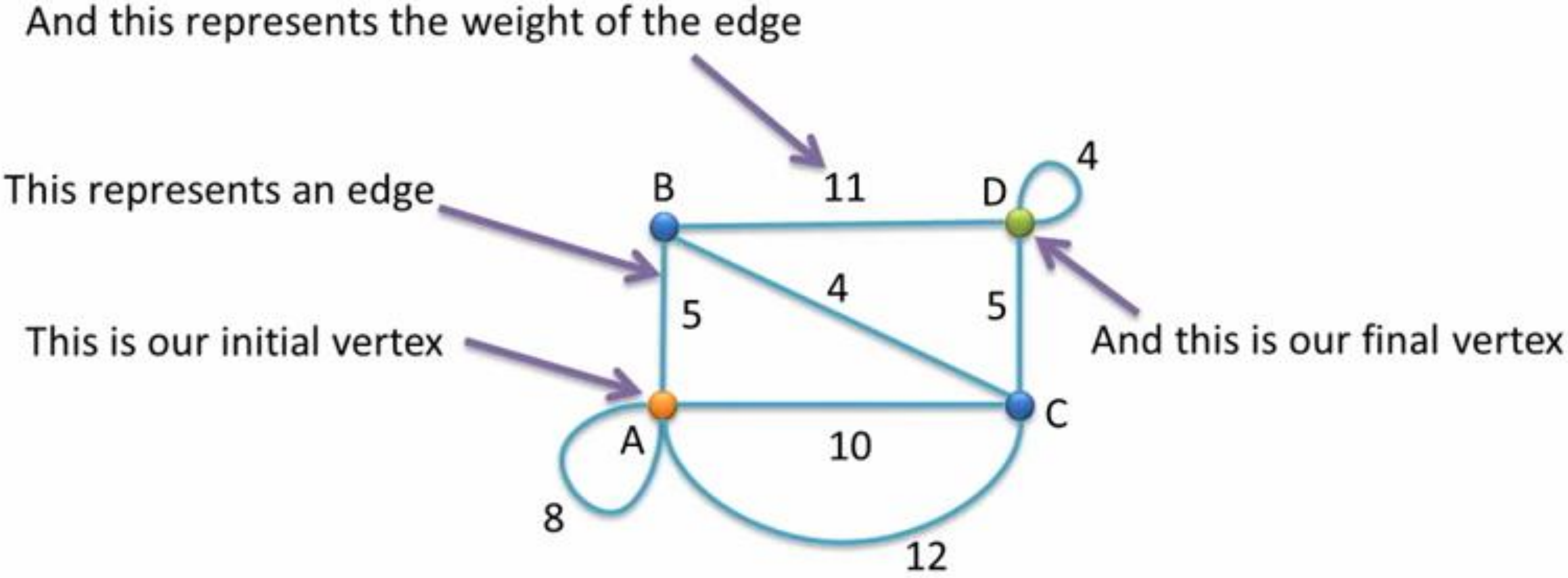
Works on both directed and undirected graphs. However, all edges must have nonnegative weights.

Approach: Greedy

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

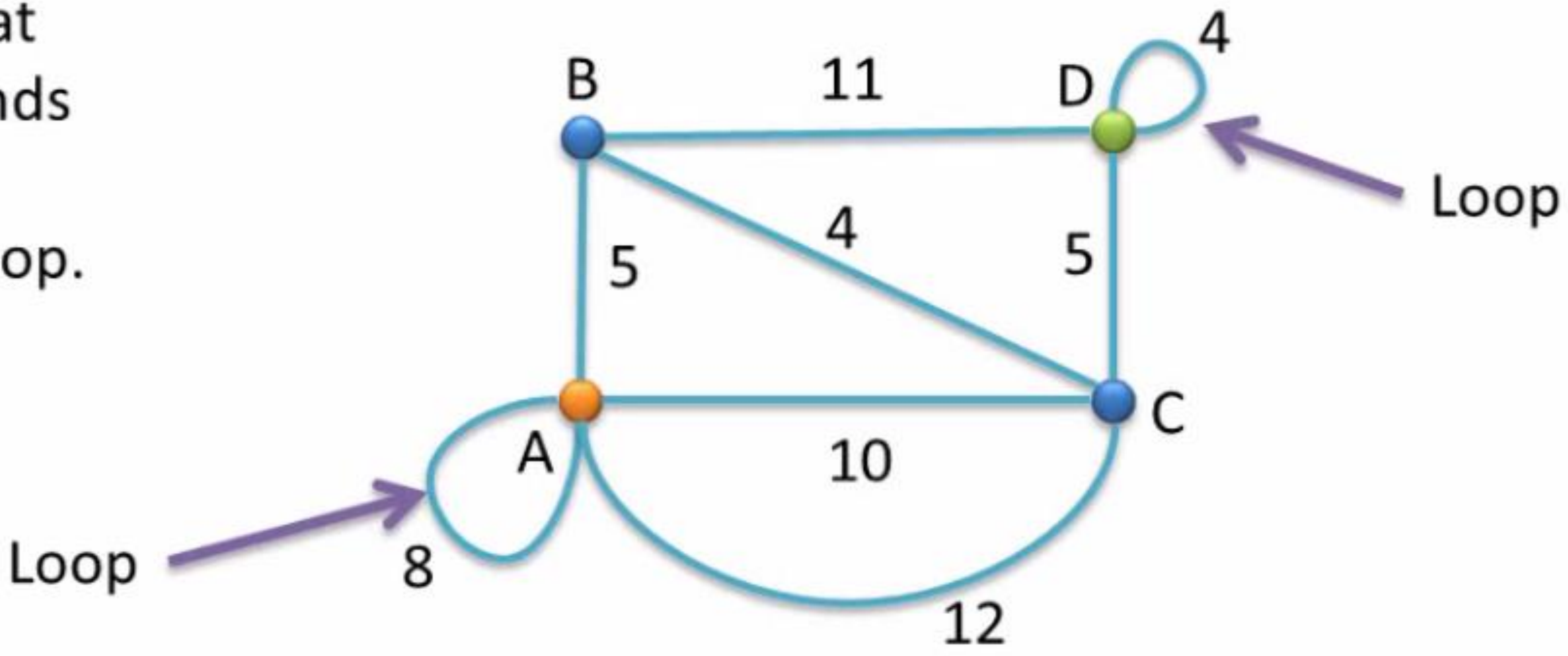
Output: Lengths of shortest paths from a given source vertex $v \in V$ to all other vertices

Here is our graph



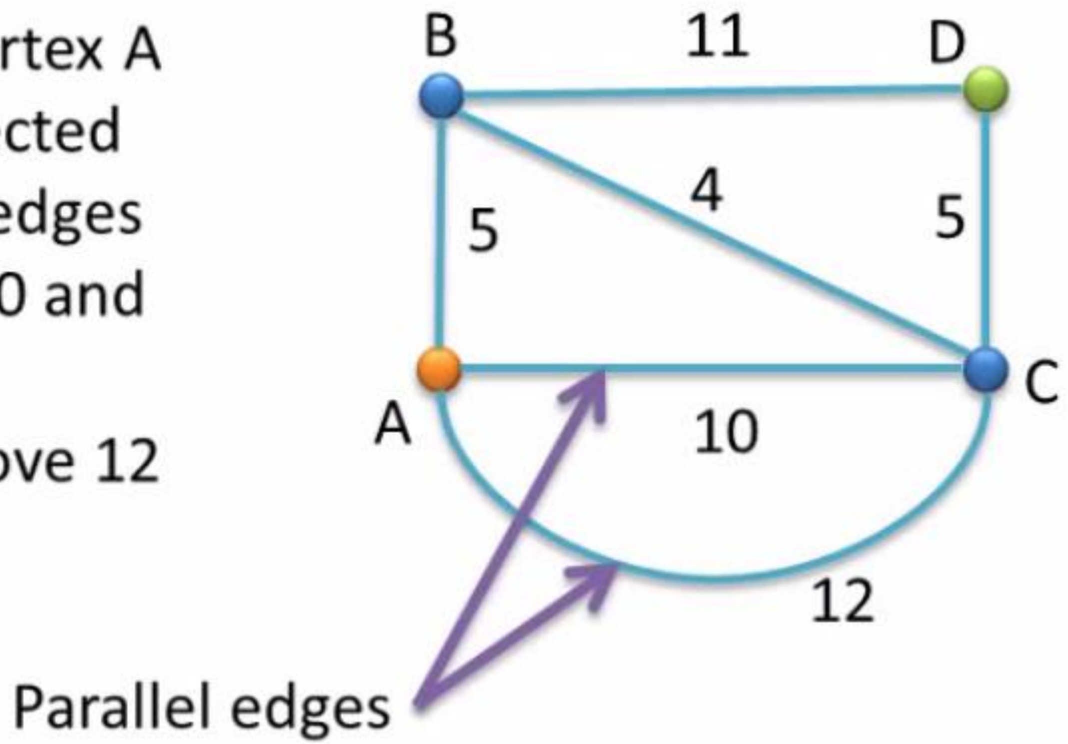
Step 1: Remove all the loops

Note!
Any edge that starts and ends at the same vertex is a loop.

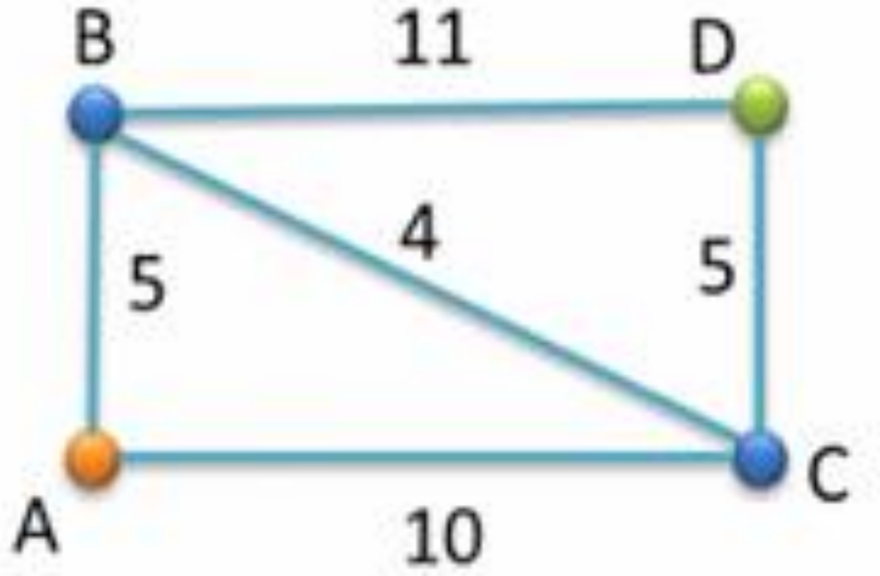


Step 2: Remove all parallel edges between two vertex except the one with least weight

Note!
In this graph, vertex A and C are connected by two parallel edges having weight 10 and 12 respectively.
So, we will remove 12 and keep 10.

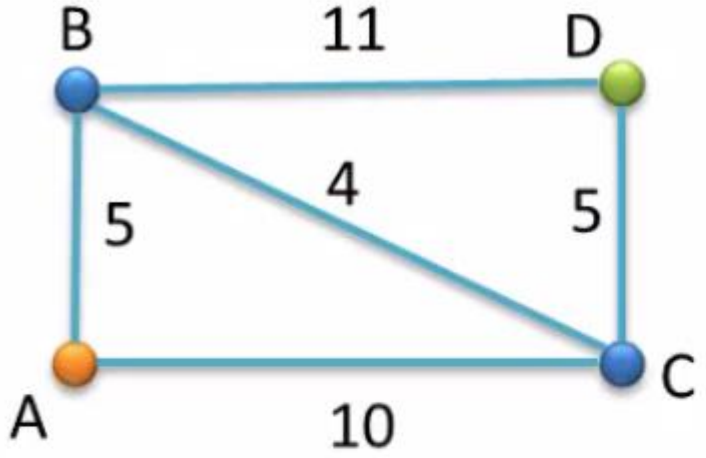


We are now ready to find the shortest path from vertex A to vertex D



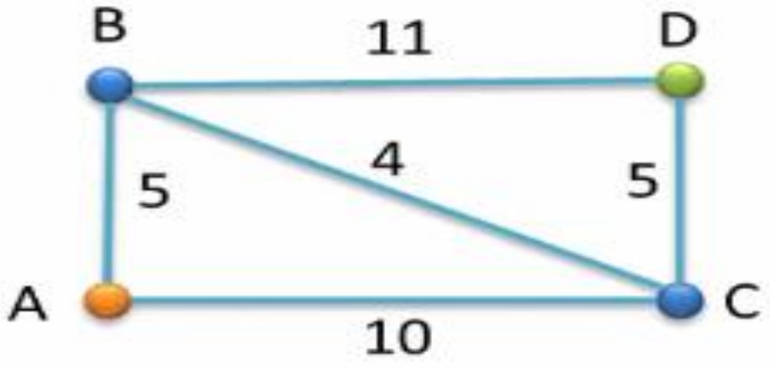
Step 3: Create shortest path table

As our graph has 4 vertices, so our table will have 4 columns



A	B	C	D

Note!
Column name is same as the name of the vertex.

