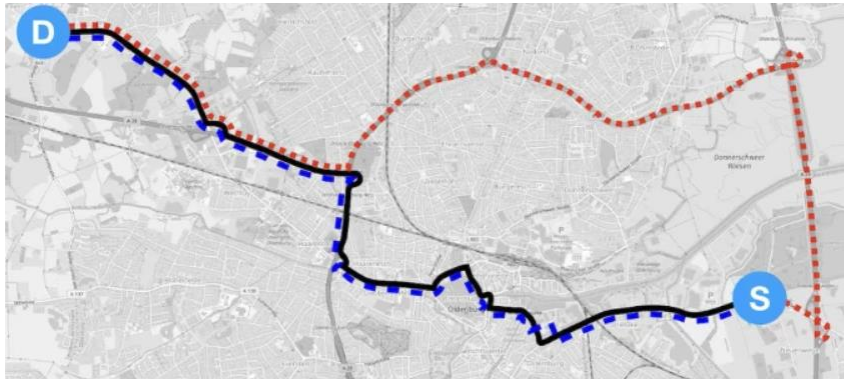
Data Structures and Algorithms

Shortest Path Algorithms



Prepared by:

Eng. Malek Al-Louzi

School of Computing and Informatics– Al Hussein Technical University

Spring 2021/2022

Outlines

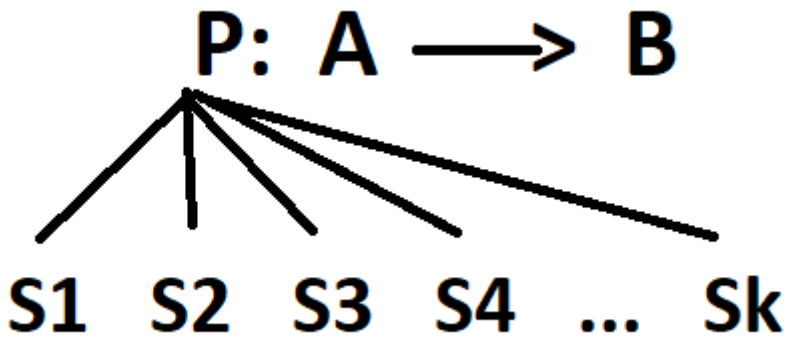
- Greedy Method
- Optimization Problem
- Dijkstra’s Algorithm
- Relaxation
- Bellman–Ford Algorithm

Greedy Method

- Is one of the strategies for solving problems, like Divide & Conquer and other strategies.
- It is used for solving Optimization Problems.
- **Optimization** problems are problems which require either **minimum** result or **maximum** result.

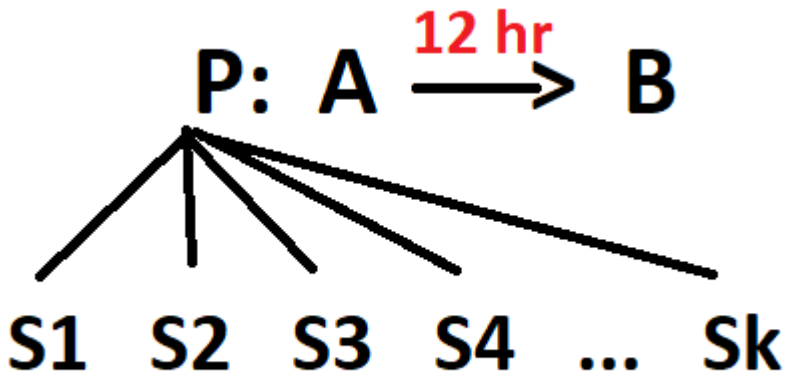
Optimization Problem

- Suppose we have a problem P, and the problem is that I want to travel from location A to location B.
- For this problem, there might be more than one solution, for example, I can travel by walk, by car, by train, by plane …etc.



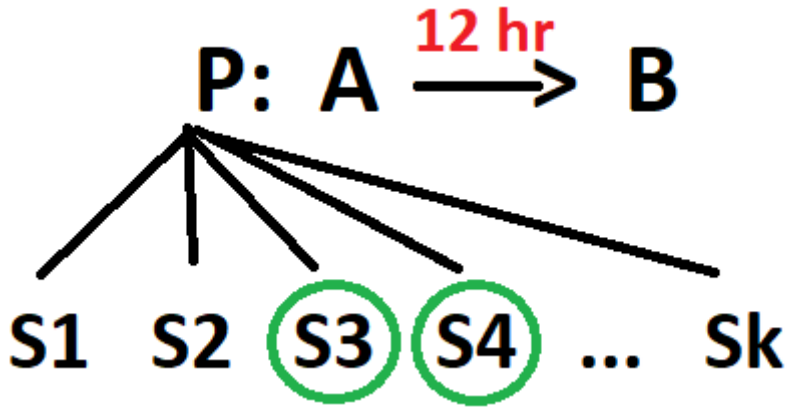
Optimization Problem

- If there is a constraint on the problem which is that we have to cover this journey within 12 Hours.



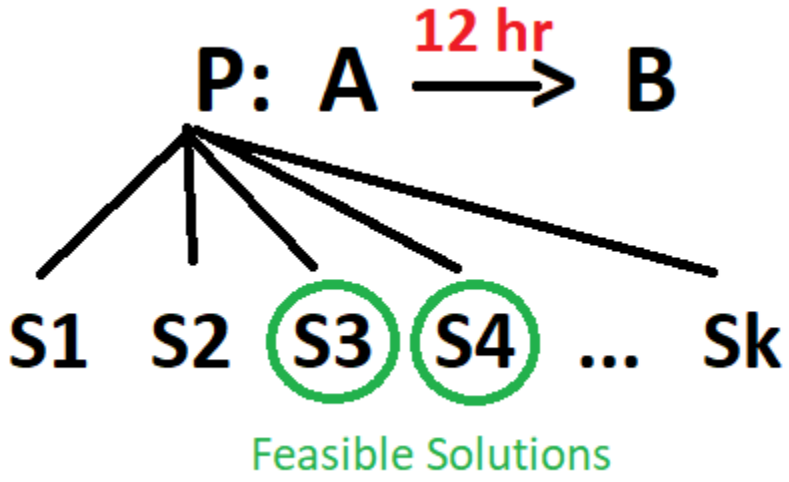
Optimization Problem

- Suppose we cannot achieve this constraint if we go by walk or by car, we can cover it only if we go by train or by plane.



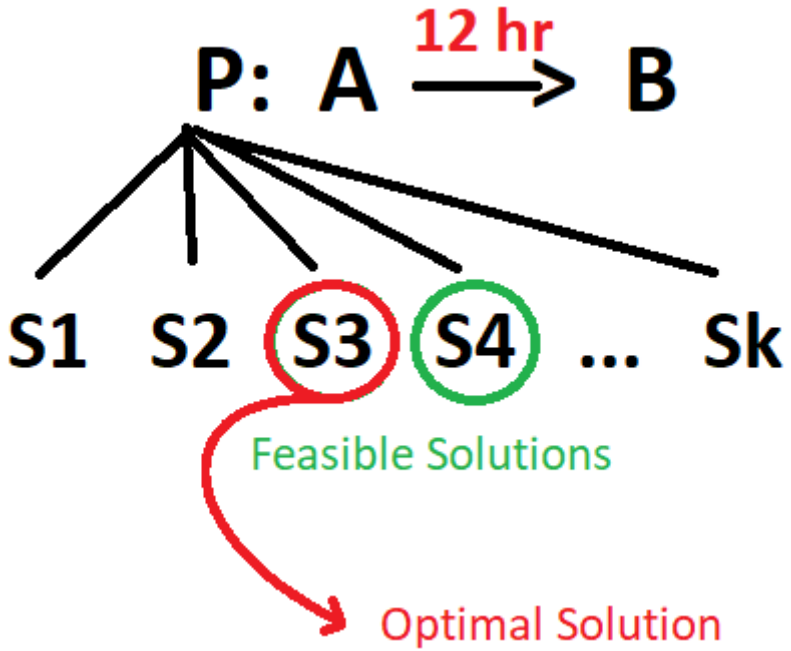
Optimization Problem

- For this problem, there are many solutions, but there are some solutions which are satisfying the condition that given in the problem.
- The solutions that satisfying the condition are called **Feasible Solutions**.



Optimization Problem

- Now, if we say that we want to cover this journey in minimum cost, then this problem becomes minimization problem.
- Suppose that traveling by train is the minimum cost, then travelling by train is called the optimal solution.



Optimization Problem

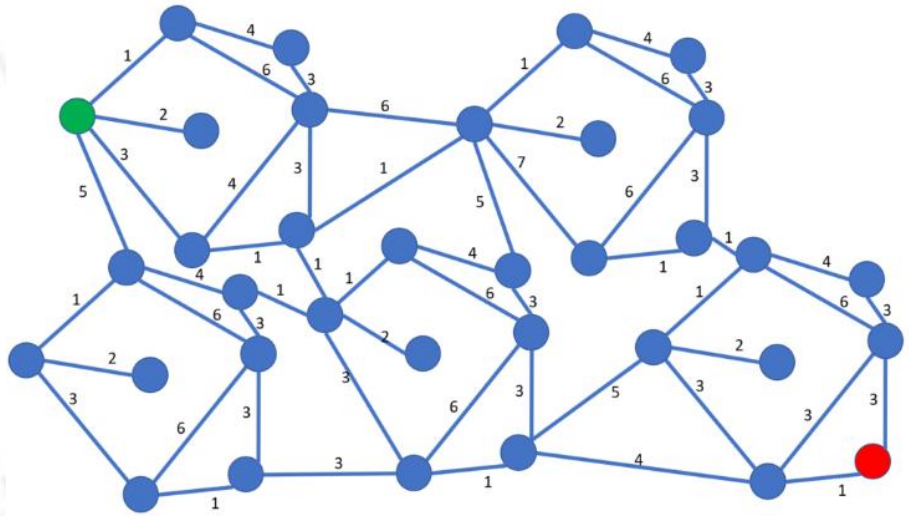
- The optimal solution is a solution which is already feasible which also give us the best result.
- For any problem there can be only **One** optimal solution.
- The following strategies are used to solve optimization problems:
  - **Greedy method.**
  - **Dynamic programming.**
  - **Branch and Bound.**

Greedy Method

- Greedy method says that a problem should be solved in stages, in each stage we will consider one input from a given problem.
- If that input is feasible, then we will include it in the solution, by including all these feasible inputs we will get the optimal solution.
- In other words, it is solving the problem by selecting the locally optimal choice at each stage hoping to get the globally optimum solution.

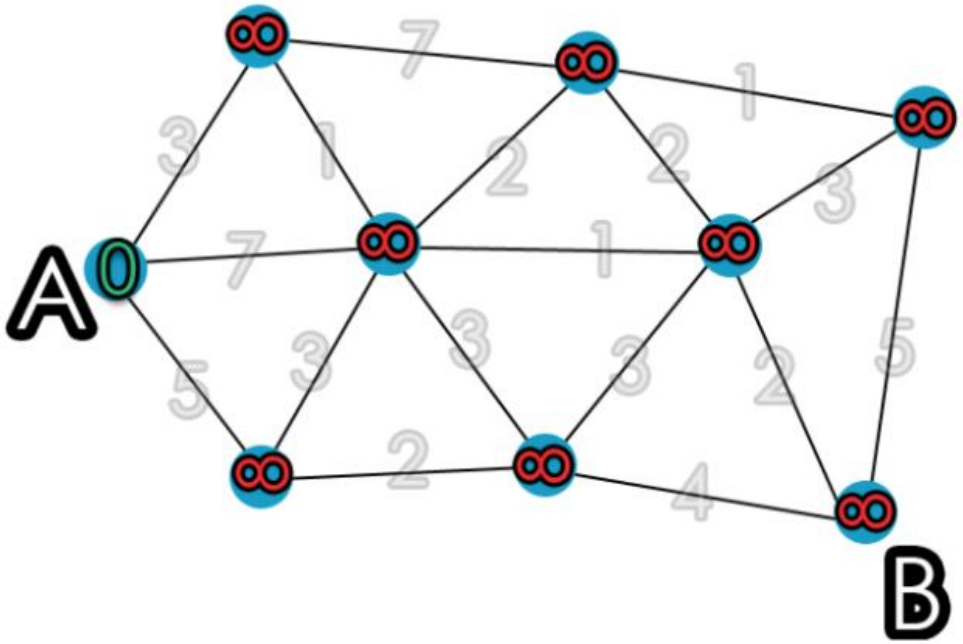
Dijkstra's Algorithm

- It was invented by a famous Dutch Computer Scientist.
- The objective of the algorithm is to find the shortest path between any two vertices in a graph.
- In fact, Dijkstra's Algorithm will find the shortest path from a given starting vertex to every other vertex in the graph.



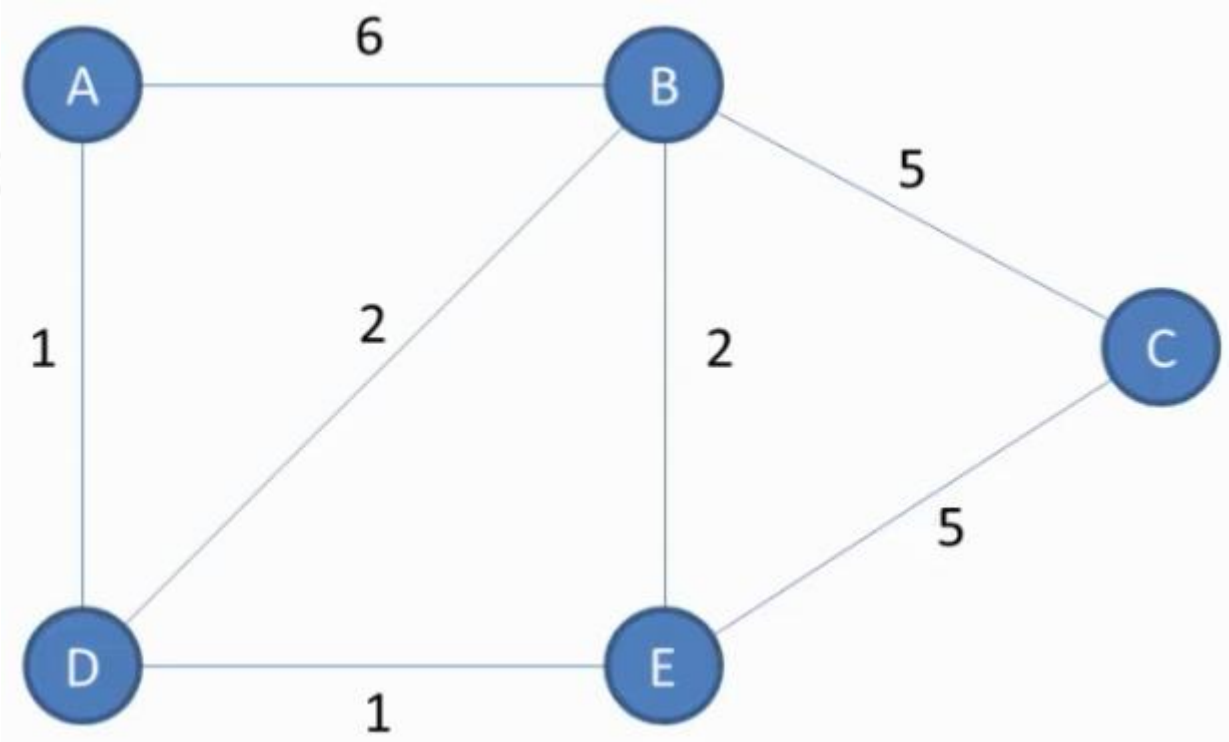
Dijkstra's Algorithm

- We can select any vertex in a graph as a start vertex.
- As we have to find the **shortest** path, so it is a **minimization** problem.
- As we said, a minimization problem is an optimization problem.