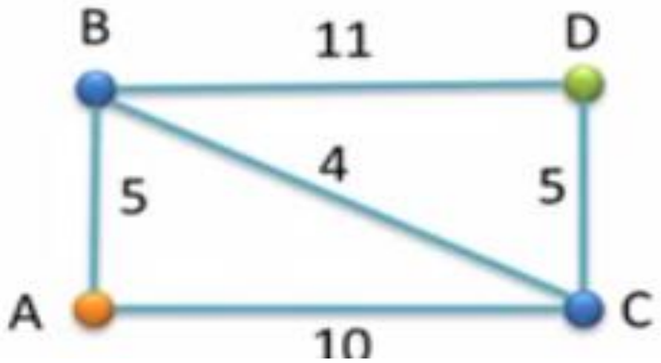


A	B	C	D
0	∞	∞	∞

Note!
We have written 0 in column A, as because vertex A is our source.

Now in the 1st row write 0 in column A and ∞ in other columns.

∞ denotes **Infinity**



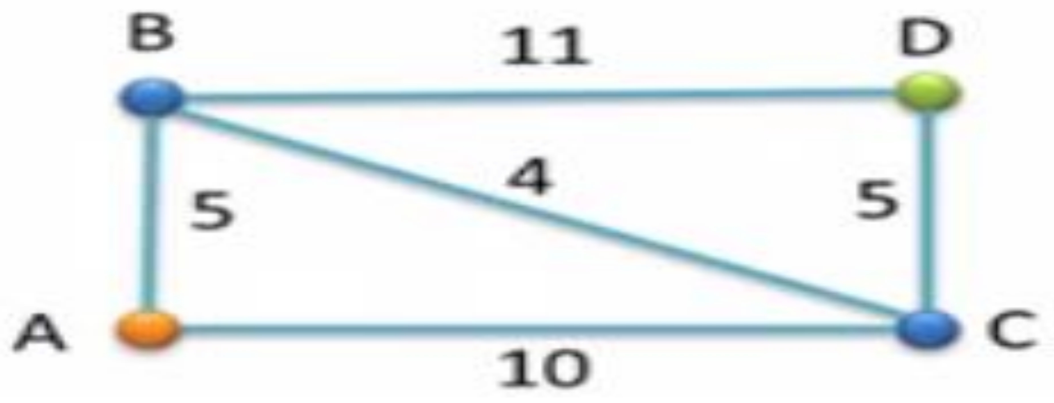
Value in the columns denote the weight of the shortest path.

So, 0 in column A denotes we are at vertex A and ∞ in all other columns denote unexplored vertices.

A	B	C	D
$\boxed{0}$	∞	∞	∞

Now that the 1st row is completely filled, our next job is to find the smallest unmarked value in the 1st row.

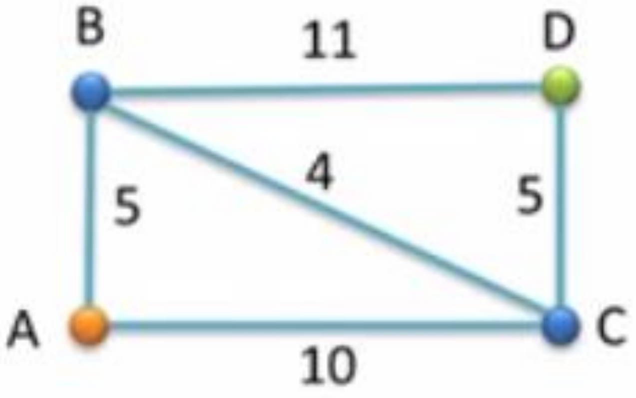
Looking at the table we can say that 0 is the smallest unmarked value in the 1st row. So we will mark it with a square box.



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$			

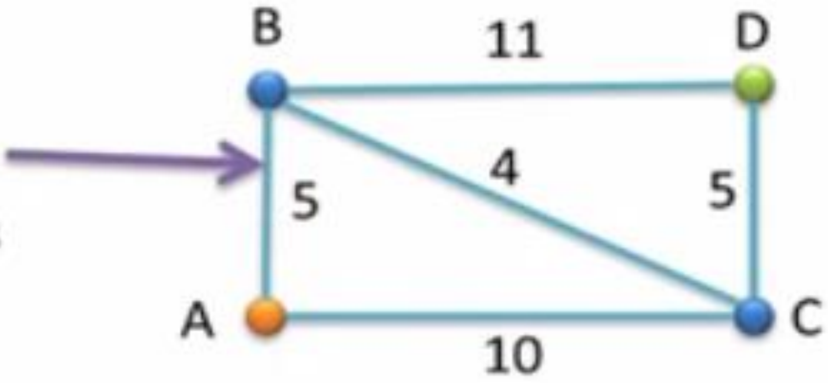
As column A was marked in the previous step, so we will now look for edges that are directly connected with vertex A.

Now draw another row and copy all the marked values in the new row.
In this case 0 is copied.



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$			

In this case we have an edge of weight 5 that directly connects A and B

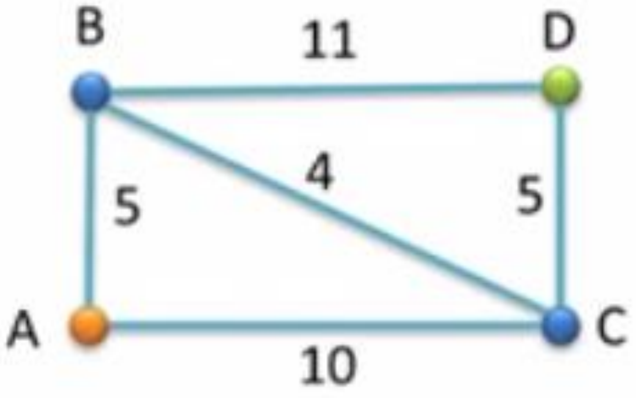


Find the edge that **directly** connects vertex A and vertex B

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$			

As column A was marked in the previous step, so we will now look for edges that are directly connected with vertex A.

Now draw another row and copy all the marked values in the new row.
In this case 0 is copied.



Now before we proceed further I want to tell you something about the minimum value formula that we are going to using in our calculation.

Minimum Value Formula

If we consider two vertices X (Source vertex) and Y (Destination vertex) and an edge that directly connects them.

Then we will have the following formula:

Min(DestValue, MarkedValue + EdgeWeight)

DestValue = The value in the destination vertex (i.e., Y) column.

MarkedValue = The value in the source vertex (i.e., X) column.

EdgeWeight = The weight of the edge that connects the source (i.e., X) and the destination (i.e., Y) vertex.

Minimum Value Formula

Solving:

Min(DestValue, MarkedValue + EdgeWeight)

We will get the minimum value that we will put in the destination vertex (i.e., Y) column.

For example:

If DestValue = 10, MarkedValue = 5 and EdgeWeight = 4

Putting the value we get

Min(10, 5+4)

= Min(10, 9)

= 9   As 9 is smaller than 10.

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$			

DestValue

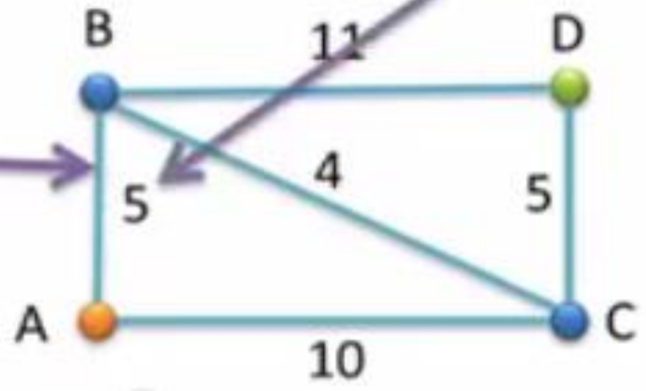
MarkedValue

As we are considering an edge between A and B so,
Source vertex = A
Destination vertex = B

EdgeWeight

Find the edge that **directly** connects vertex A and vertex B

In this case we have an edge of weight 5 that directly connects A and B



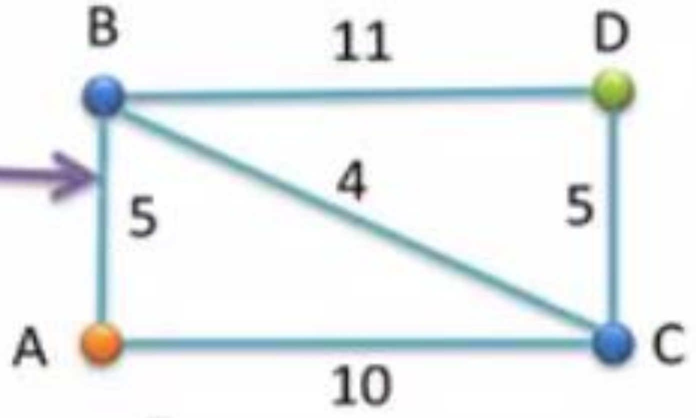
Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$	Min(∞, 0+5)		

So, we will write Min(∞, 0+5) in column B

Solving Min(∞, 0+5)
we get 5.
So, we will put the
value 5 in column B.

Find the edge that **directly** connects vertex A and vertex B

In this case we have an edge of weight 5 that directly connects A and B



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$	Min(∞, 0+5) 5		

DestValue

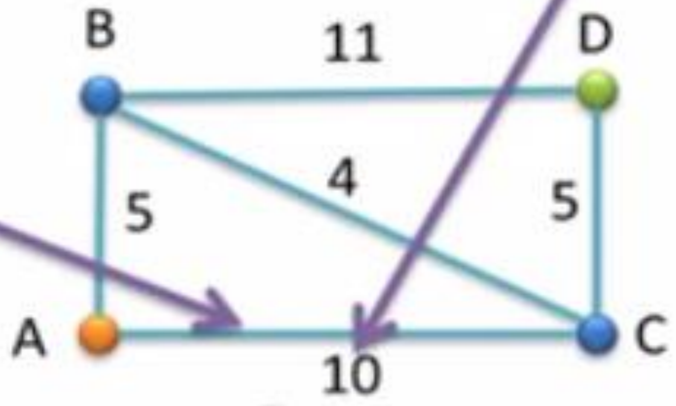
MarkedValue

As we are considering an edge between A and C so,
Source vertex = A
Destination vertex = C

EdgeWeight

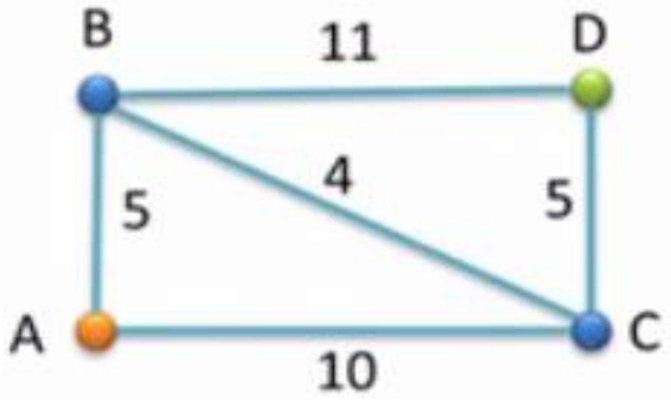
Now find the edge that **directly** connects vertex A and vertex C

In this case we have an edge of weight 10 that directly connects A and C



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
	$\boxed{0}$	Min(∞, 0+5) 5	Min(∞, 0+10) 10	∞

Looking at our graph we find no such edge that directly connects vertex A and D so we will simply copy the previous value in column D i.e., ∞

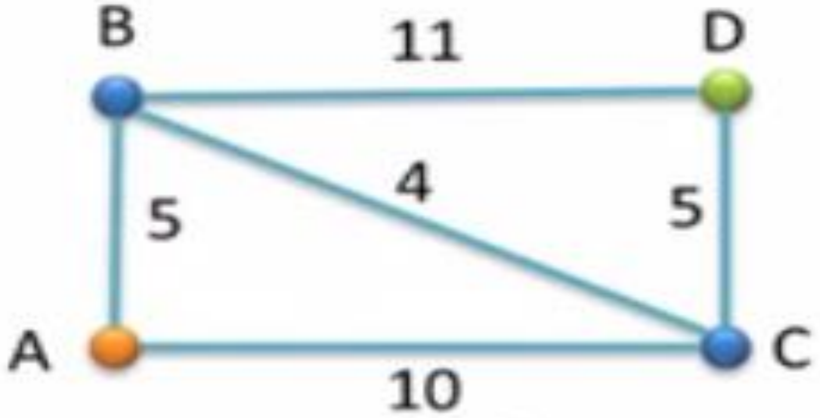


Now find the edge that **directly** connects vertex A and vertex D

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞

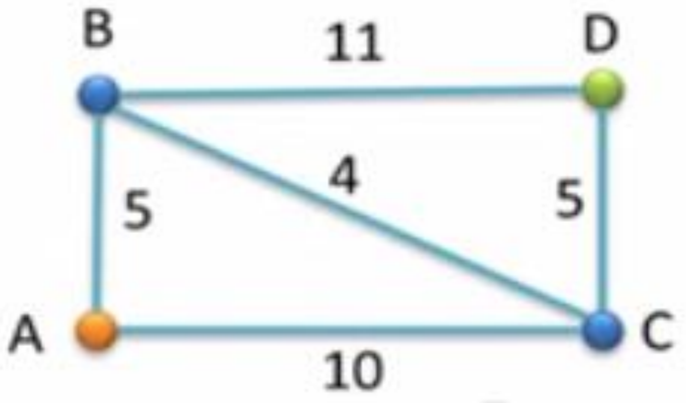
Now that the 2nd row is completely filled, our next job is to find the smallest unmarked value in the 2nd row.

Looking at the table we can say that 5 is the smallest unmarked value in the 2nd row. So we will mark it with a square box.



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
	$\boxed{0}$	$\boxed{5}$		

As column B was marked in the previous step, so we will now look for edges that are directly connected with vertex B. Note! We don't have to consider vertex A as it is already marked.



Now draw another row and copy all the marked values in the new row.
In this case 0 and 5 are copied.

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
	$\boxed{0}$	$\boxed{5}$		

MarkedValue

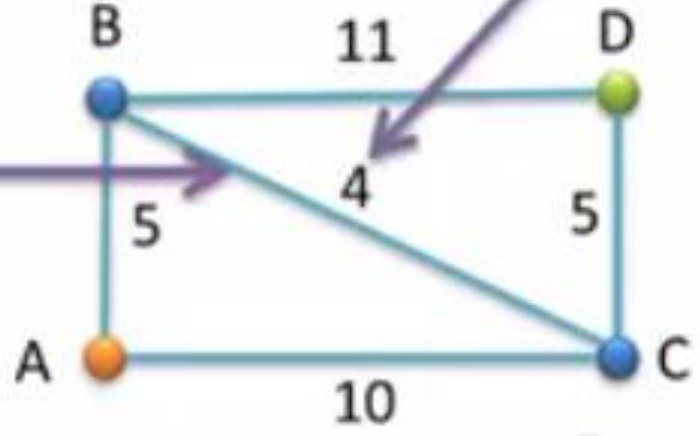
DestValue

As we are considering an edge between B and C so,
Source vertex = B
Destination vertex = C

EdgeWeight

Find the edge that **directly** connects vertex B and vertex C

In this case we have an edge of weight 4 that directly connects B and C

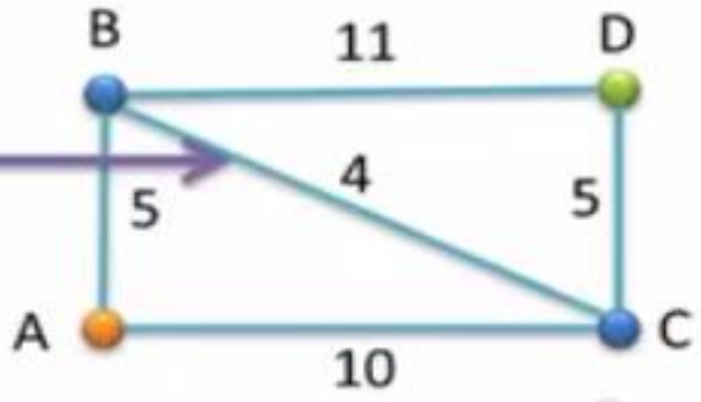


Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
	$\boxed{0}$	$\boxed{5}$		

So, we will write Min(10, 5+4) in column C.
Solving it we get 9.

Find the edge that **directly** connects vertex B and vertex C

In this case we have an edge of weight 4 that directly connects B and C



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) 9	

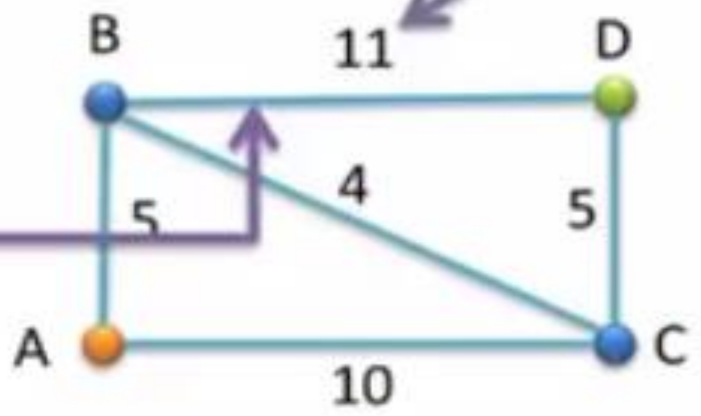
DestValue

As we are considering an edge between B and D so,
Source vertex = B
Destination vertex = D

MarkedValue

EdgeWeight

Now find the edge that **directly** connects vertex B and vertex D



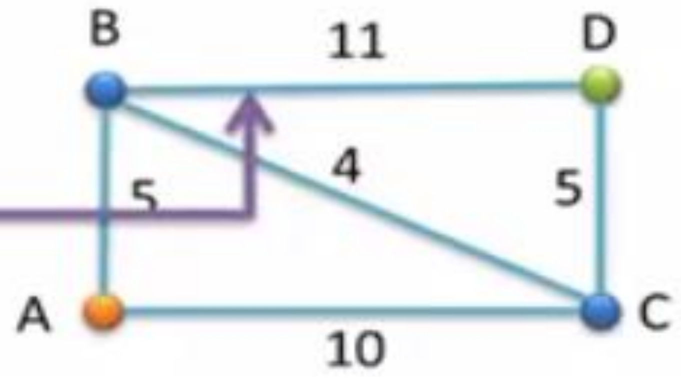
In this case we have an edge of weight 11 that directly connects B and D

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) 9	Min(∞, 5+11) 16

So, we will write Min(∞, 5+11) in column D.
Solving it we get 16.

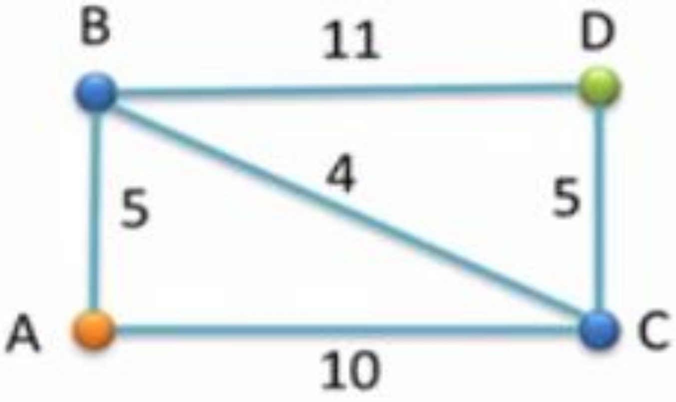
Now find the edge that **directly** connects vertex B and vertex D

In this case we have an edge of weight 11 that directly connects B and D



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16

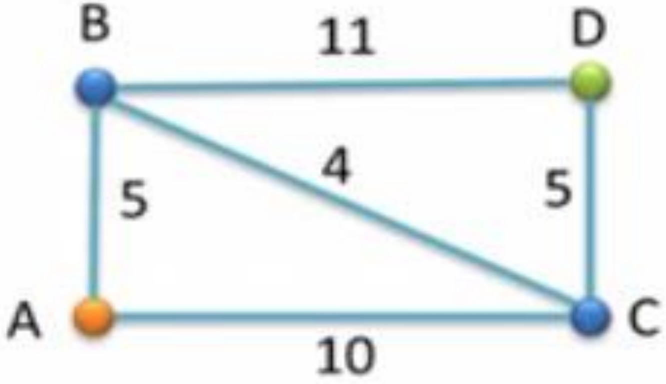
Now that the 3rd row is completely filled, our next job is to find the smallest unmarked value in the 3rd row.



Looking at the table we can say that 9 is the smallest unmarked value in the 3rd row.
So we will mark it with a square box.

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	

As column C was marked in the previous step, so we will now look for edges that are directly connected with vertex C.
Note! We don't have to consider vertex A and B as they are already marked.



Now draw another row and copy all the marked values in the new row.
In this case 0, 5 and 9 are copied.

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	

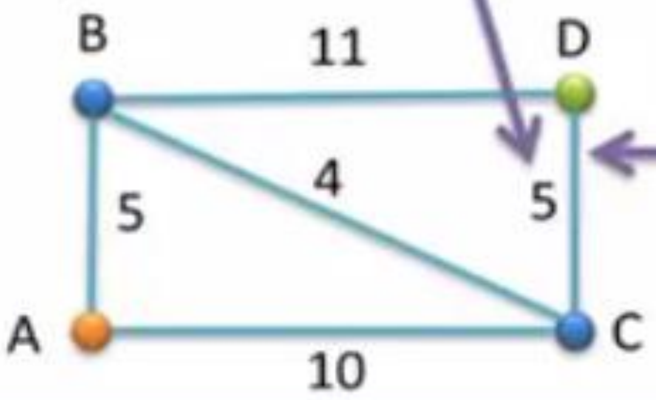
DestValue

MarkedValue

Find the edge that **directly** connects vertex C and vertex D

EdgeWeight

As we are considering an edge between C and D so,
Source vertex = C
Destination vertex = D

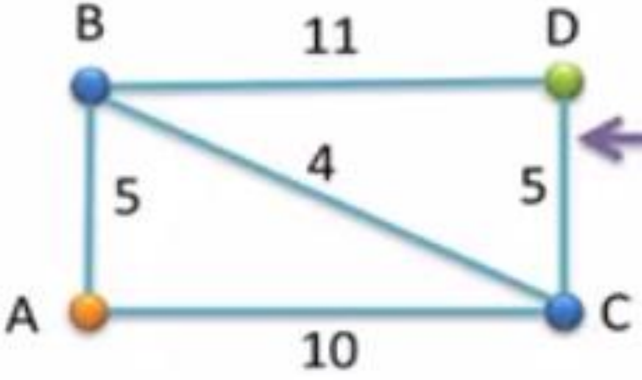


In this case we have an edge of weight 5 that directly connects C and D

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) 14

So, we will write Min(16, 9+5) in column D. Solving it we get 14.

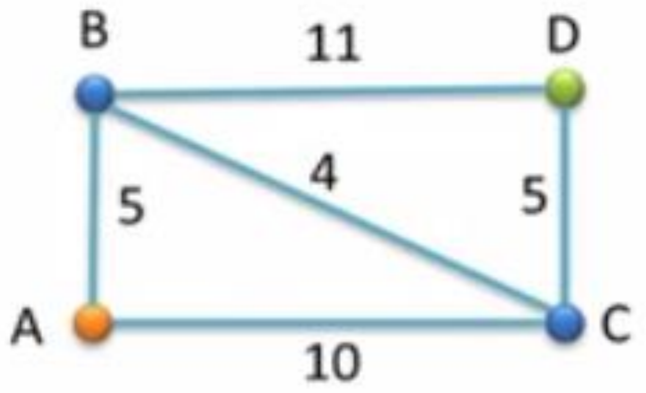
Find the edge that **directly** connects vertex C and vertex D



In this case we have an edge of weight 5 that directly connects C and D

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) 14

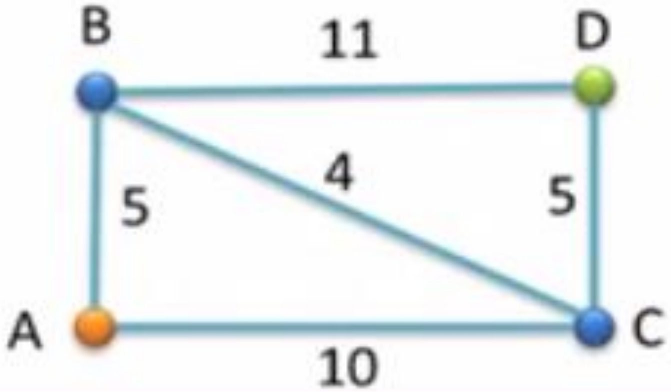
Now that the 4th row is completely filled, our next job is to find the smallest unmarked value in the 4th row.



Looking at the table we can say that 14 is the smallest unmarked value in the 4th row.
So we will mark it with a square box.

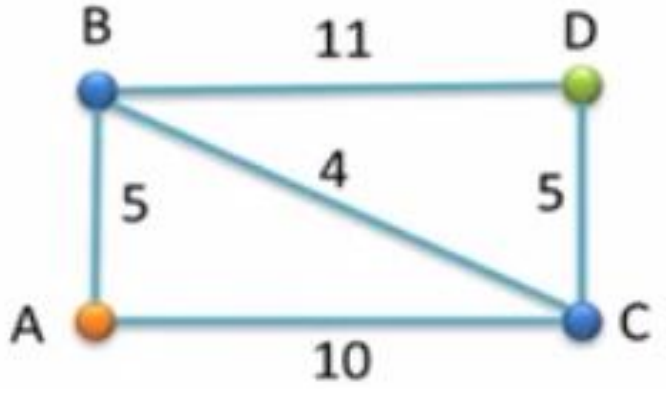
Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

As final vertex **D** is marked so we will stop here.



Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Note! Column D has the marked value 14.
This means our shortest path has the weight 14.



ALGORITHM

//Algorithm:Dijkstra's Algorithm

Let S be the set of explored nodes
    For each $u \in S$, we store a distance $d(u)$ 1
Initially $S = \{s\}$ and $d(s) = 0$

While $S \neq V$

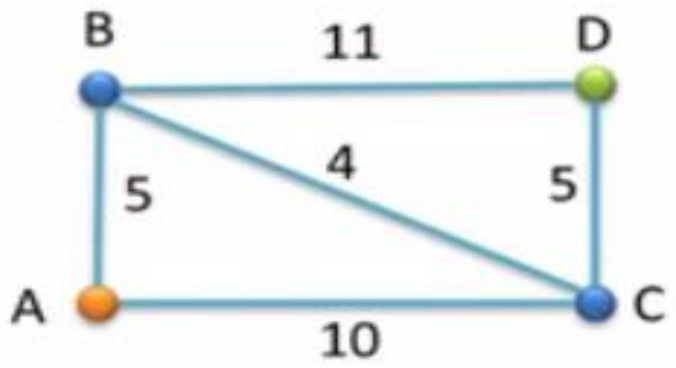
    Select a node $v \notin S$ with at least one edge from S for which
        $d'(v) = \min_{e=(u,v): u \in S} d(u) + \ell_e$ is as small as possible   2
    Add v to S and define $d(v) = d'(v)$

EndWhile

Time to find the shortest path using backtracking

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Backtracking is very simple.
We will move upwards row by row and will stop only when we find a change in value.



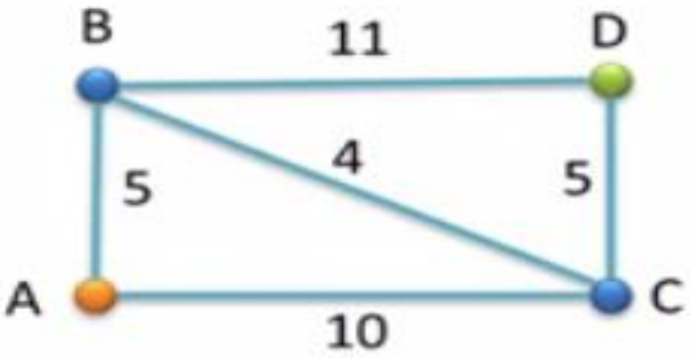
Start from the final marked value 14.
Follow the steps carefully.

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C ✓	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D ✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Move one row upwards.

Has the value changed?

YES
It is 16.

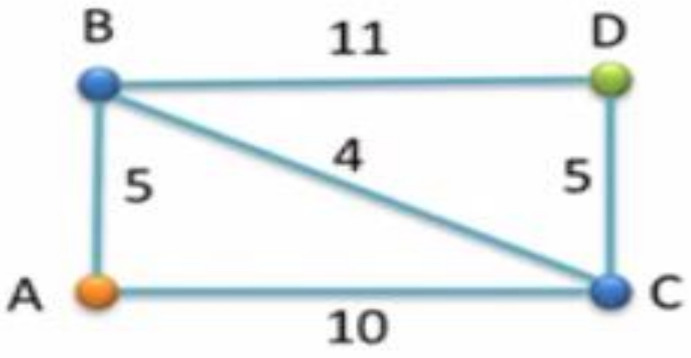


So, tick the vertex that was marked in that row.

i.e., vertex C

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C ✓	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D ✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Move one row upwards.

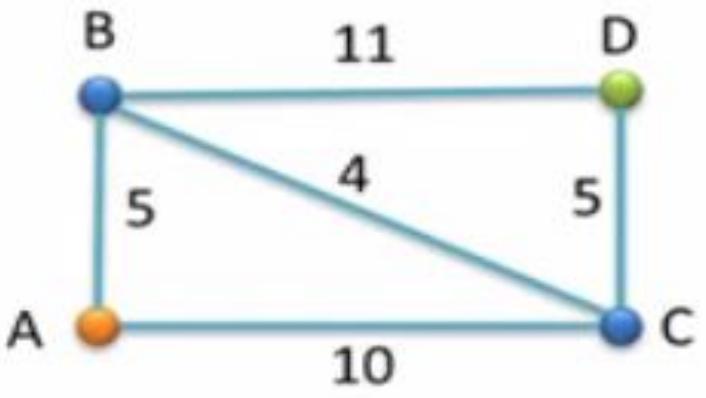


Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B ✓	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C ✓	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D ✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Move one row upwards.

Has the value changed?

YES
It is 10.

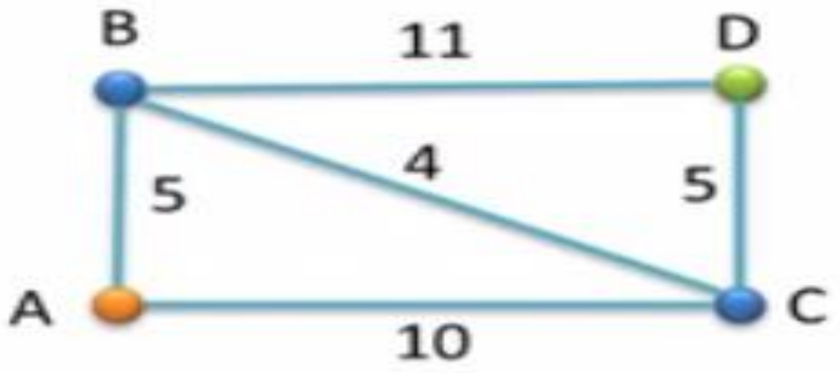


So, tick the vertex that was marked in that row.

i.e., vertex B

Marked	A	B	C	D
A	$\boxed{0}$	∞	∞	∞
B ✓	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C ✓	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D ✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Move the pointer to point at value 5 in column B

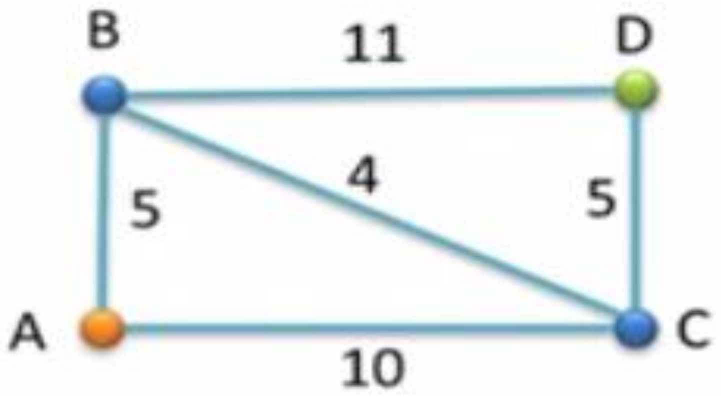


Marked	A	B	C	D
A✓	$\boxed{0}$	∞	∞	∞
B✓	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C✓	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Move one row upwards.

Has the value changed?

YES
It is ∞ (Infinity).

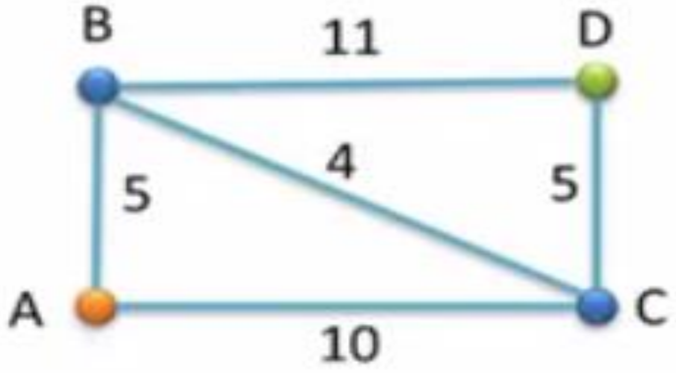


So, tick the vertex that was marked in that row.

i.e., vertex A

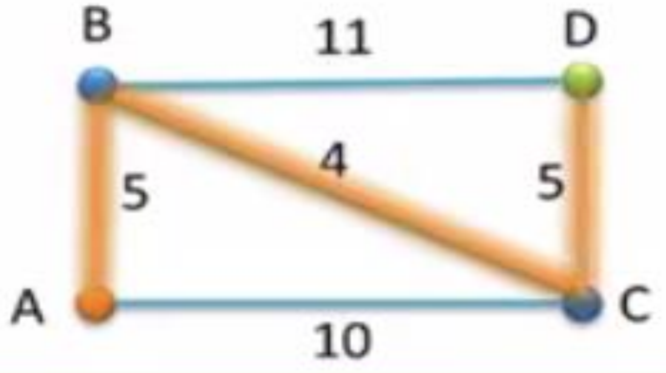
Marked	A	B	C	D
A ✓	$\boxed{0}$	∞	∞	∞
B ✓	$\boxed{0}$	Min(∞, 0+5) $\boxed{5}$	Min(∞, 0+10) 10	∞
C ✓	$\boxed{0}$	$\boxed{5}$	Min(10, 5+4) $\boxed{9}$	Min(∞, 5+11) 16
D ✓	$\boxed{0}$	$\boxed{5}$	$\boxed{9}$	Min(16, 9+5) $\boxed{14}$

Since the initial vertex A is ticked.
So we will stop here.



Therefore, the required shortest path is

A → B → C → D

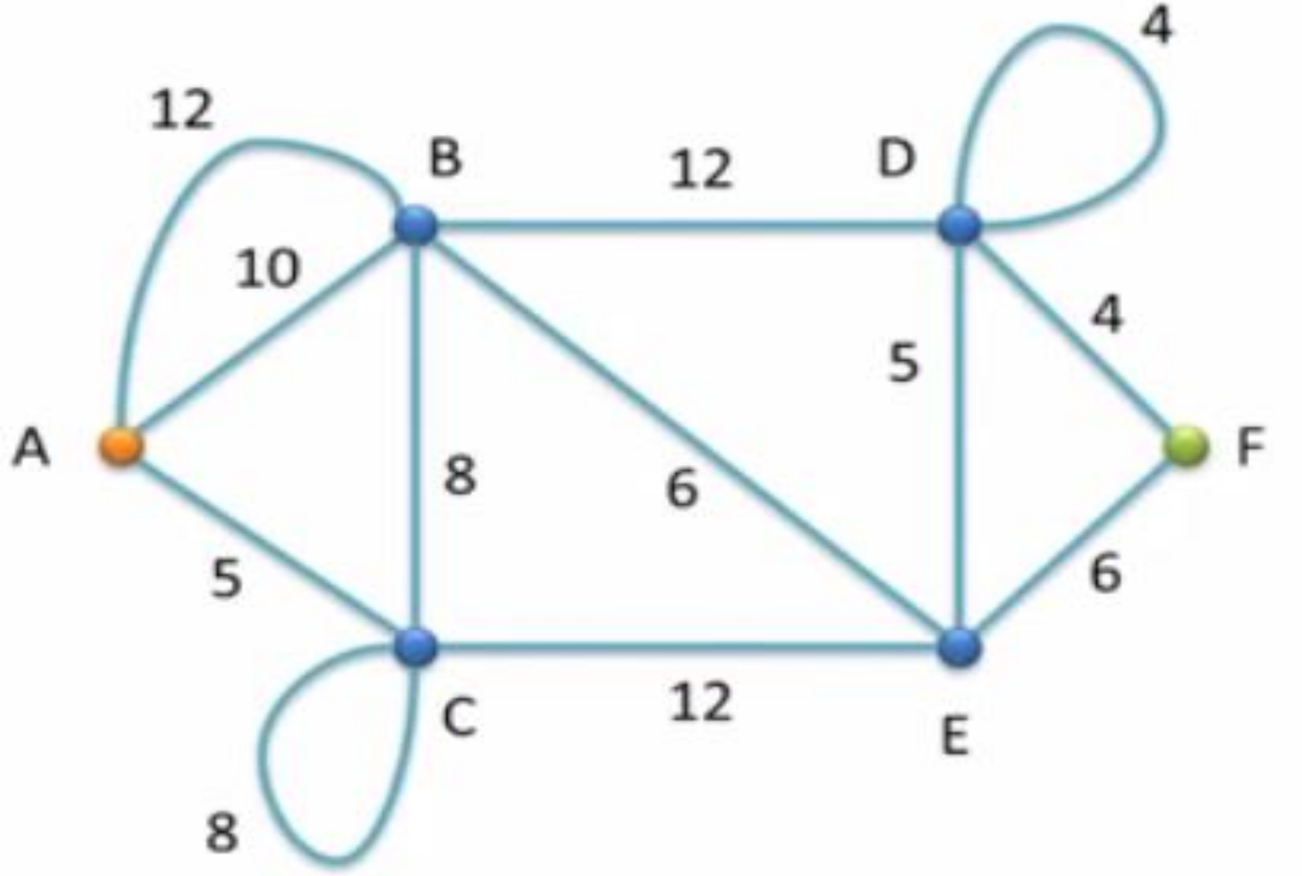


Point to Remember

In case there are two or more columns with the same smallest unmarked value, then you can choose anyone of them.