Dijkstra's Algorithm
Example 1

- Consider this simple graph, our objective is to find is to find the shortest path from A to C.



Dijkstra's Algorithm
Example 1

- Once it is run, the algorithm will generate the following information which includes everything we need to know.

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

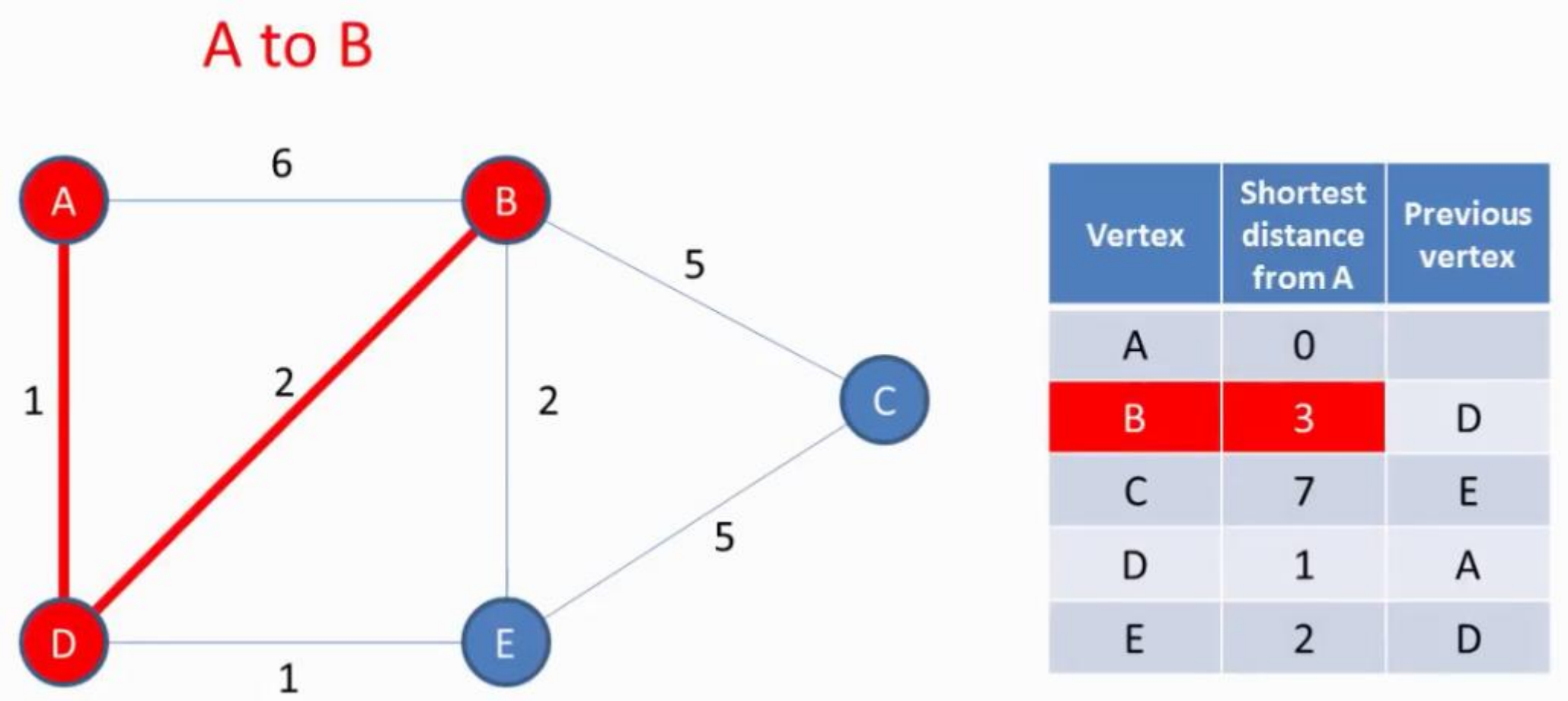
Dijkstra's Algorithm
Example 1

- In the middle, we have the total shortest distance from the starting vertex to all the other vertices.

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

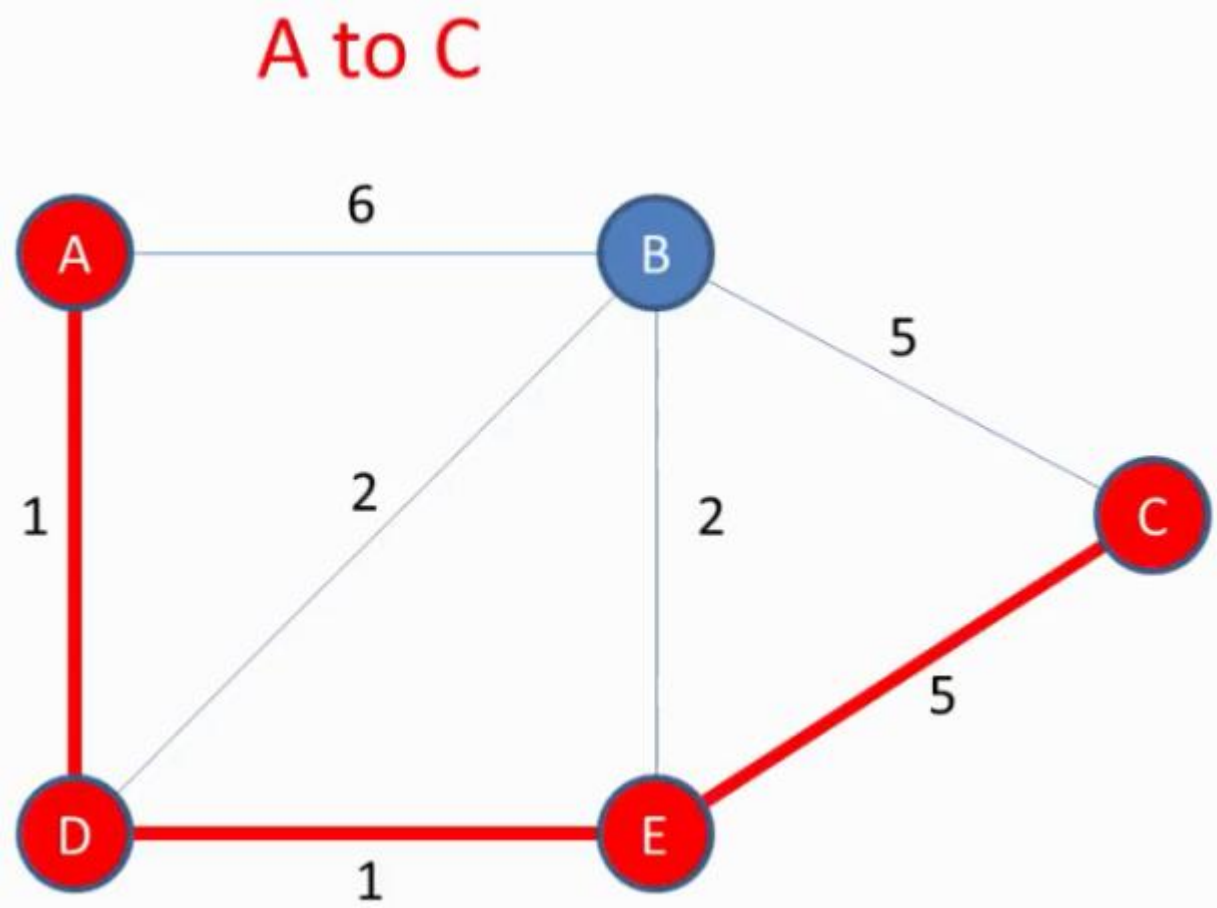
- For example, the total shortest distance from A to B is 3.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

- For example, the total shortest distance from A to C is 7.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

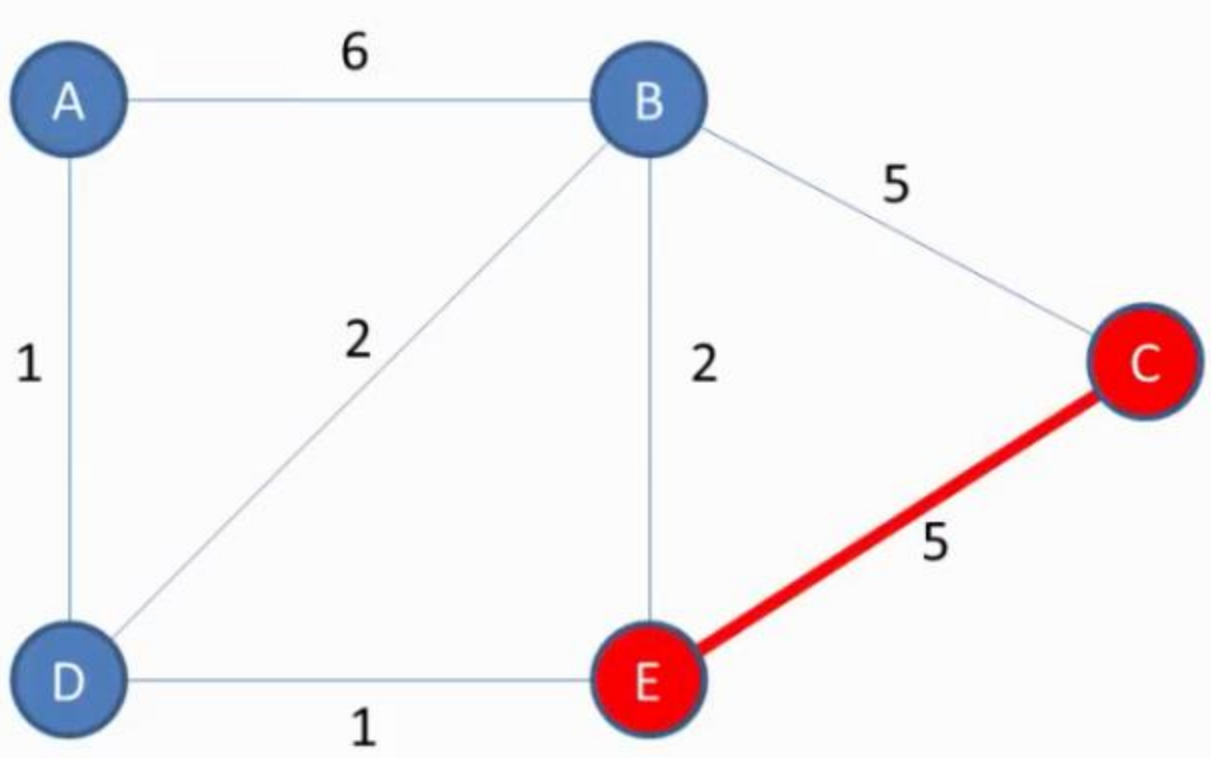
Dijkstra's Algorithm
Example 1

- We also have the shortest sequence of vertices from A to every other vertex.

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

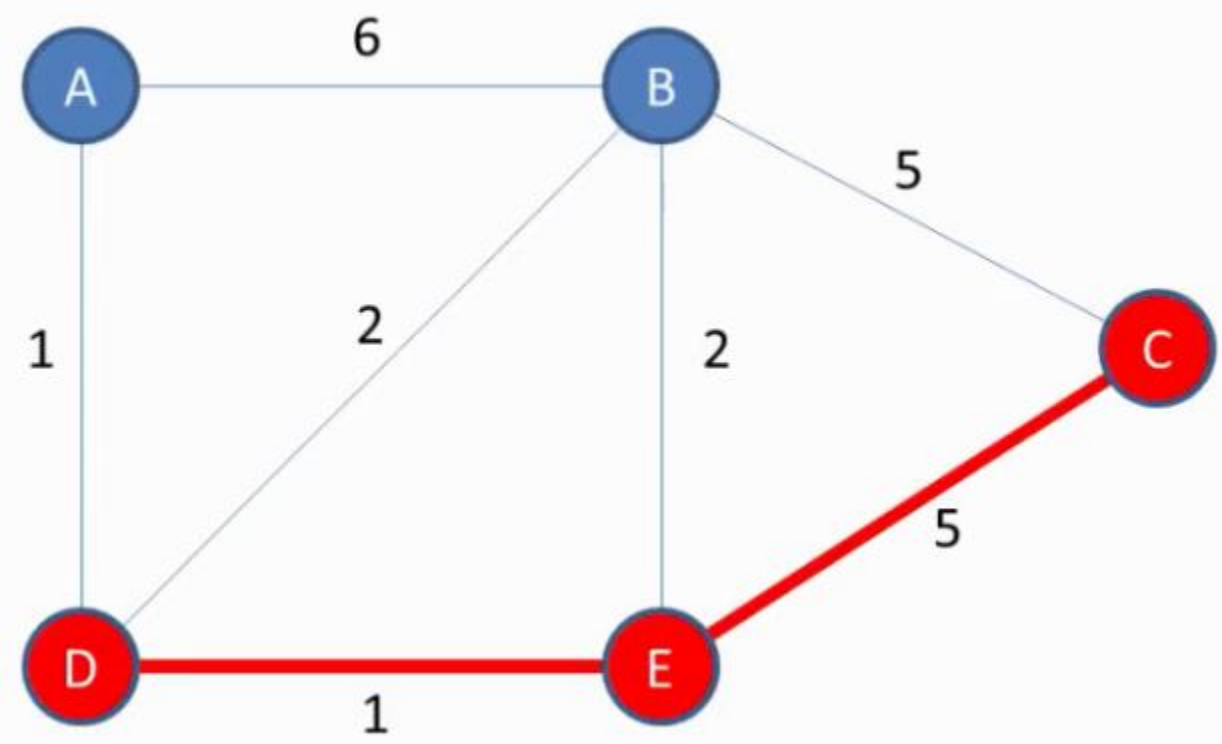
- For example, to get from A to C, notice we arrived at C via E, which is shown in the previous vertex column.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