This only denotes that there will be more than one shortest path in the graph.

Here is our graph



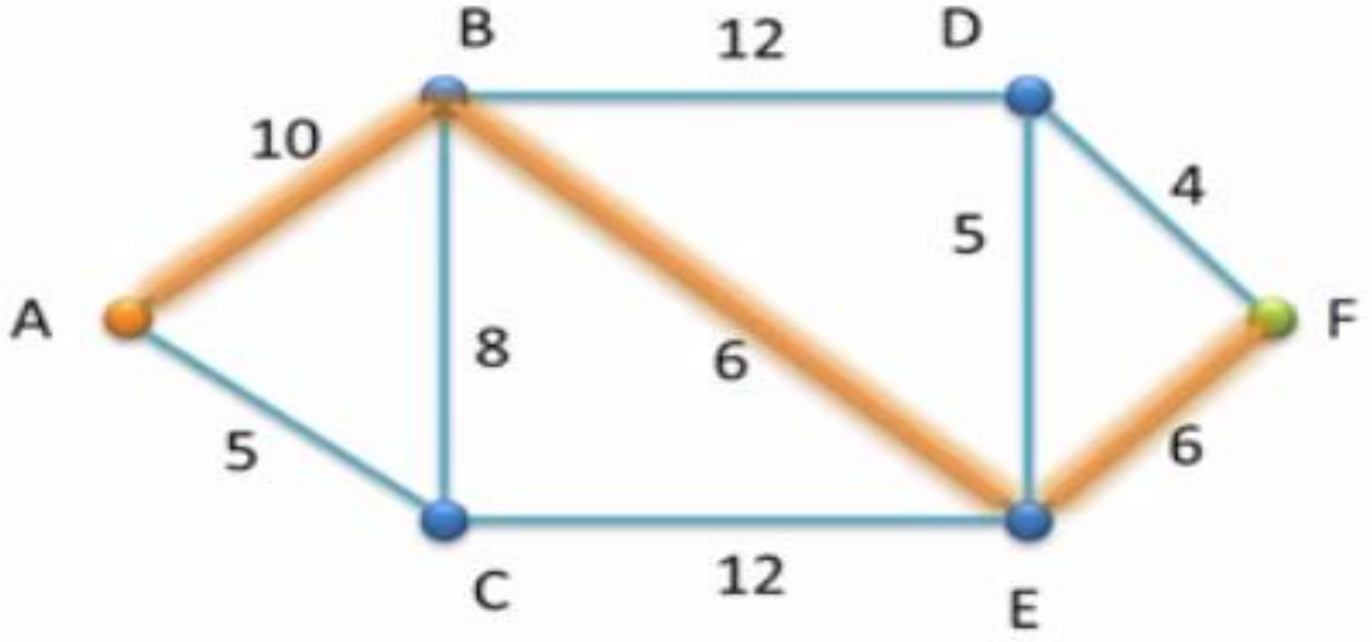
Marked	A	B	C	D	E	F
A ✓	$\boxed{0}$	∞	∞	∞	∞	∞
C	$\boxed{0}$	Min(∞, 0+10) 10	Min(∞, 0+5) $\boxed{5}$	∞	∞	∞
B ✓	$\boxed{0}$	Min(10, 5+8) $\boxed{10}$	$\boxed{5}$	∞	Min(∞, 5+12) 17	∞
E ✓	$\boxed{0}$	$\boxed{10}$	$\boxed{5}$	Min(∞, 10+12) 22	Min(17, 10+6) $\boxed{16}$	∞
D	$\boxed{0}$	$\boxed{10}$	$\boxed{5}$	Min(22, 16+5) $\boxed{21}$	$\boxed{16}$	Min(∞, 16+6) 22
F ✓	$\boxed{0}$	$\boxed{10}$	$\boxed{5}$	$\boxed{21}$	$\boxed{16}$	Min(22, 21+4) $\boxed{22}$

Since the source vertex is ticked. So we will stop here.

Therefore, the required shortest path is

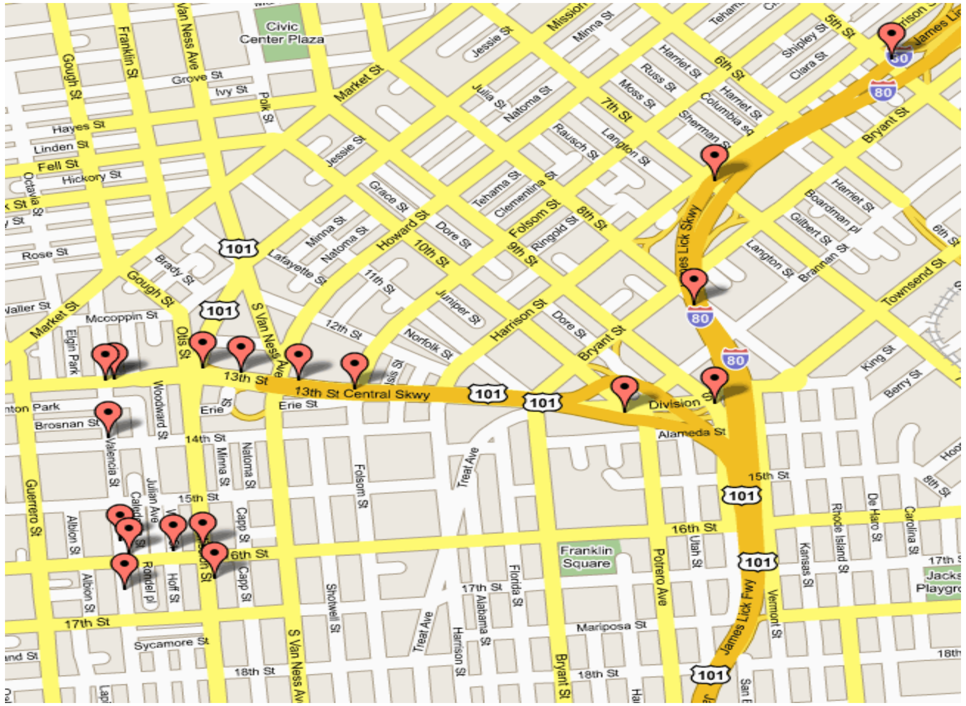
Lets mark the shortest path

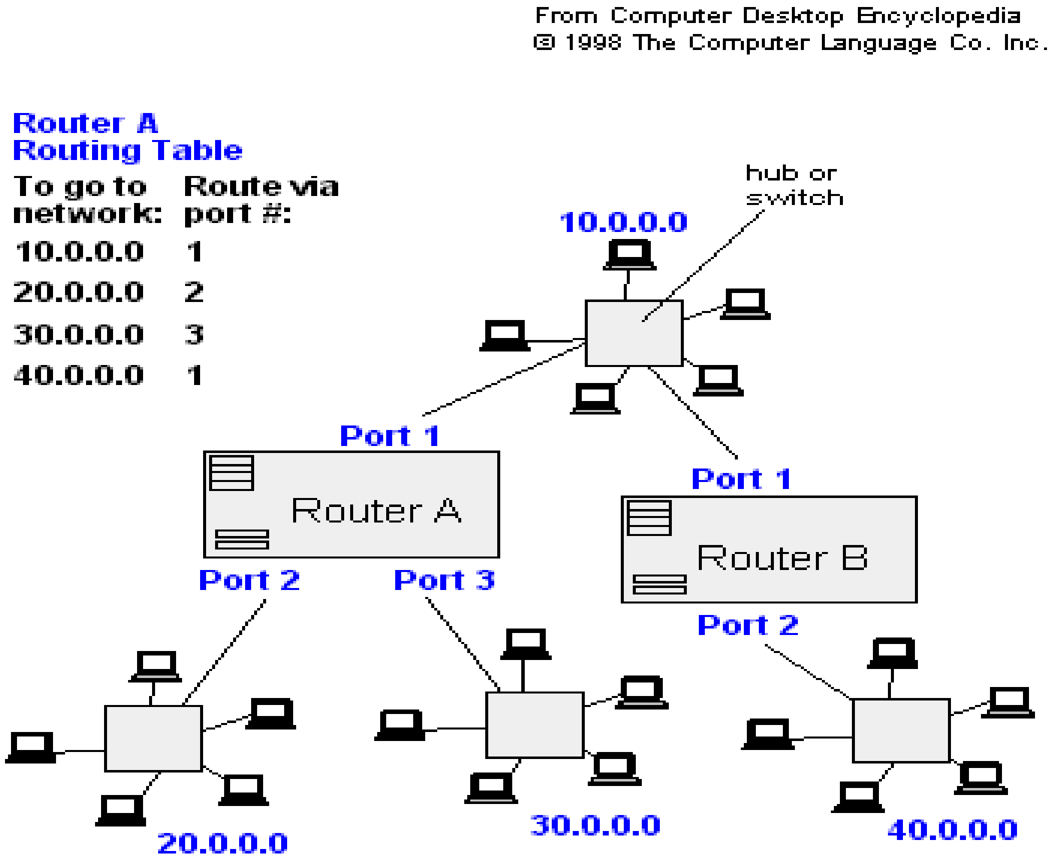
A → B → E → F



Applications of Dijkstra's Algorithm

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





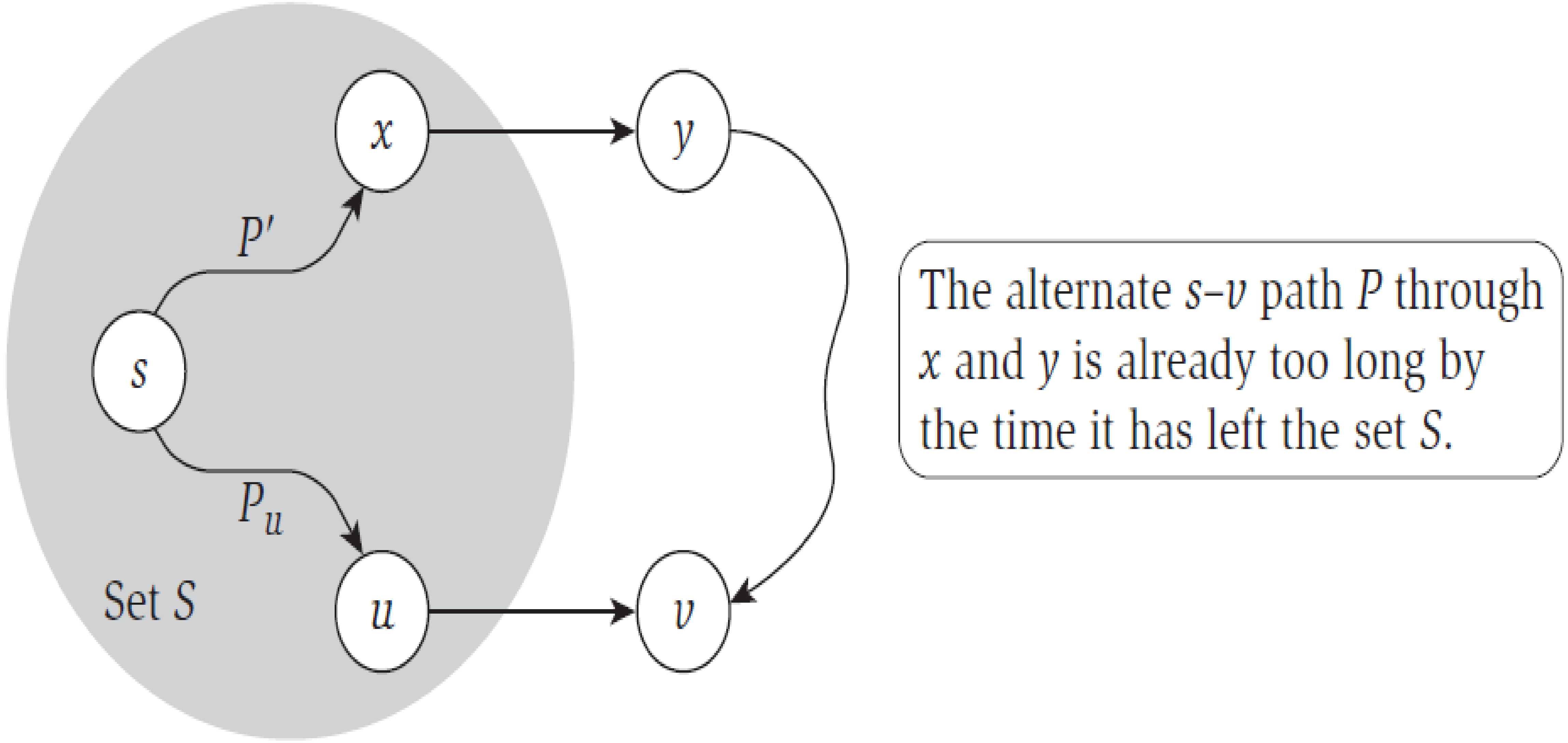
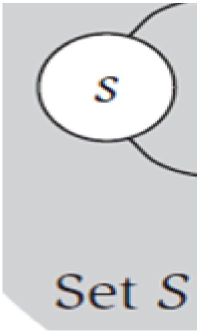
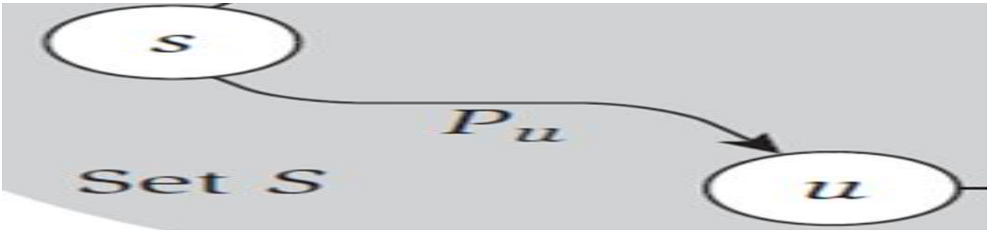


Figure 4.8 The shortest path P_v and an alternate s-v path P through the node y.

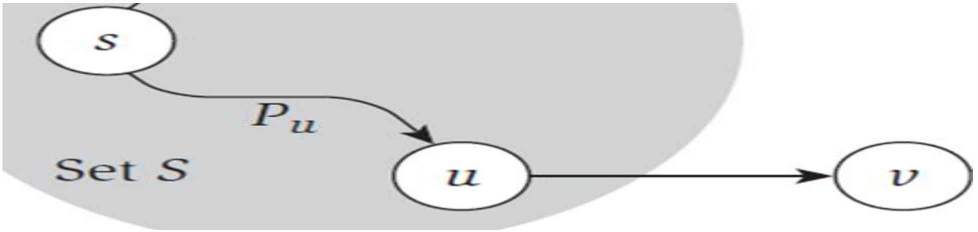
- ***Consider the set S at any point in the algorithm's execution. For each u ∈ S, the path Pu is a shortest s-u path.***



- **Proof.** We prove this by induction on the size of *S*.
- Case |*S*| = 1, we have set *S* = {*s*} and *d(s)* = 0.

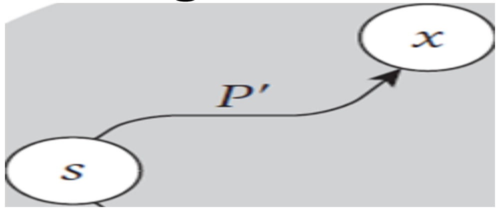


- when |*S*| = *k* for some value of *k* ≥ 1
- By induction hypothesis, *Pu* is the shortest *s*-*u* path for each *u* ∈ *S*.

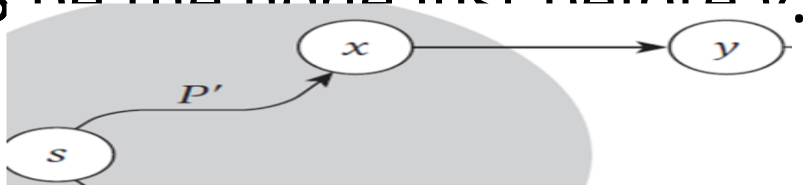


- Now grow *S* to size *k* + 1 by adding the node *v*.
- Let *(u*, *v)* be the final edge on our *s*-*v* path *Pv*.

- Consider any other *s*-*v* path *P*; we wish to show that it is at least as long as *Pv*.
- In order to reach *v*, this path *P* must leave the set *S somewhere.*

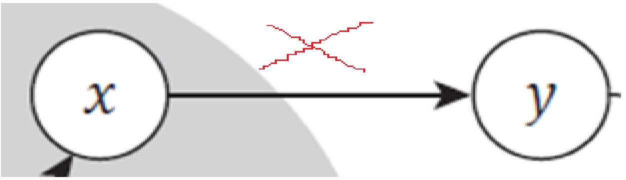


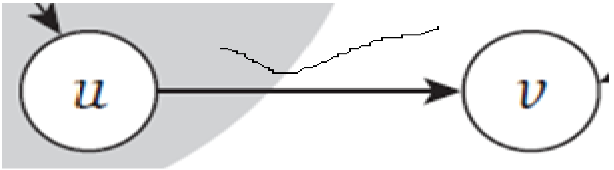
- Let *y* be the first node on *P* that is not in *S*, and let $x \in S$ be the node just before *y*.



- *P* cannot be shorter than *Pv* because it is already at least as long as *Pv* by the time it has left the set *S*.

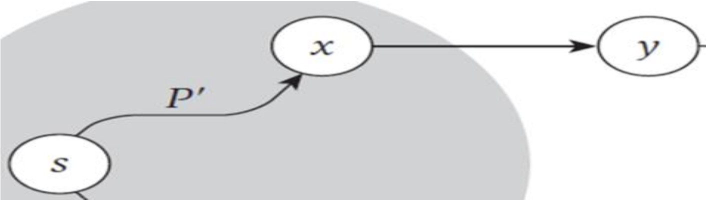
- In iteration *k* + 1, Dijkstra's Algorithm must have considered adding node *y* to the set *S* via the edge *(x, y)* and rejected this option in favor of adding *v*.



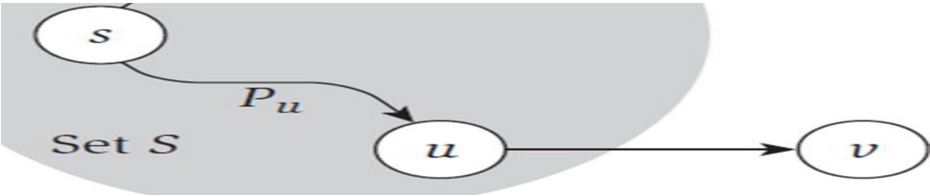


- This means that there is no path from s to y through x that is shorter than Pv

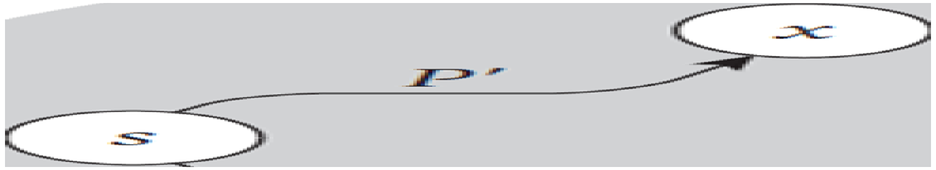
- Since edge lengths are nonnegative, the full path *P* is at least as long as *Pv*



P is at least as long as Pv



- Let P' be the subpath of P from s to x.

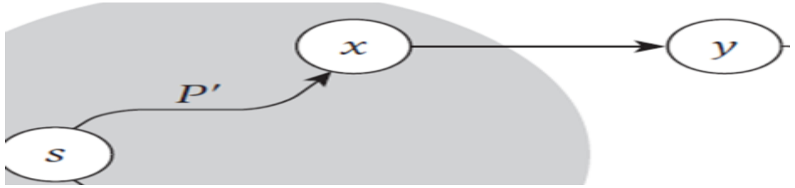


- Since $x \in S$, we know by the induction hypothesis that P_x is a shortest s-x path and so

$$L(P') \geq L(P_x) = d(x).$$

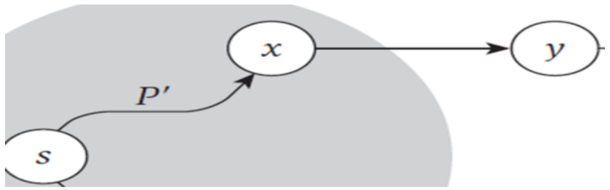
- The subpath of P out to node y has length

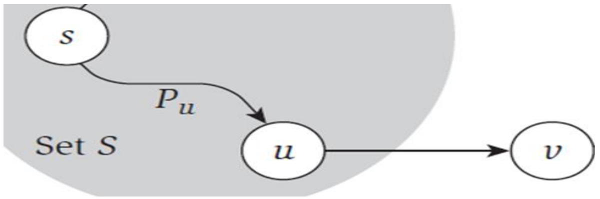
$$L(P') + L(x, y) \geq d(x) + L(x, y) \geq d'(y)$$



- Since Dijkstra's Algorithm selected *v* in this iteration. We know that

$$d'(y) \geq d'(v) = L(Pv).$$

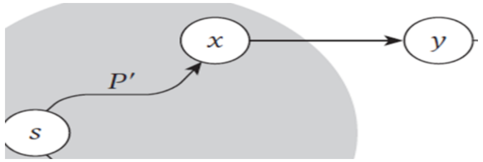


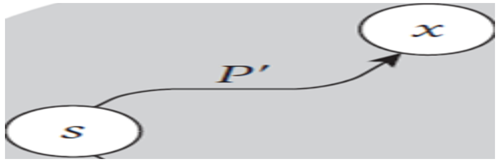


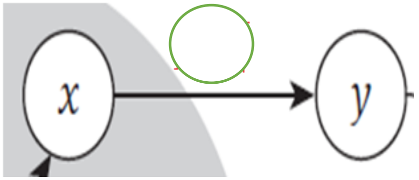
d'(Y) >= d'(v) =L(Pv)

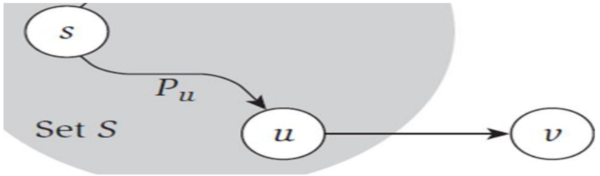
- Combining these inequalities shows that

$$L(P) \geq L(P') + L(x, y) \geq L(Pv).$$





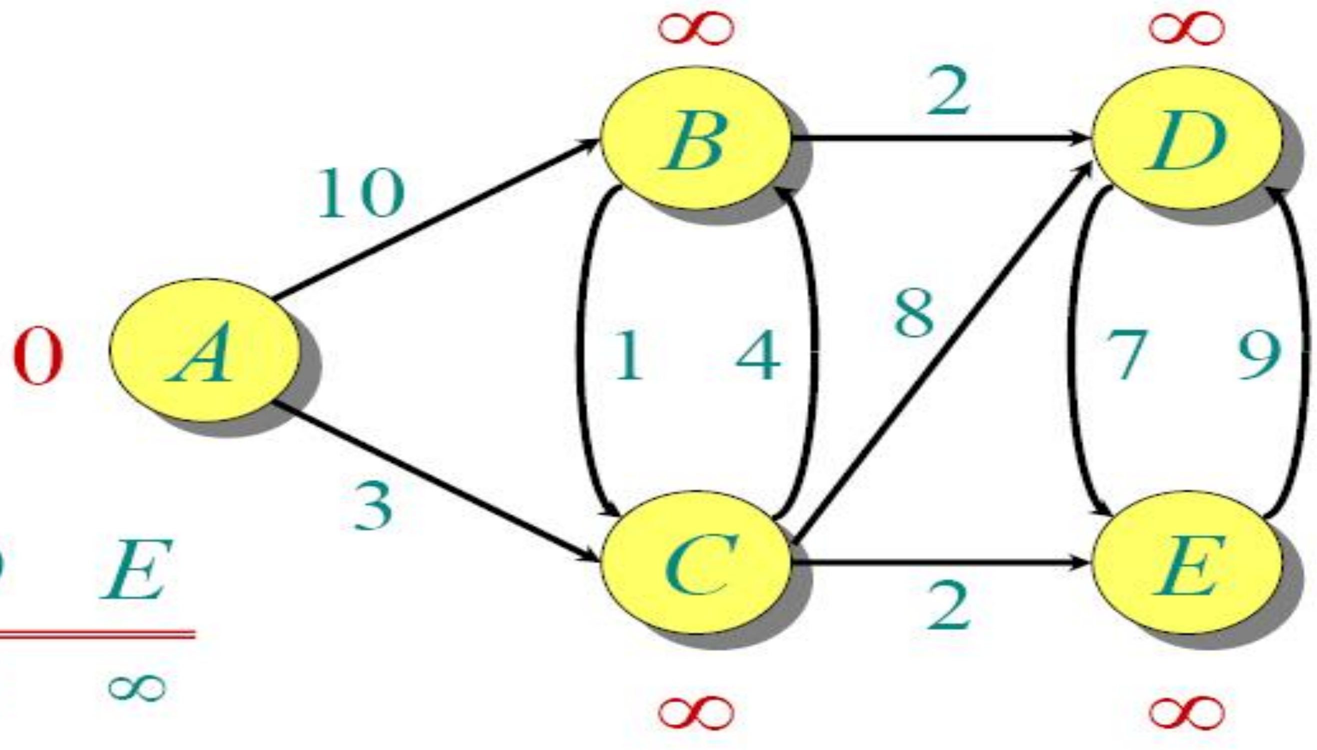




L(P) >= L(P') + L(X,Y) >= L(Pv)