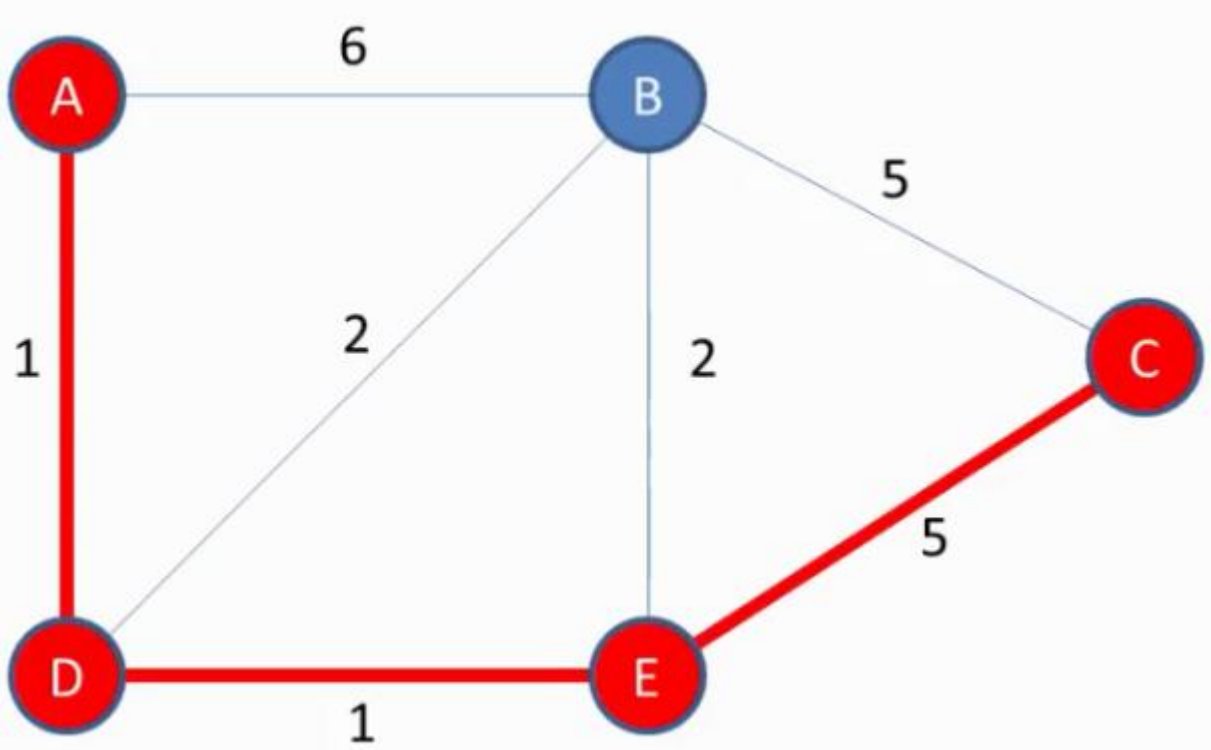
- If we check the information for E, we can see that we arrived at E via D.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

- Again, If we check the information for D, we can see that we arrived at E directly from A.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

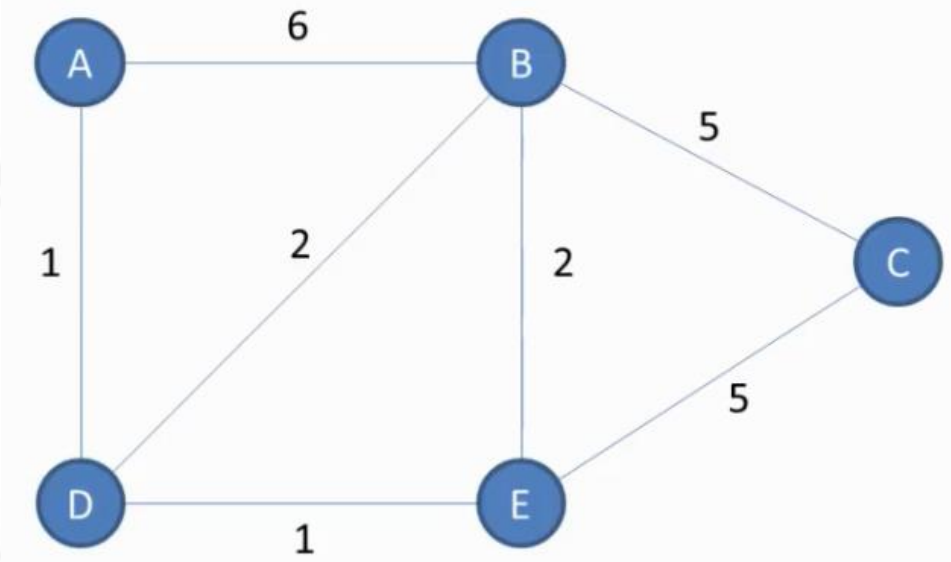
Dijkstra's Algorithm
Example 1

- Now, how does Dijkstra's Algorithm work and generate this information table?

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

- We will use two lists:
  - One to keep track the of the vertices that we visited.
  - The other one to keep track of the vertices that we haven't visited yet.

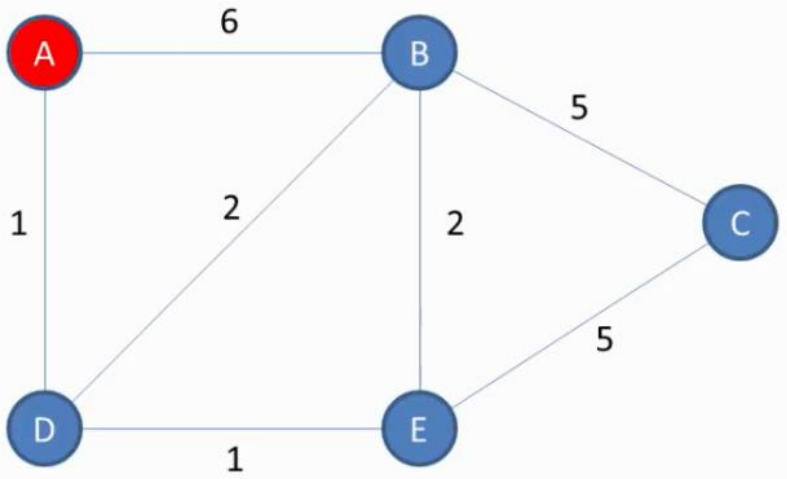


Vertex	Shortest distance from A	Previous vertex
A		
B		
C		
D		
E		

Visited = [ ] Unvisited = [A, B, C, D, E]

Dijkstra's Algorithm
Example 1

- Step 1: Consider the starting vertex A.
  - Distance from A to A is 0.
  - Distances to all other vertices from A are unknown, therefore ∞.
  - We will include these information in our table directly.

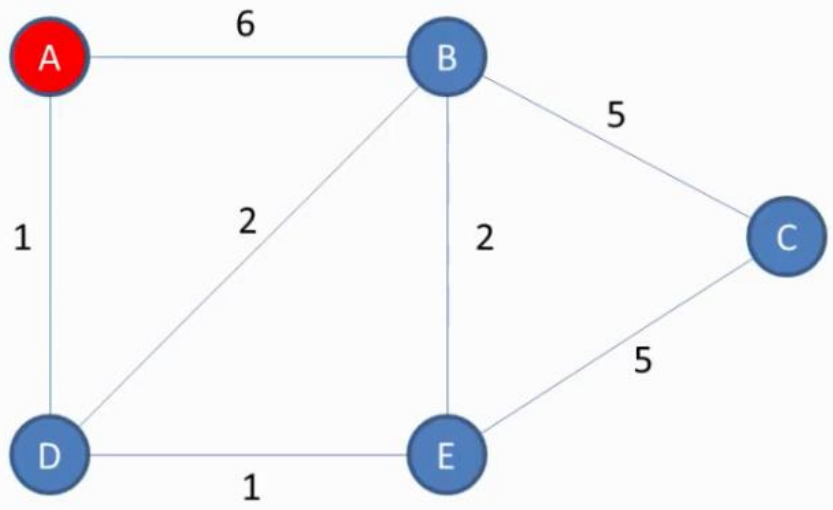


Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

Visited = [ ] Unvisited = [A, B, C, D, E]

Dijkstra's Algorithm
Example 1

- Step 2: Visit the unvisited vertex with the smallest known distance from the start vertex.
- At this stage, it is the start vertex itself A.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

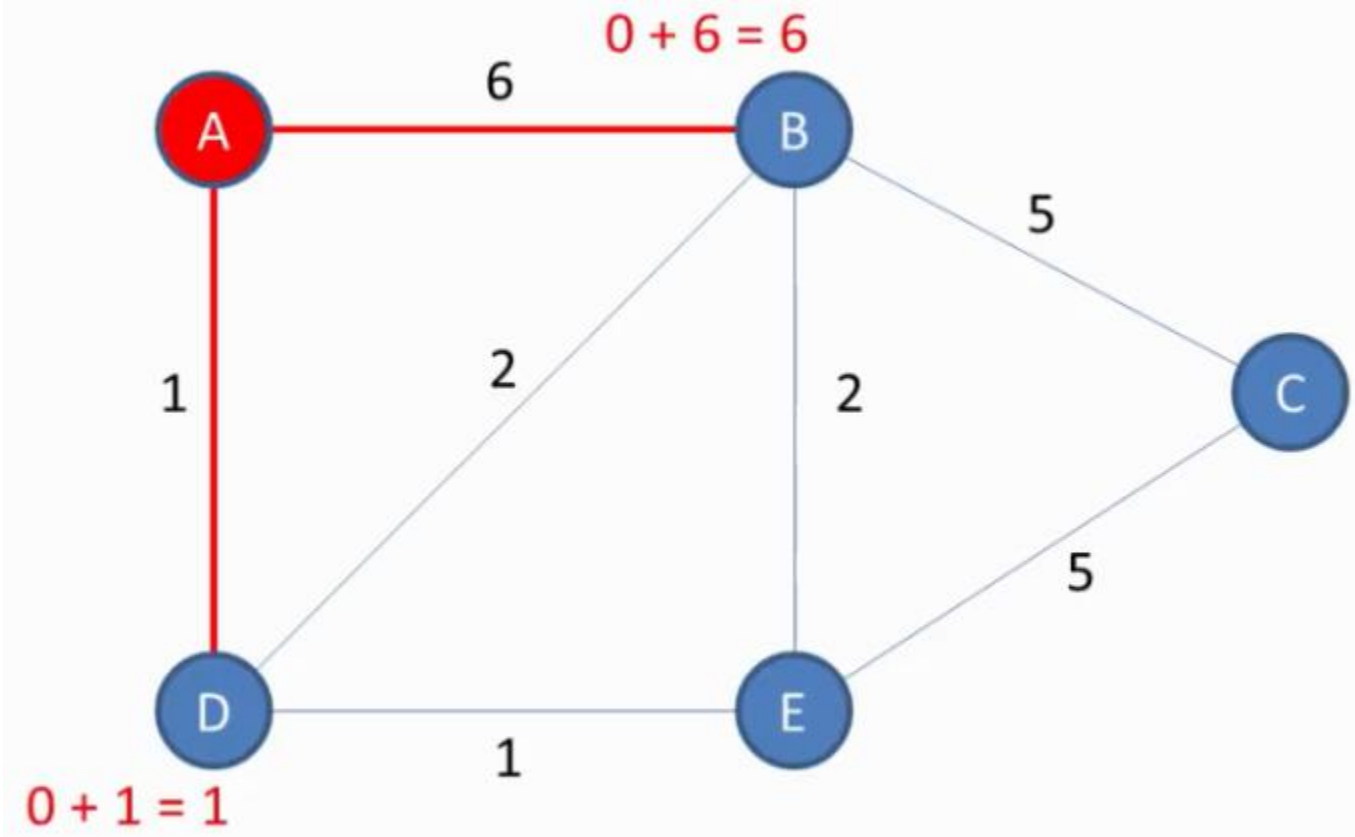
Visited = [ ]

Unvisited = [A, B, C, D, E]

Dijkstra's Algorithm
Example 1

- Step 3: for the current vertex, examine its unvisited neighbors.
  - We currently visiting A, and its unvisited neighbors are B and D, these are the vertices that A shared edges with.
  - Calculate the distance of each neighbor from the start vertex.
  - If the calculated distance of a vertex is less than the known distance, update the shortest distance in the table.
    - In this case update the previous vertex for each of the updated distances.

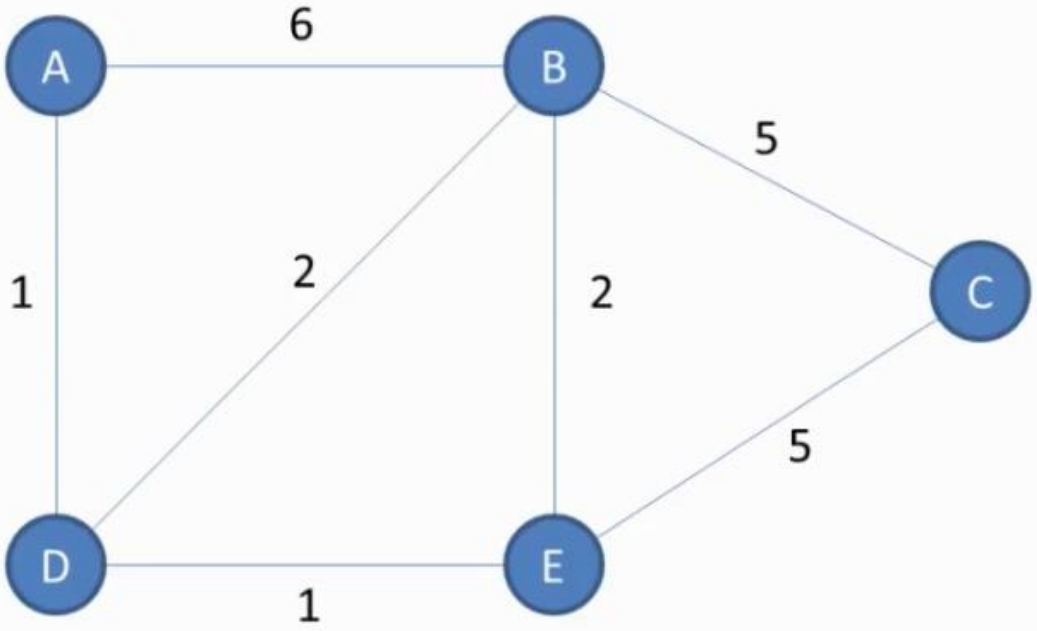
Dijkstra's Algorithm
Example 1



Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Dijkstra's Algorithm
Example 1

- Step 4: add the current vertex to the list of visited vertices.
- We will not visit this vertex again.