Dijkstra Example for Directed Graph

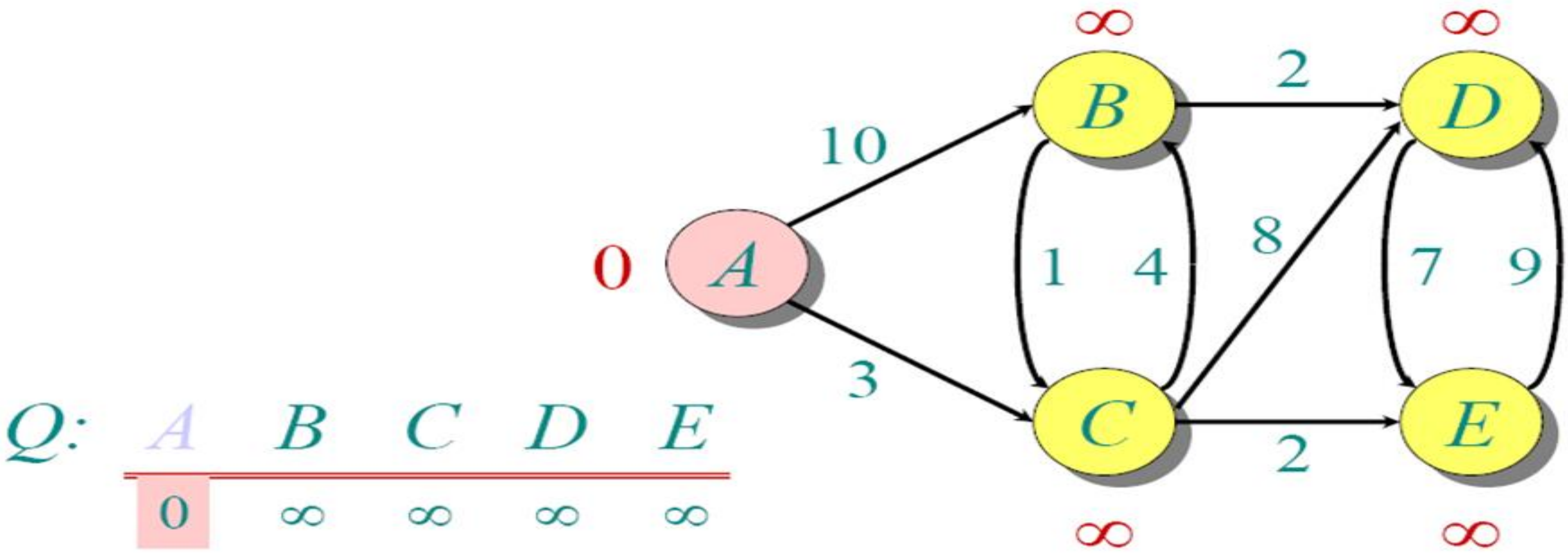
Initialize:



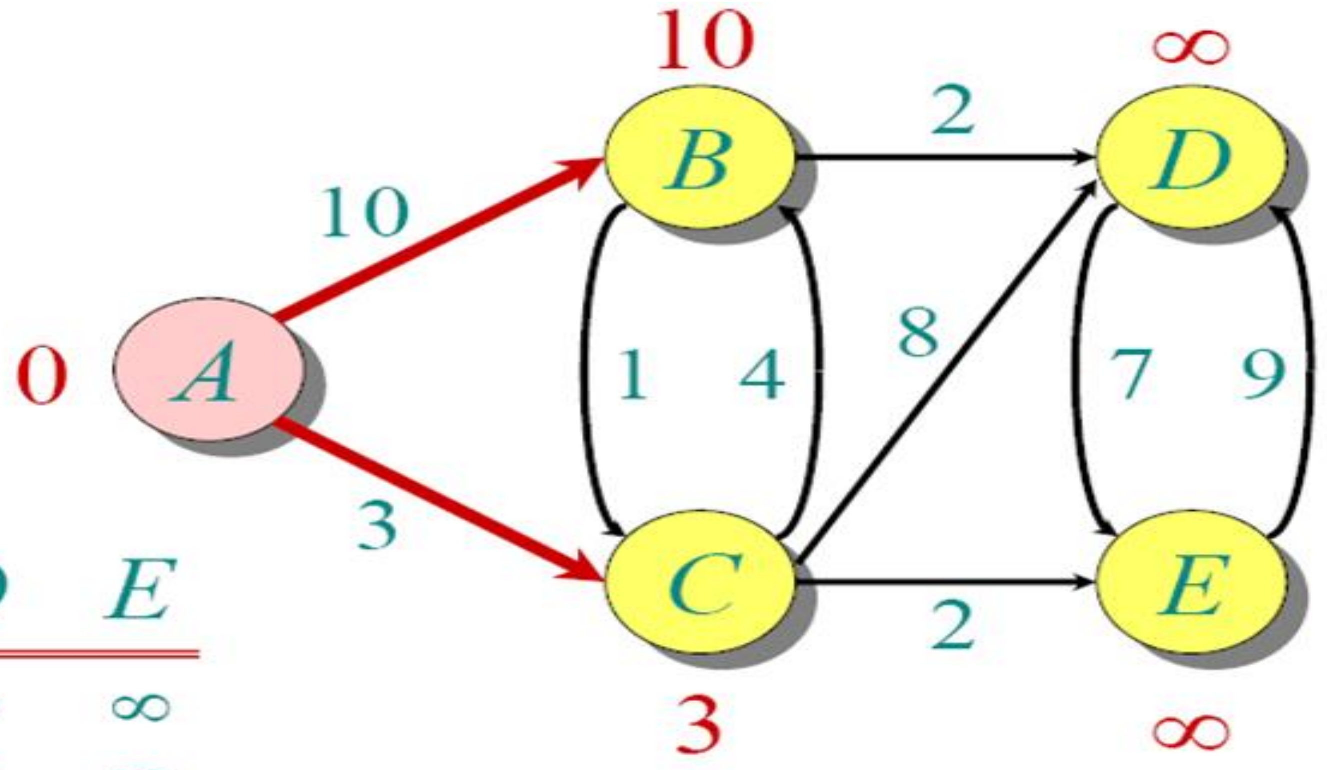
Q:	A	B	C	D	E
	0	∞	∞	∞	∞

S: {}

Dijkstra Animated Example



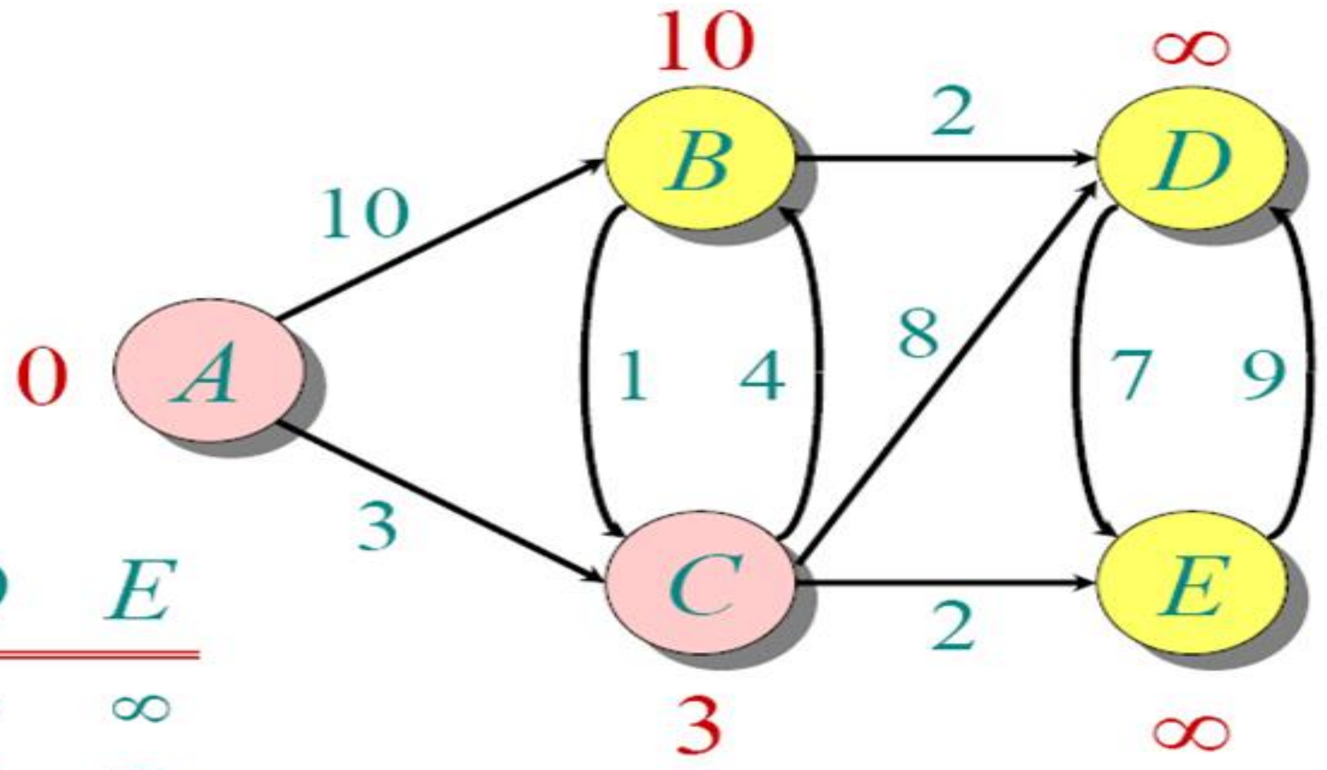
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A }

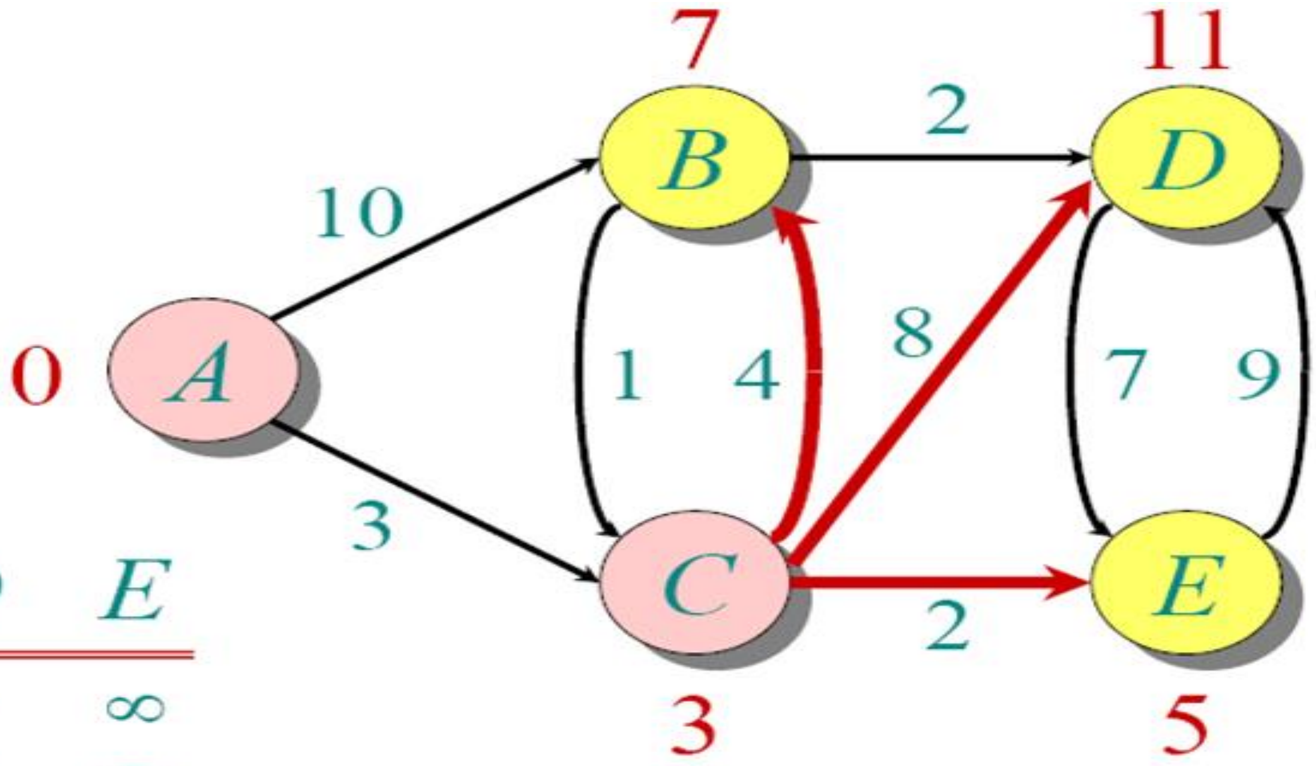
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞

S: { A, C }

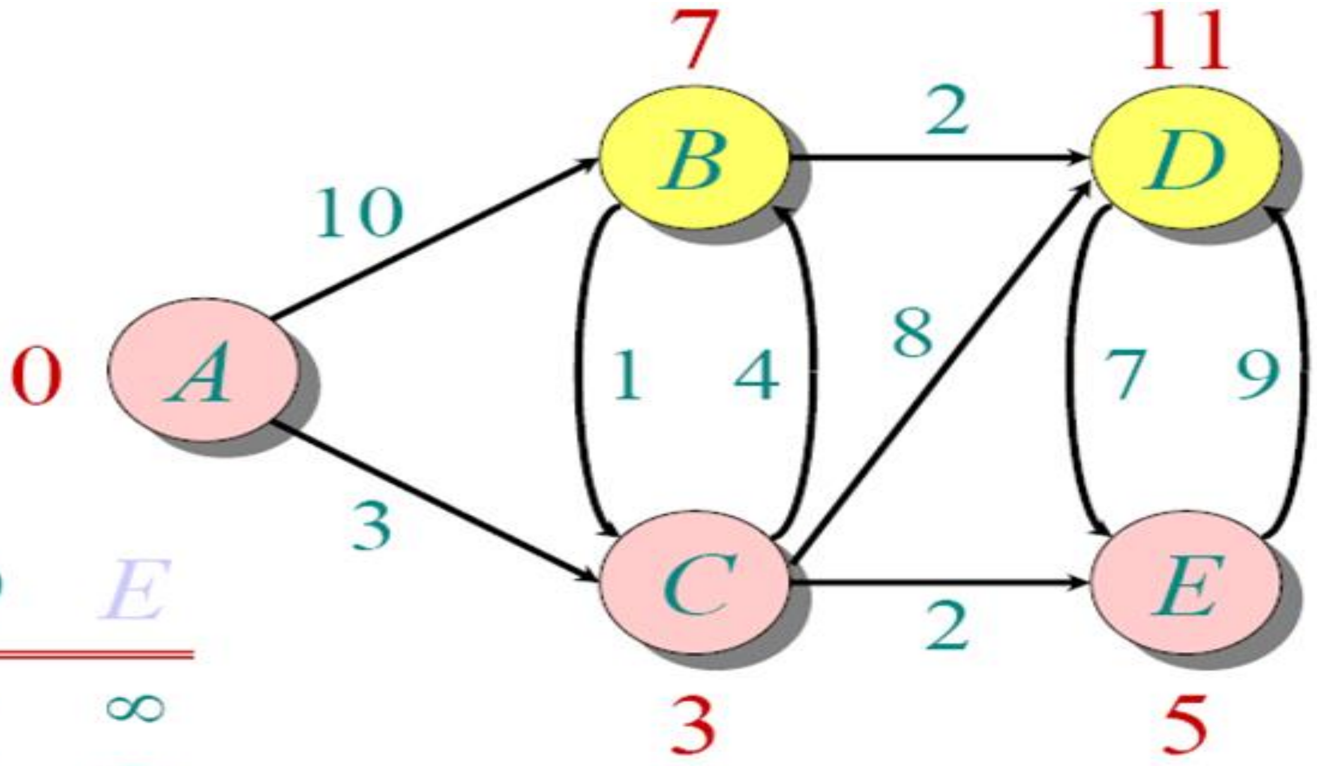
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C }

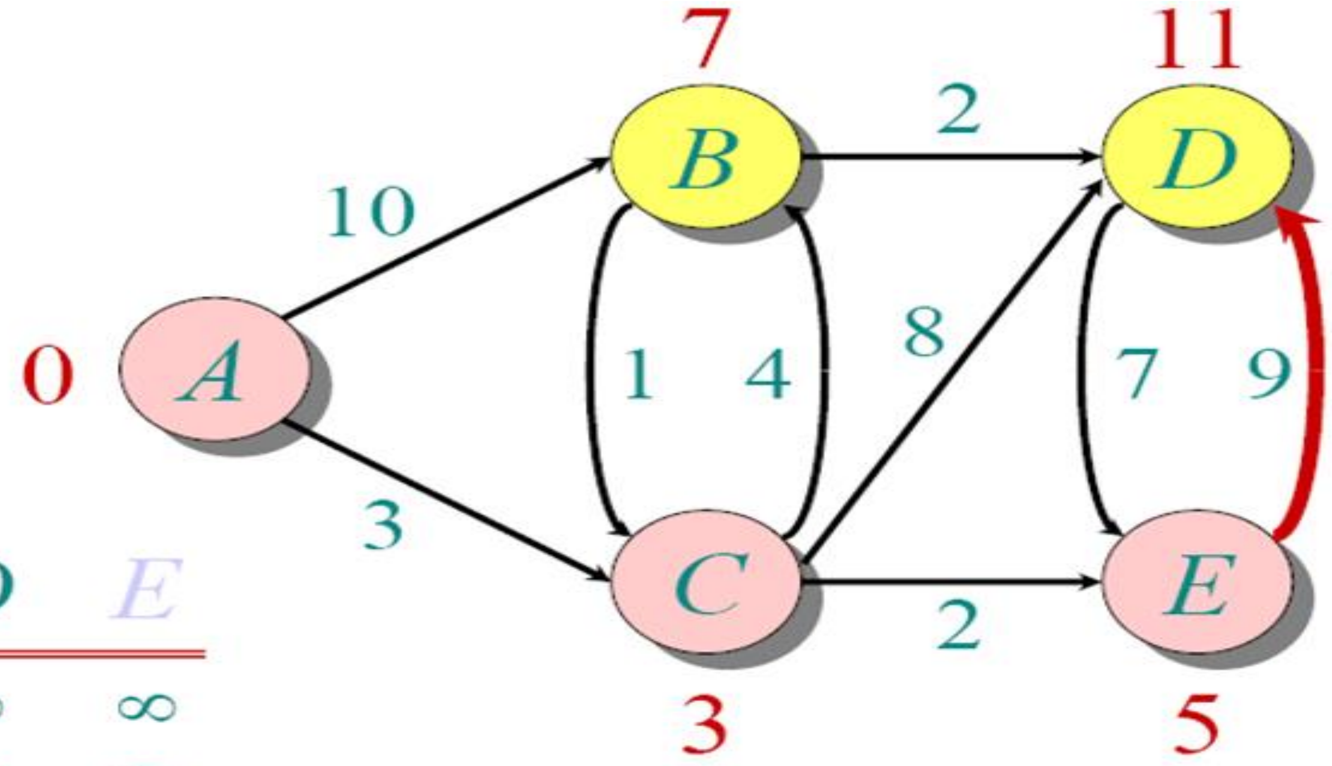
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5

S: { A, C, E }

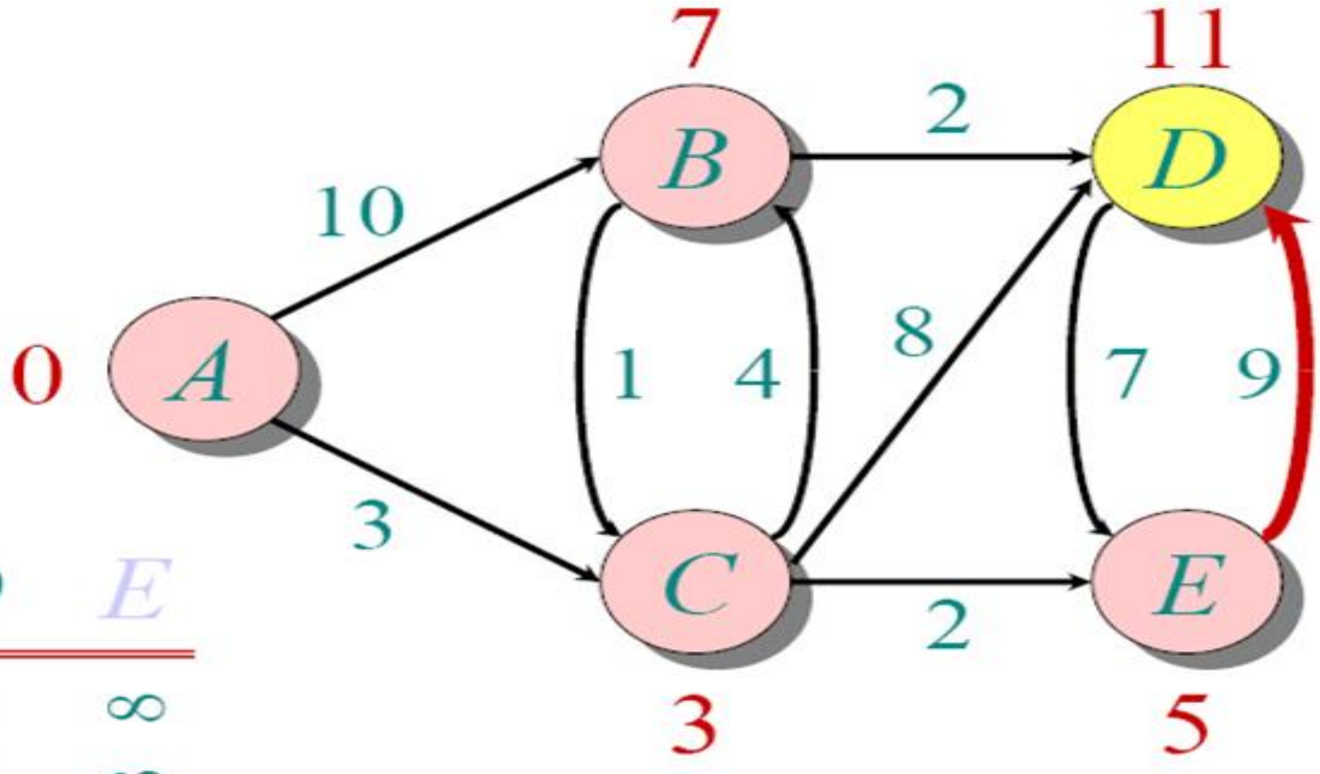
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E }

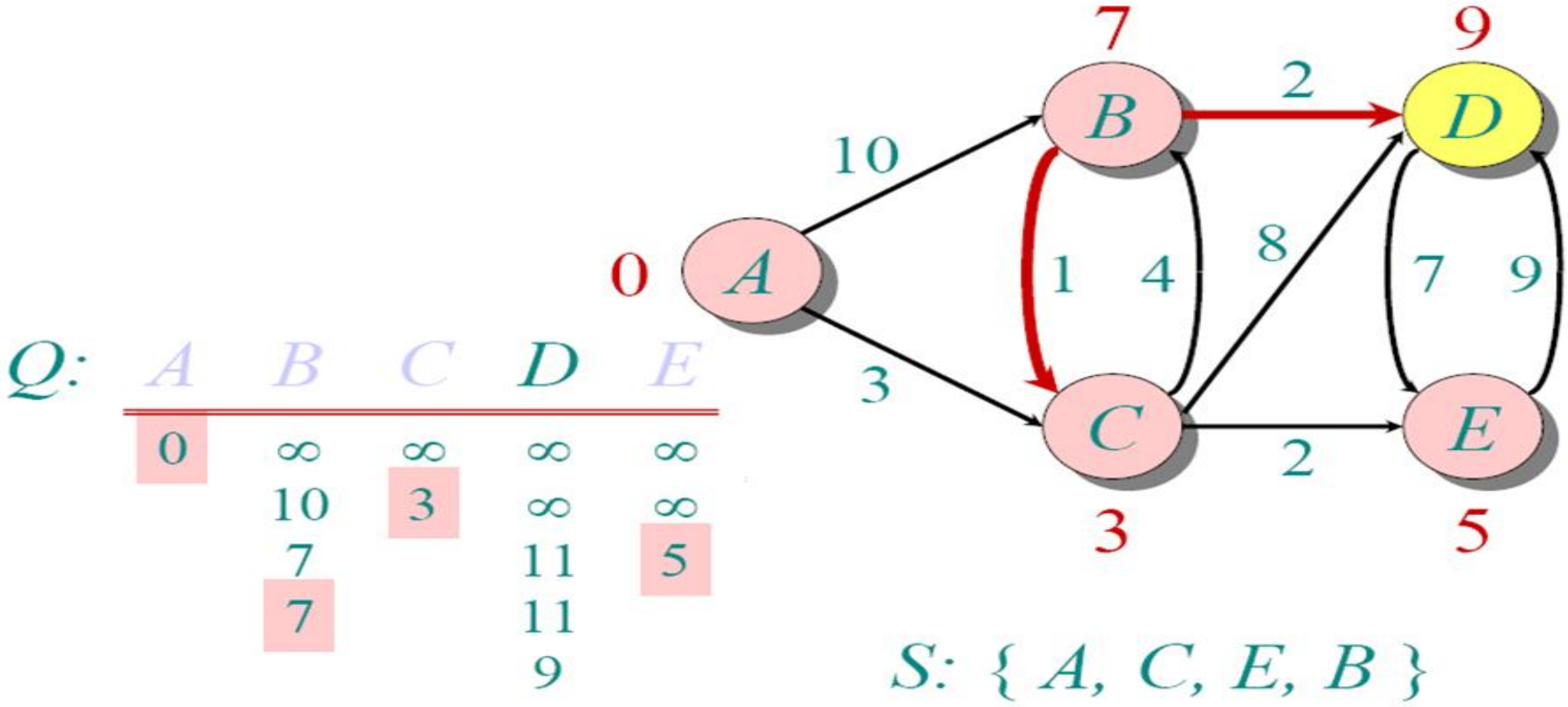
Dijkstra Animated Example



Q:	A	B	C	D	E
	0	∞	∞	∞	∞
		10	3	∞	∞
		7		11	5
		7		11	

S: { A, C, E, B }

Dijkstra Animated Example



Dijkstra Animated Example