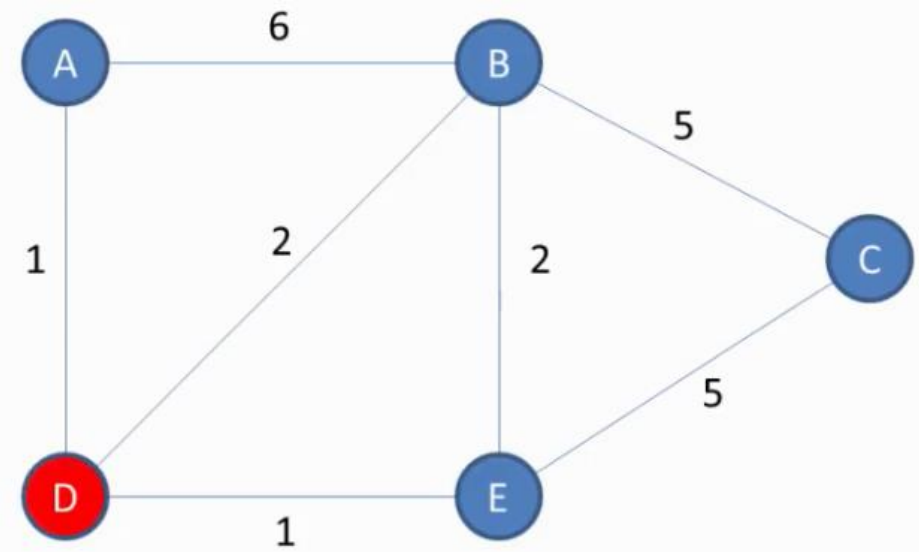
Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Visited = [A]

Unvisited = [B, C, D, E]

Dijkstra's Algorithm
Example 1

- Now the algorithm begins to repeat.
- Repeat the steps from step 2 to step 4.
- At this stage, follow step 2:



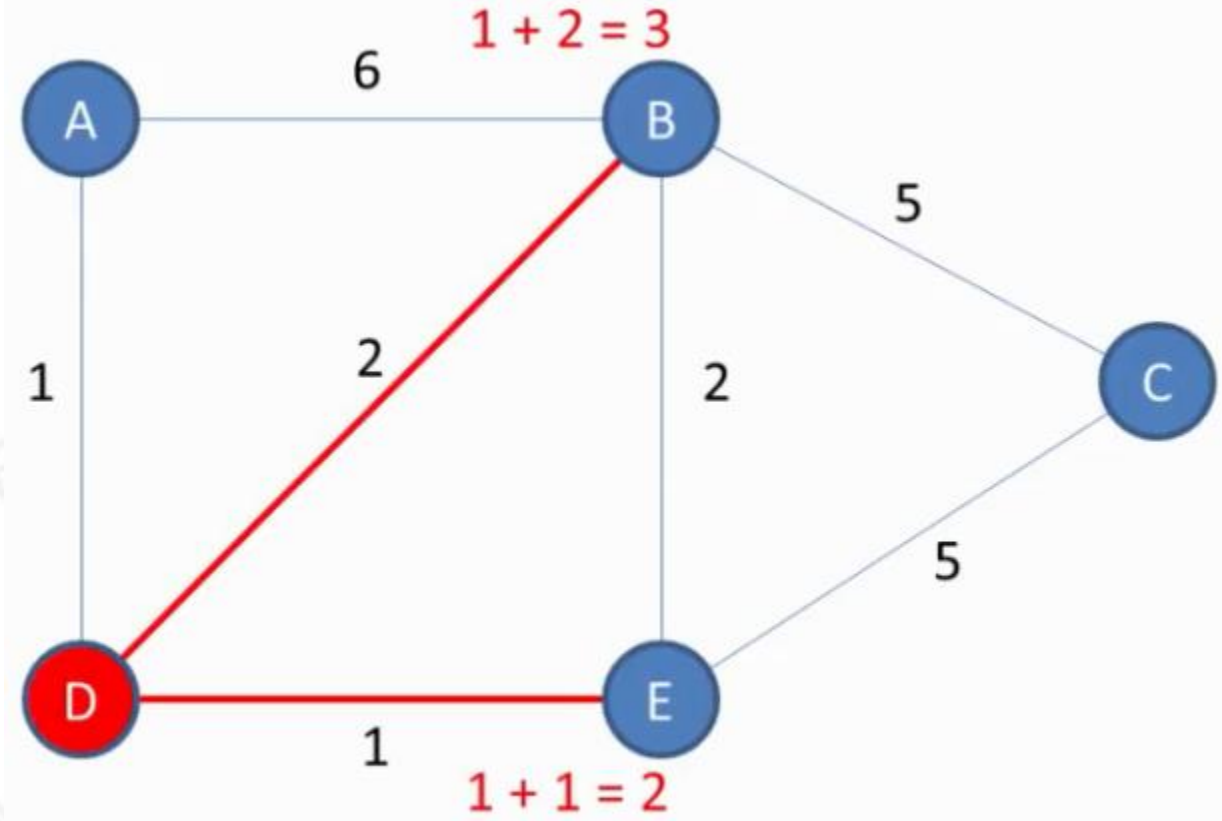
Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Visited = [A]

Unvisited = [B, C, D, E]

Dijkstra's Algorithm
Example 1

- Then step 3:



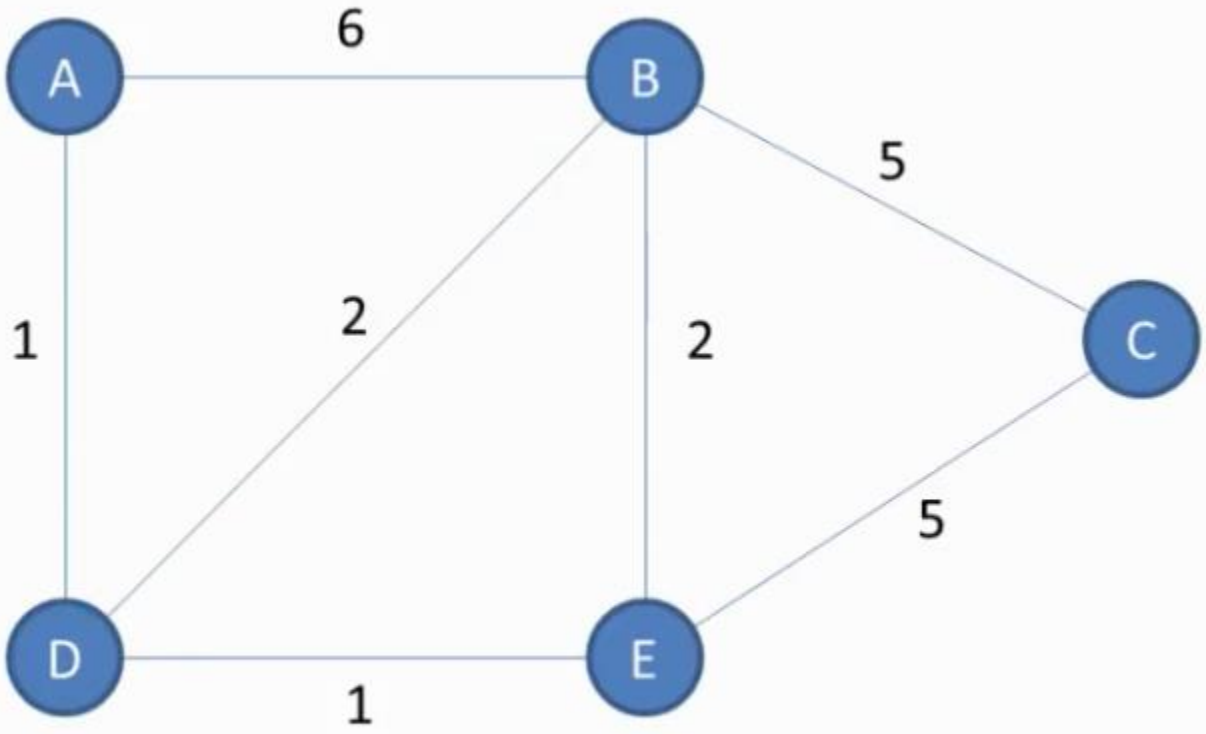
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

Visited = [A]

Unvisited = [B, C, D, E]

Dijkstra's Algorithm
Example 1

- Finally, step 4:



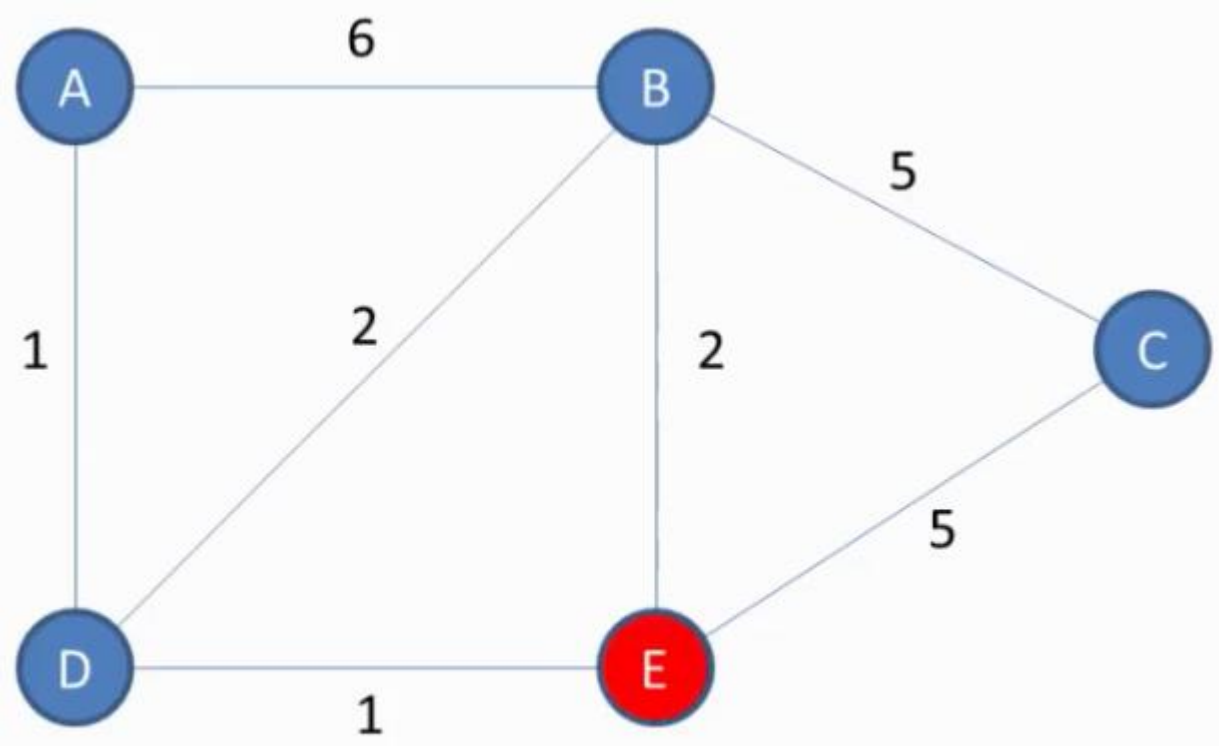
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

Visited = [A, D]

Unvisited = [B, C, E]

Dijkstra's Algorithm
Example 1

- Again, step 2:



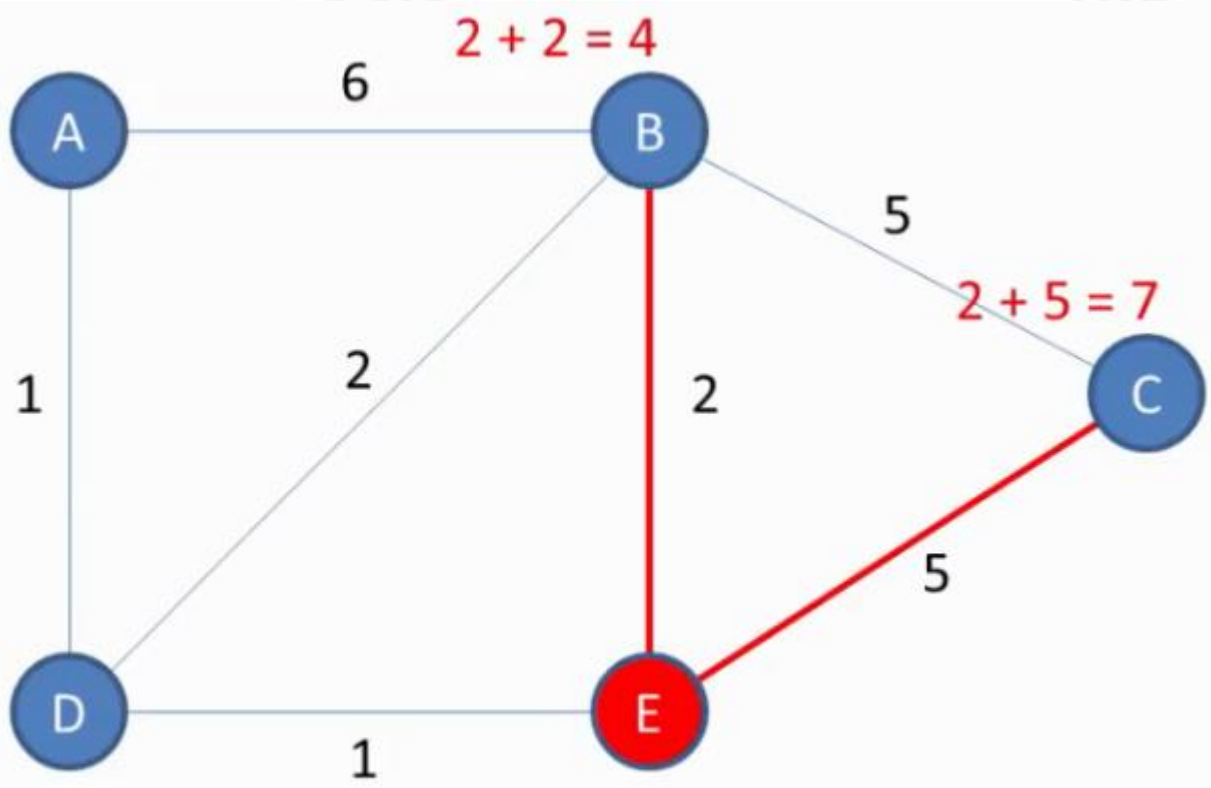
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

Visited = [A, D]

Unvisited = [B, C, E]

Dijkstra's Algorithm
Example 1

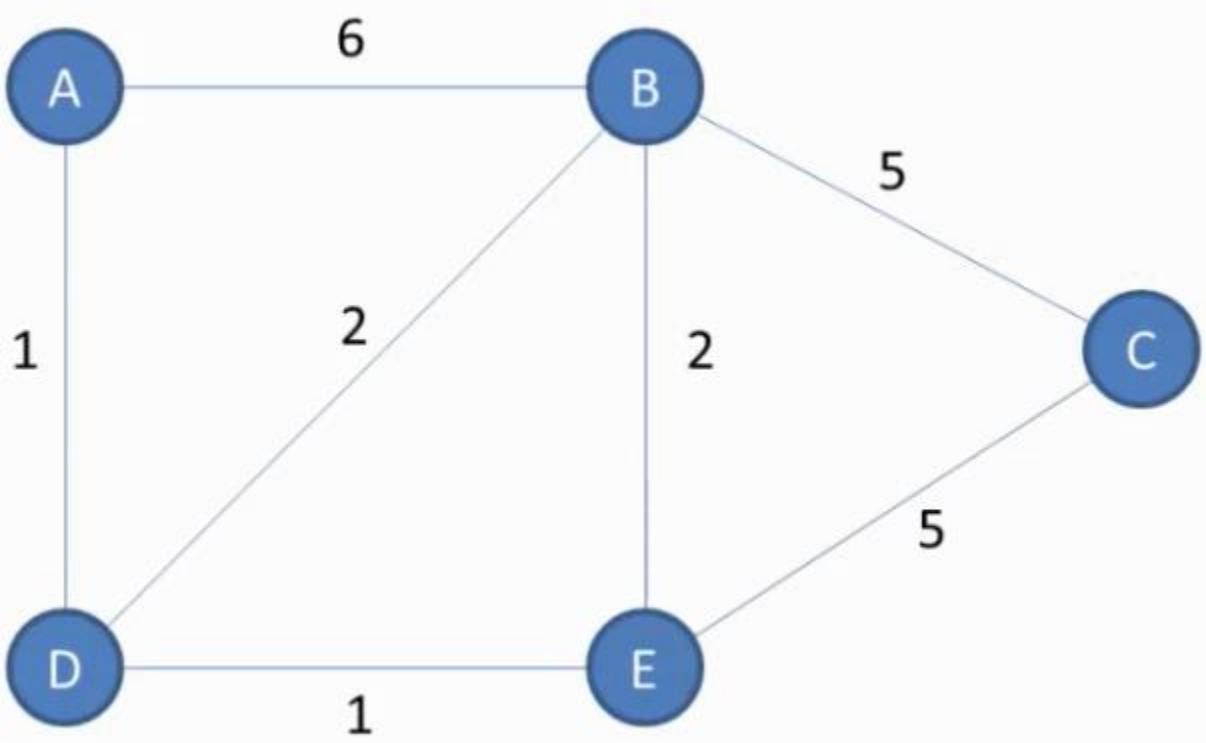
- then, step 3:



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Dijkstra's Algorithm
Example 1

- Finally, step 4:

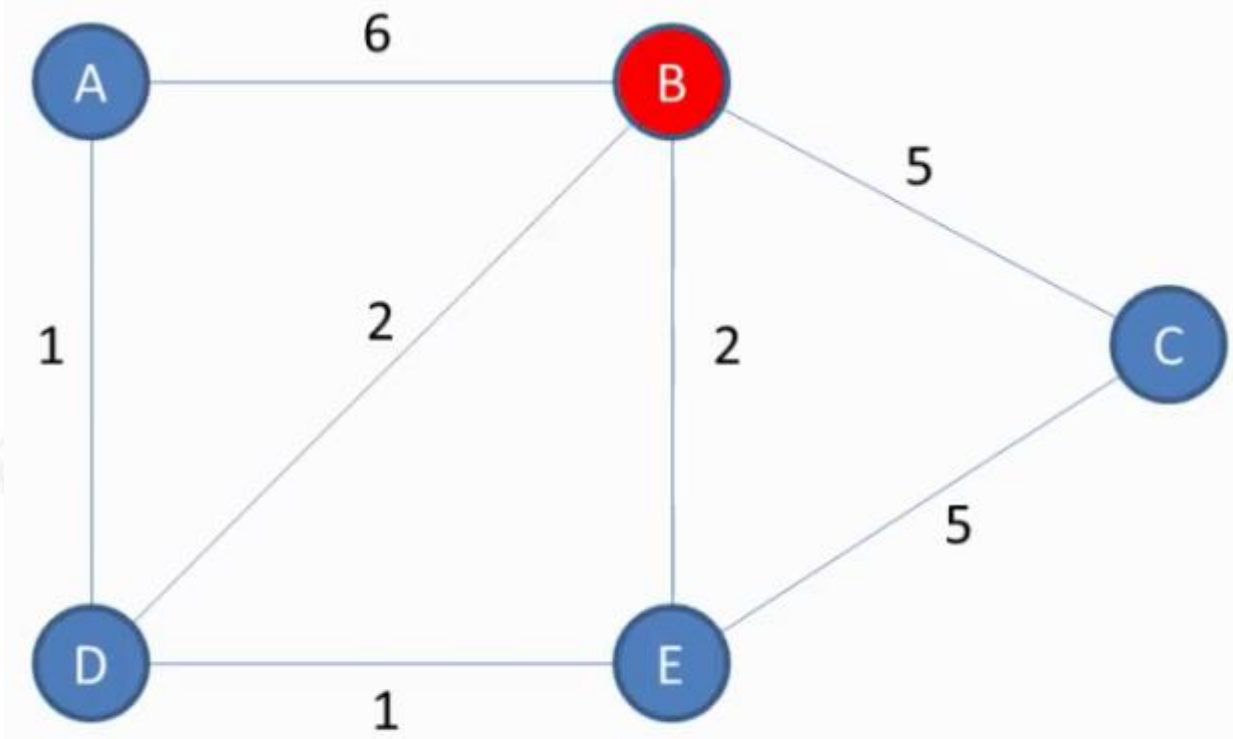


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E] Unvisited = [B, C]

Dijkstra's Algorithm
Example 1

- Again, step 2:



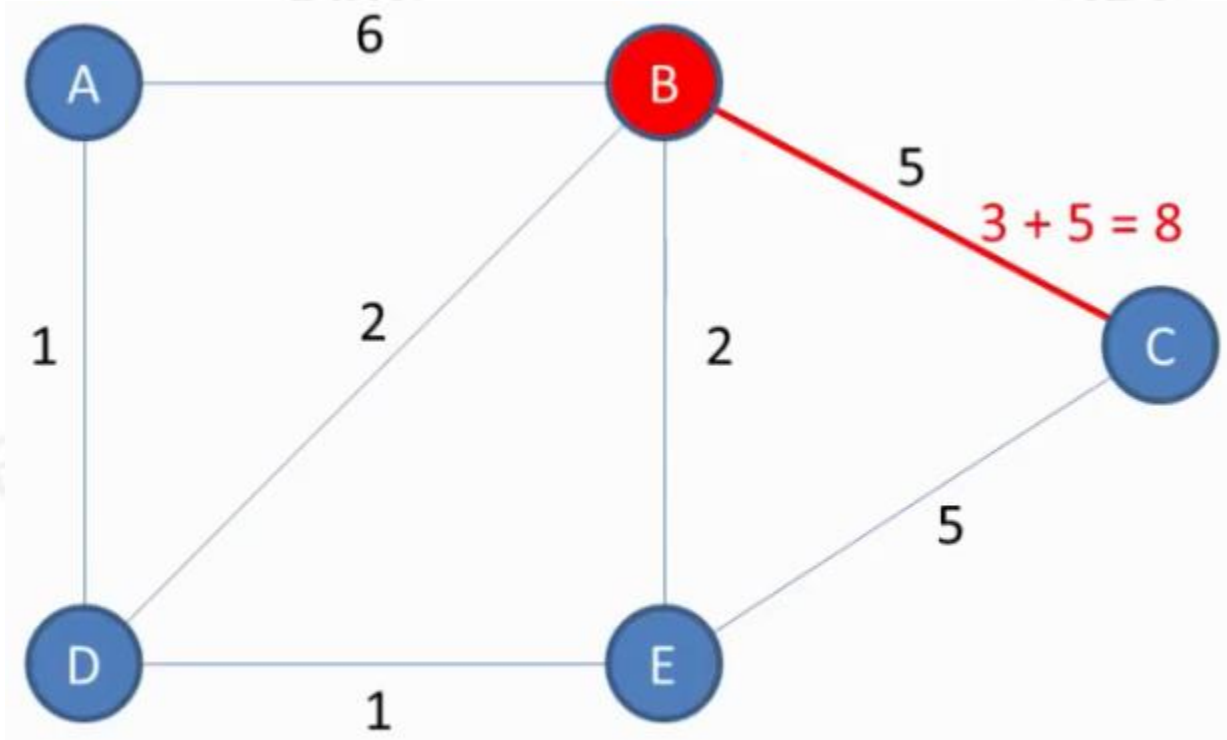
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E]

Unvisited = [B, C]

Dijkstra's Algorithm
Example 1

- Then, step 3:

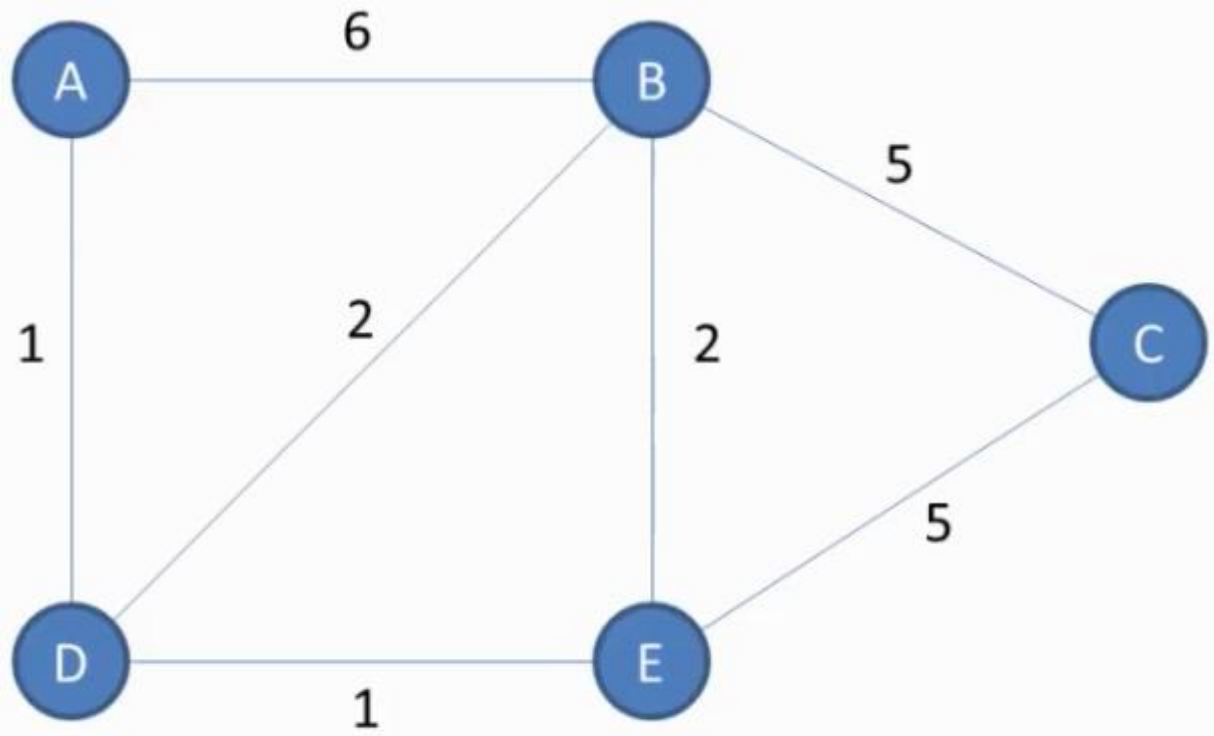


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E] Unvisited = [B, C]

Dijkstra's Algorithm
Example 1

- Finally, step 4:

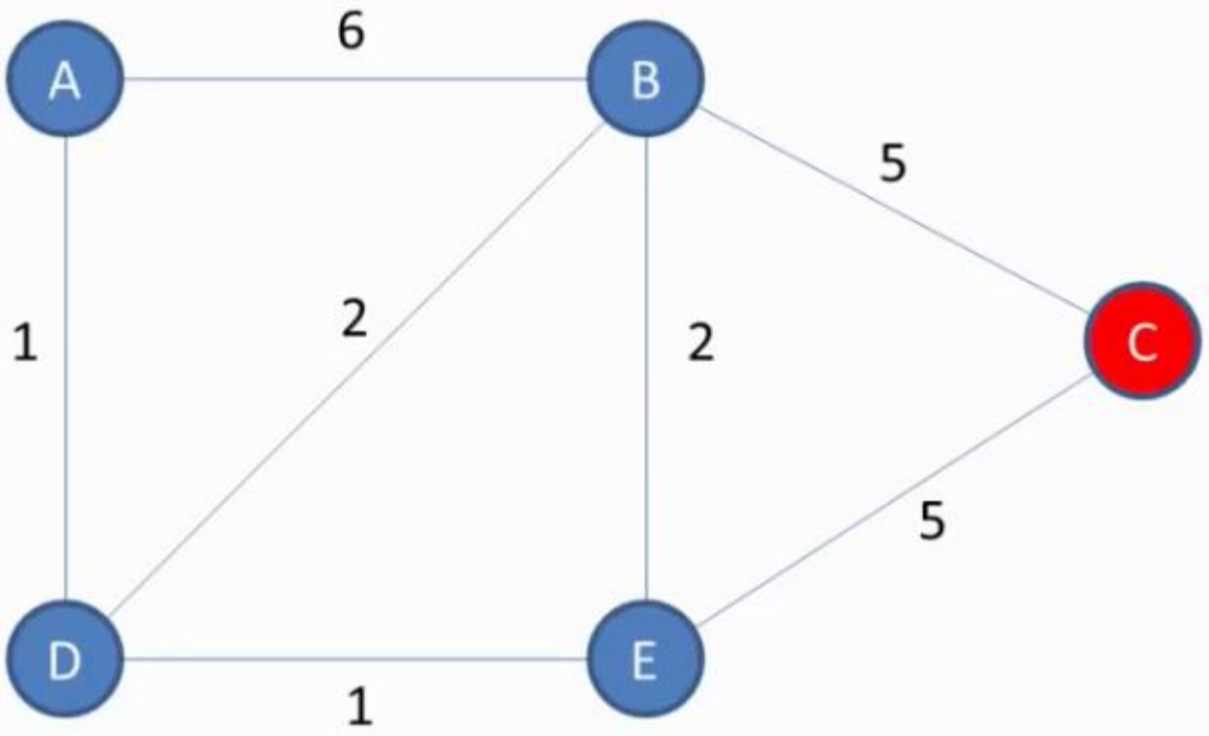


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E, B] Unvisited = [C]

Dijkstra's Algorithm
Example 1

- Again, step 2:

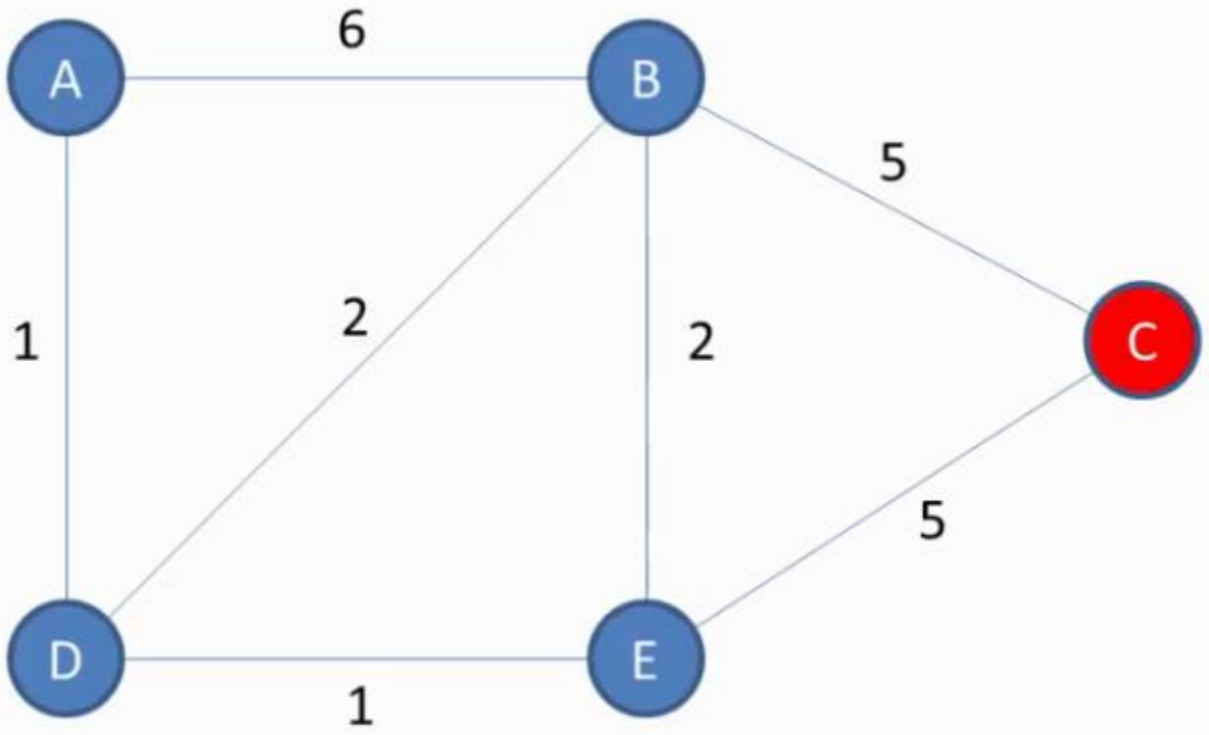


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E, B] Unvisited = [C]

Dijkstra's Algorithm
Example 1

- Then, step 3:

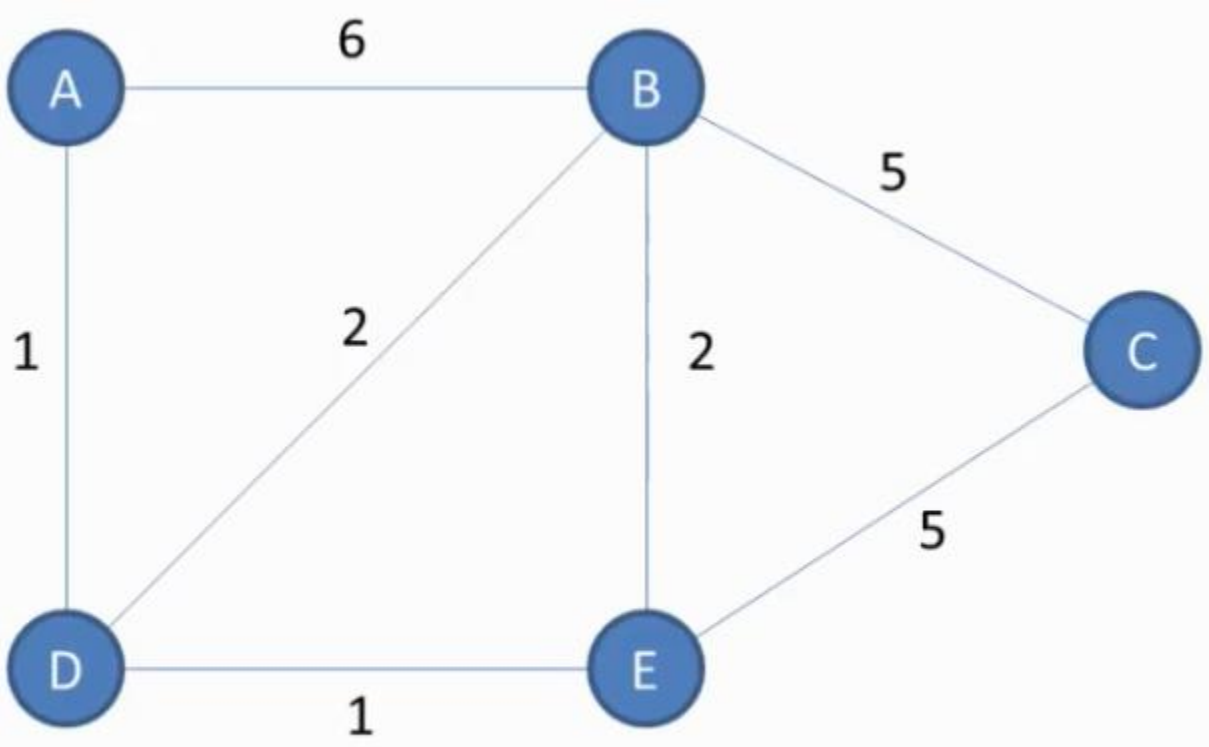


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E, B] Unvisited = [C]

Dijkstra's Algorithm
Example 1

- Finally, step 4:



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Visited = [A, D, E, B, C] Unvisited = []

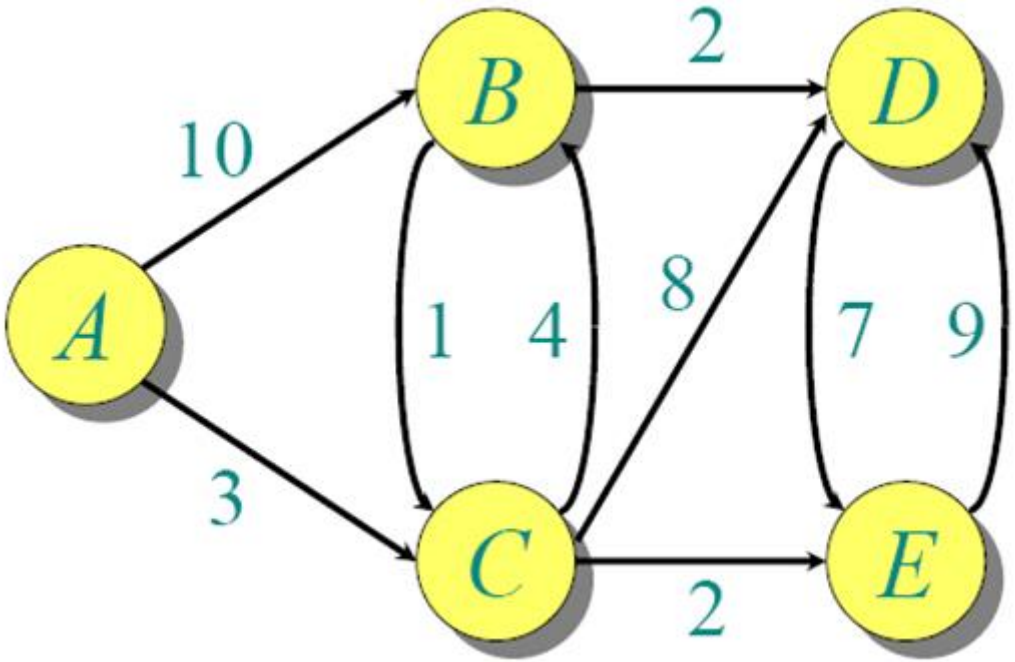
Dijkstra's Algorithm
Example 1

- There is no more vertices to visit, so our table is complete.

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

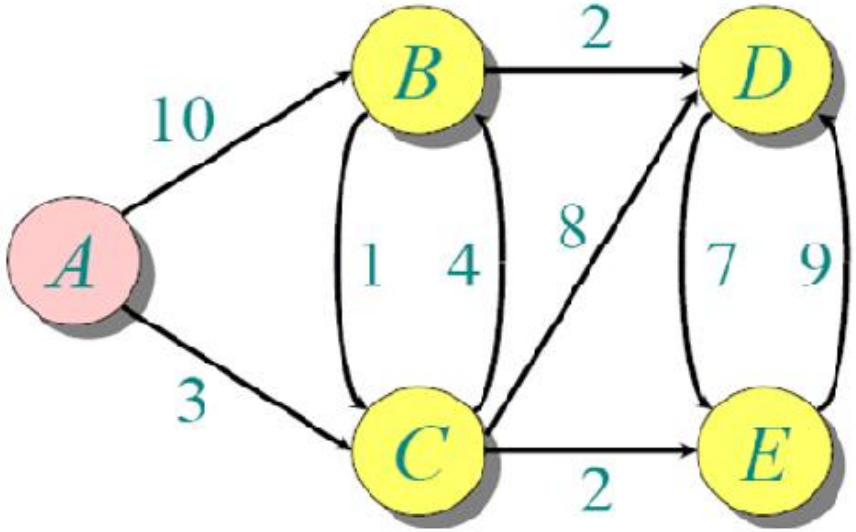
Dijkstra's Algorithm
Example 2

- Consider this simple graph, our objective is to find is to find the shortest path from A to all other vertices.



Dijkstra's Algorithm
Example 2

- Step 1: Consider the starting vertex A.
  - Distance from A to A is 0.
  - Distances to all other vertices from A are unknown, therefore ∞.
  - We will include these information in our table directly.

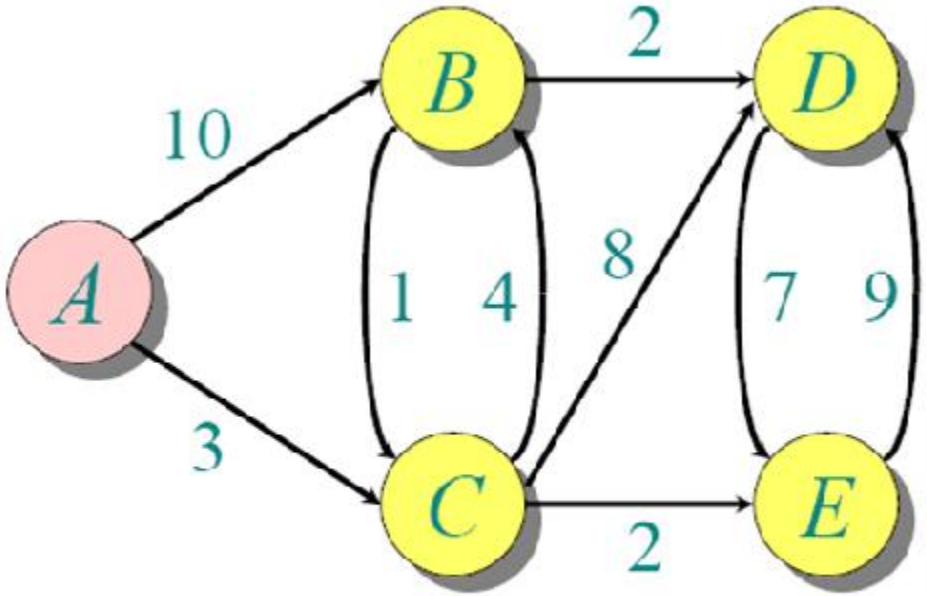


Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

Visited = [ ] Unvisited = [A, B, C, D, E]

Dijkstra's Algorithm
Example 2

- Step 2: Visit the unvisited vertex with the smallest known distance from the start vertex.
- At this stage, it is the start vertex itself A.



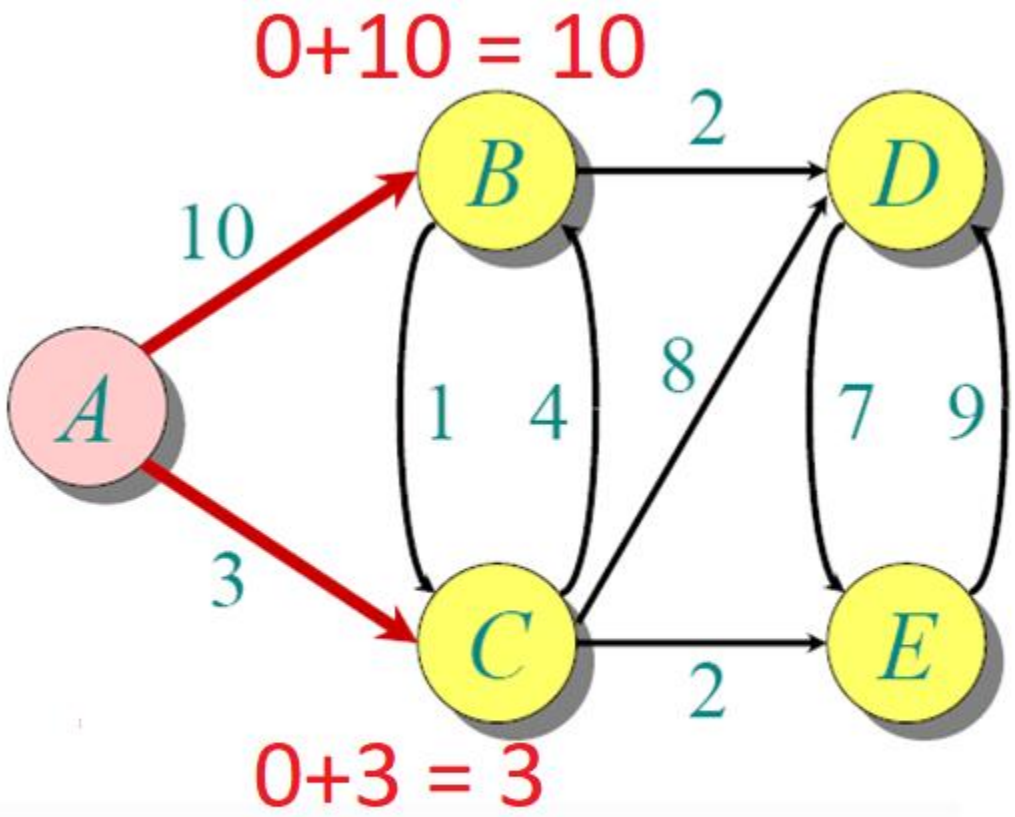
Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

Visited = [ ] Unvisited = [A, B, C, D, E]

Dijkstra's Algorithm
Example 2

- Step 3: for the current vertex, examine its unvisited neighbors.
  - We currently visiting A, and its unvisited neighbors are B and C, these are the vertices that A shared edges with.
  - Calculate the distance of each neighbor from the start vertex.
  - If the calculated distance of a vertex is less than the known distance, update the shortest distance in the table.
    - In this case update the previous vertex for each of the updated distances.

Dijkstra's Algorithm
Example 2



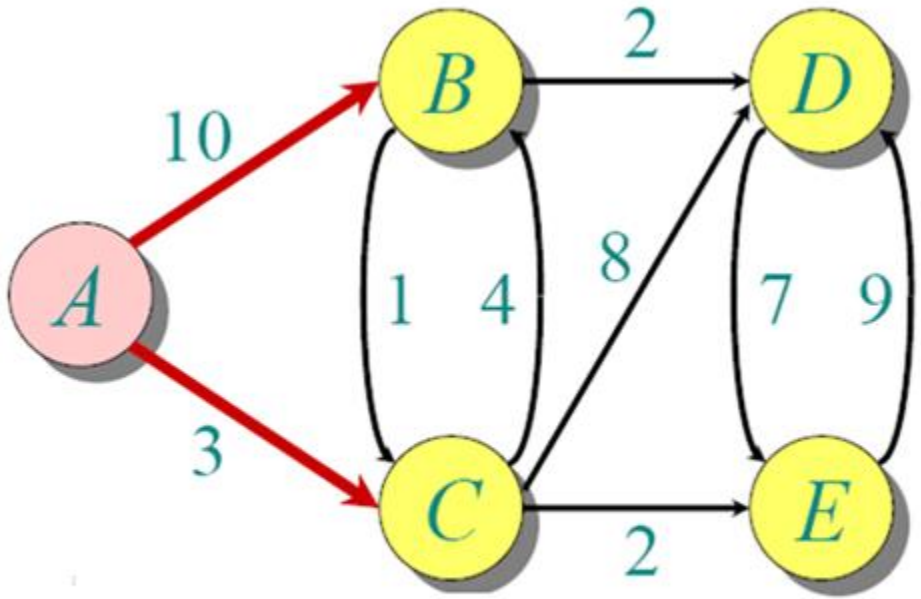
Vertex	Shortest distance from A	Previous vertex
A	0	
B	10	A
C	3	A
D	∞	
E	∞	

Visited = [ ]

Unvisited = [A, B, C, D, E]

Dijkstra's Algorithm
Example 2

- Step 4: add the current vertex to the list of visited vertices.
- We will not visit this vertex again.



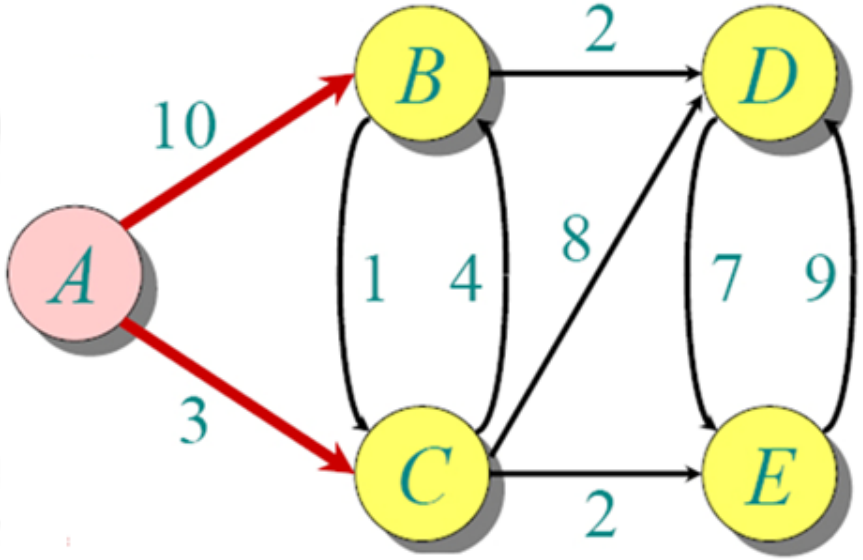
Vertex	Shortest distance from A	Previous vertex
A	0	
B	10	A
C	3	A
D	∞	
E	∞	

Visited = [A]

Unvisited = [B, C, D, E]

Dijkstra's Algorithm
Example 2

- Now the algorithm begins to repeat.
- Repeat the steps from step 2 to step 4.
- At this stage, follow step 2:

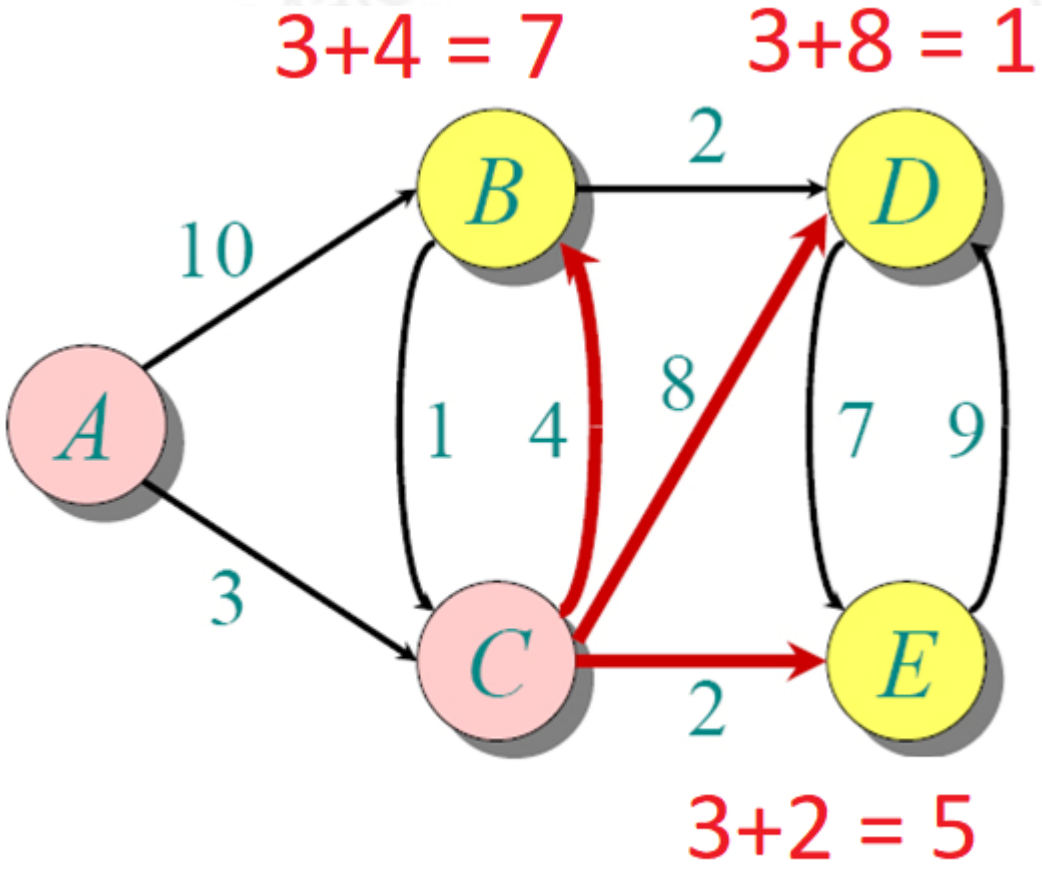


Vertex	Shortest distance from A	Previous vertex
A	0	
B	10	A
C	3	A
D	∞	
E	∞	

Visited = [A] Unvisited = [B, C, D, E]

Dijkstra's Algorithm
Example 2

- Then step 3:



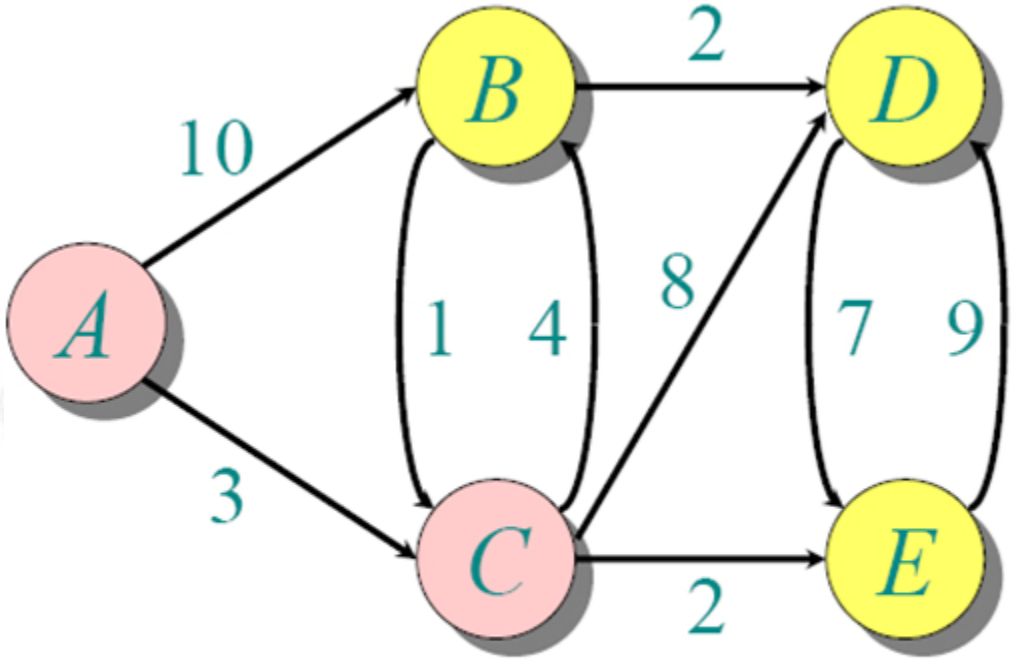
Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	11	C
E	5	C

Visited = [A]

Unvisited = [B, C, D, E]

Dijkstra's Algorithm
Example 2

- Finally, step 4:

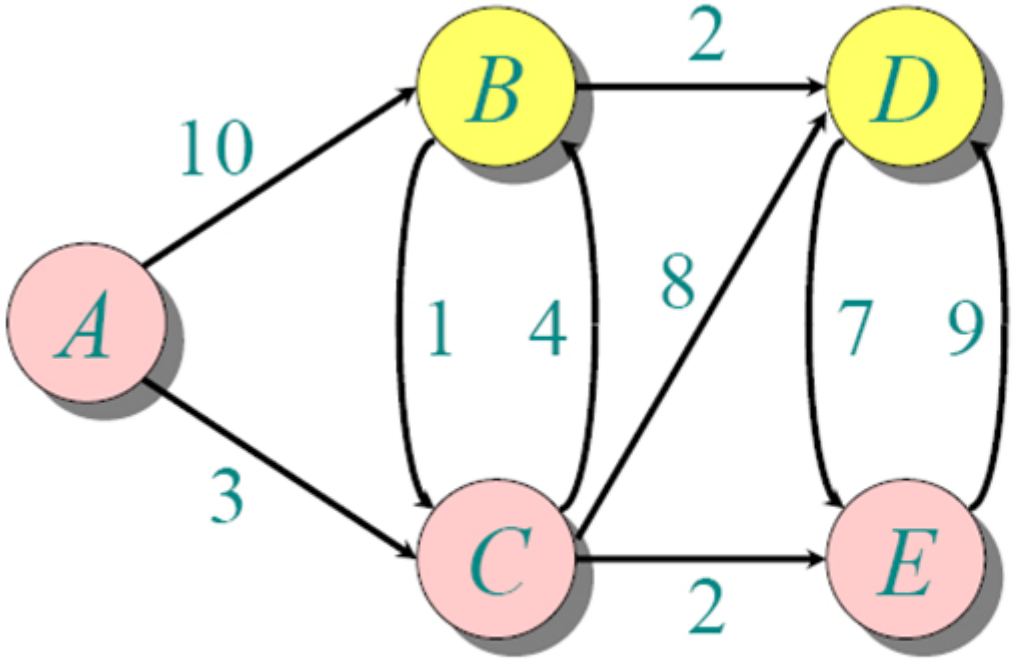


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	11	C
E	5	C

Visited = [A, C] Unvisited = [B, D, E]

Dijkstra's Algorithm
Example 2

- Again, step 2:

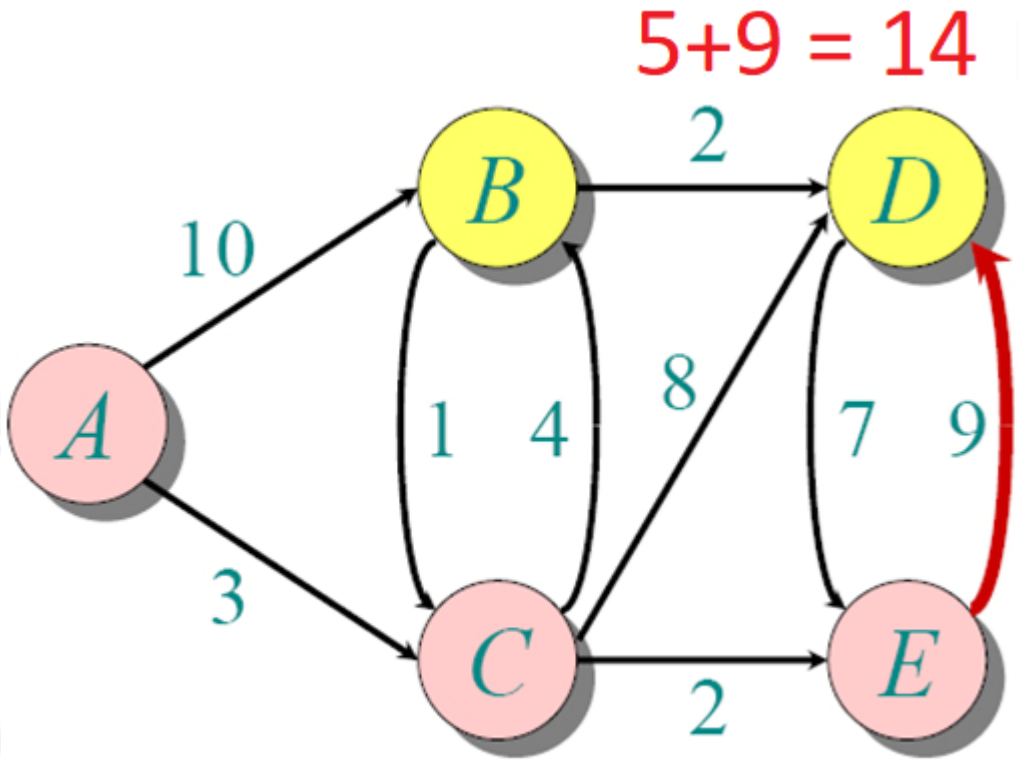


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	11	C
E	5	C

Visited = [A, C] Unvisited = [B, D, E]

Dijkstra's Algorithm
Example 2

- then, step 3:

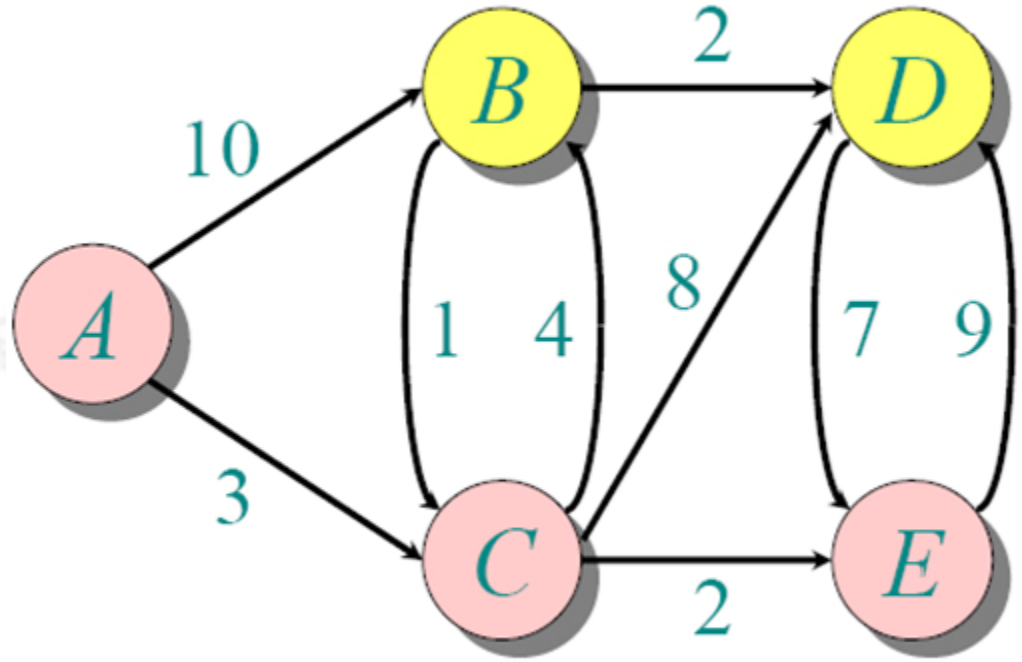


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	11	C
E	5	C

Visited = [A, C] Unvisited = [B, D, E]

Dijkstra's Algorithm
Example 2

- Finally, step 4:

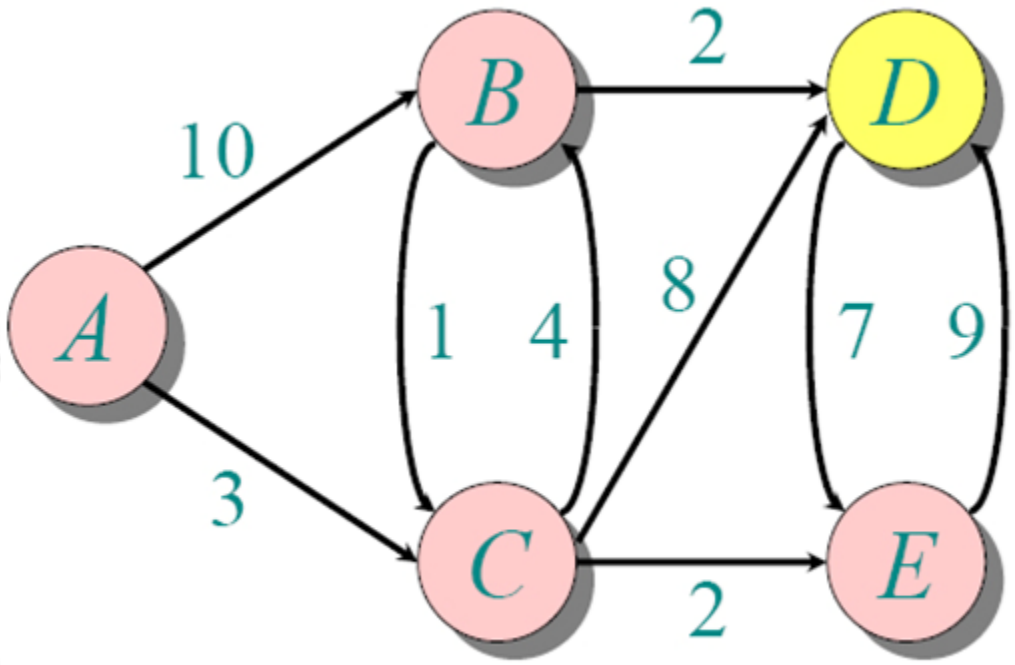


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	11	C
E	5	C

Visited = [A, C, E] Unvisited = [B, D]

Dijkstra's Algorithm
Example 2

- Again, step 2:

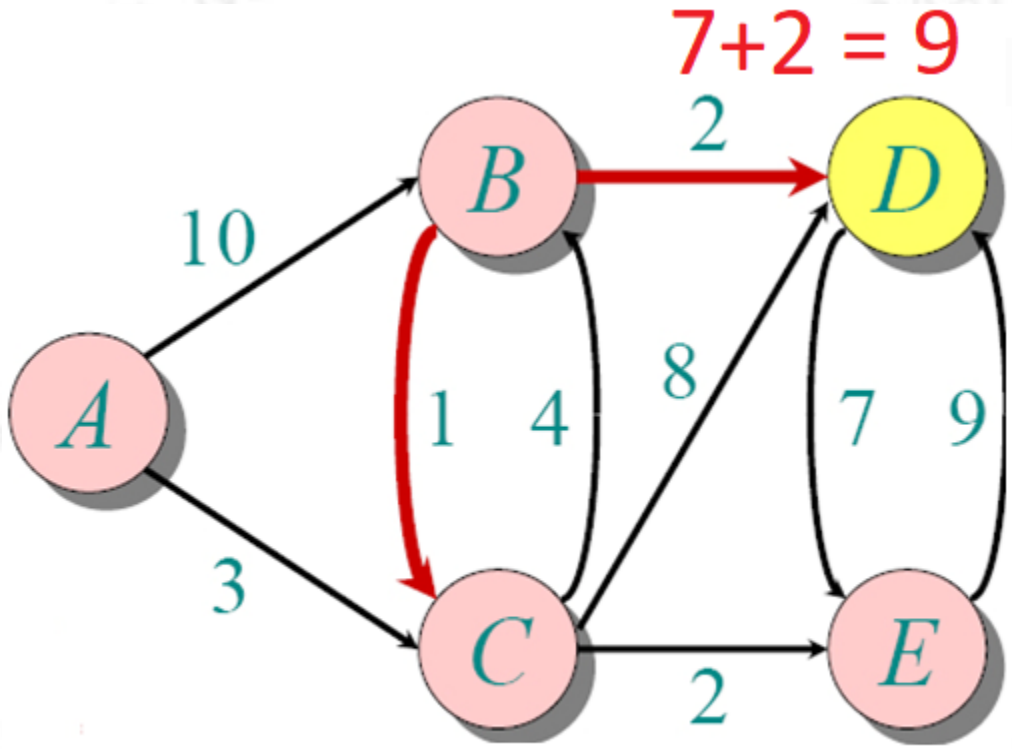


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	11	C
E	5	C

Visited = [A, C, E] Unvisited = [B, D]

Dijkstra's Algorithm
Example 2

- Then, step 3:

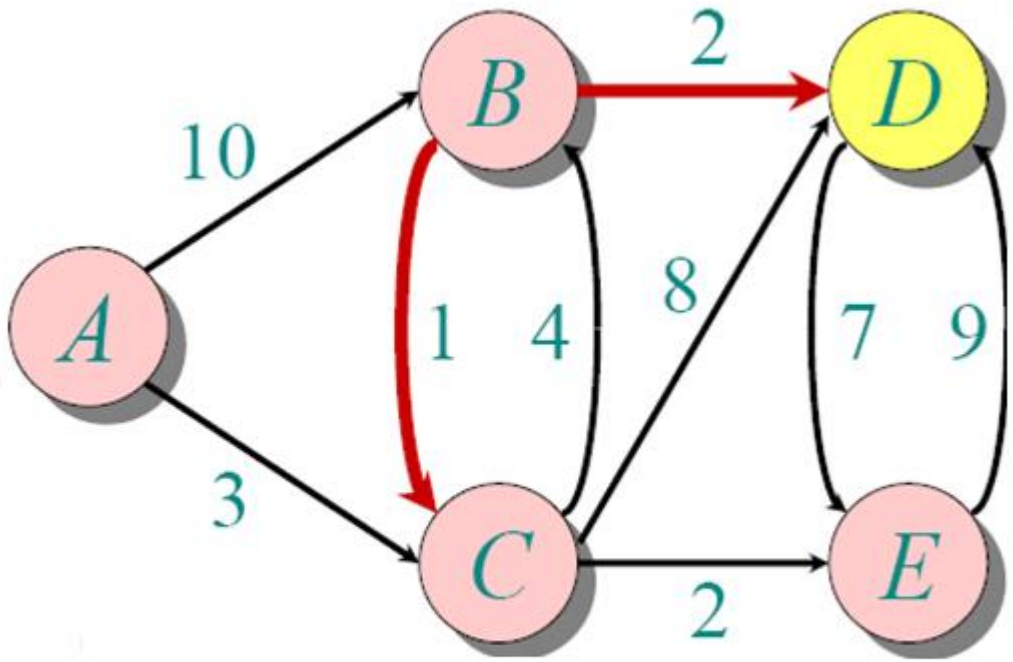


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	9	B
E	5	C

Visited = [A, C, E] Unvisited = [B, D]

Dijkstra's Algorithm
Example 2

- Finally, step 4:

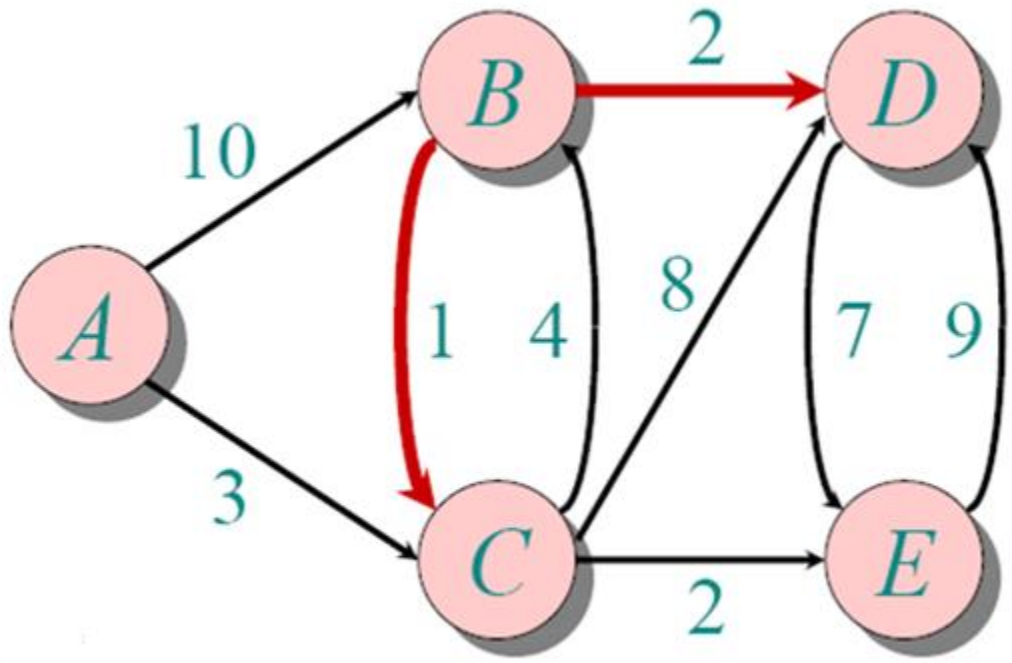


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	9	B
E	5	C

Visited = [A, C, E, B] Unvisited = [D]

Dijkstra's Algorithm
Example 2

- Again, step 2:

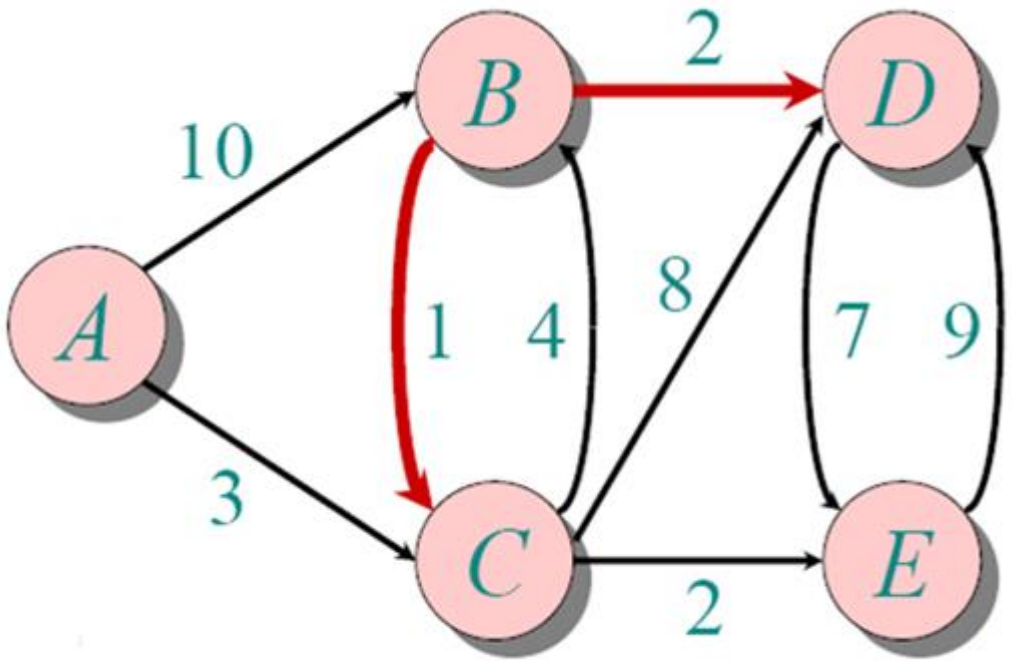


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	9	B
E	5	C

Visited = [A, C, E, B] Unvisited = [D]

Dijkstra's Algorithm
Example 1

- Then, step 3:

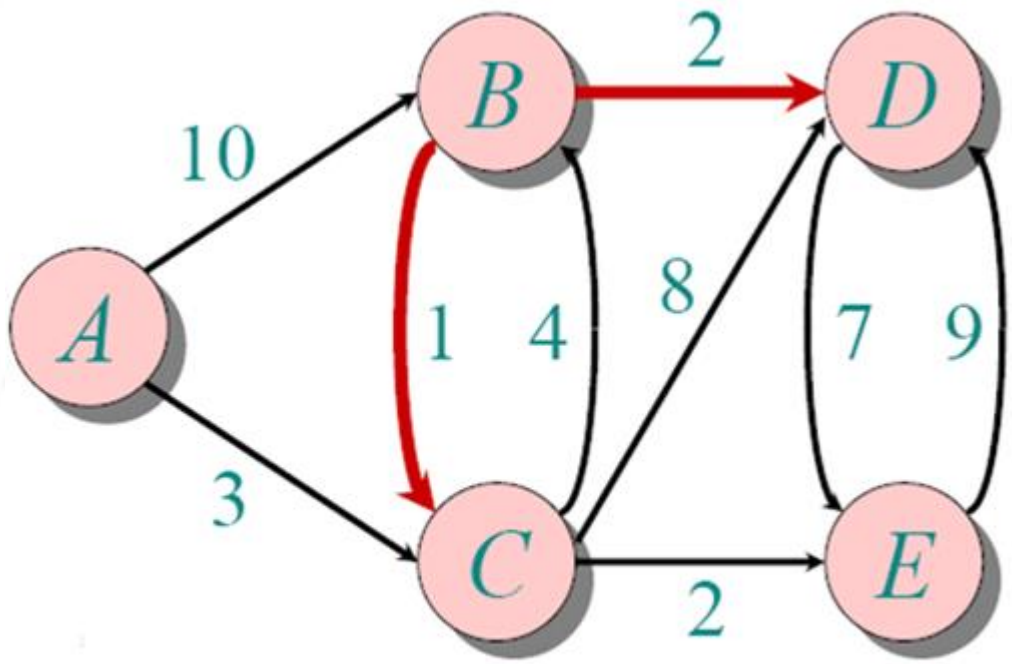


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	9	B
E	5	C

Visited = [A, C, E, B] Unvisited = [D]

Dijkstra's Algorithm
Example 1

- Finally, step 4:

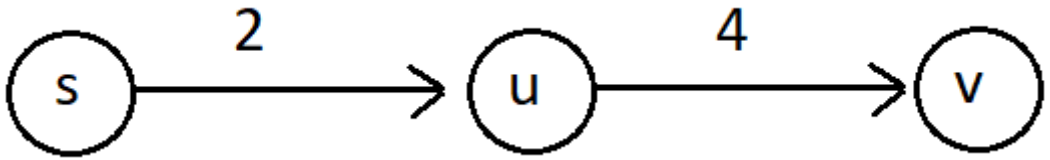


Vertex	Shortest distance from A	Previous vertex
A	0	
B	7	C
C	3	A
D	9	B
E	5	C

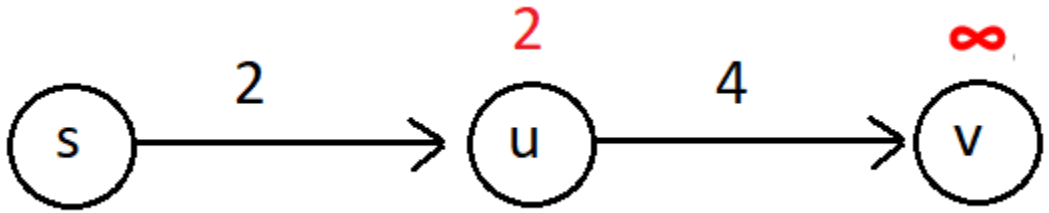
Visited = [A, C, E, B, D] Unvisited = []

Relaxation

- If we have the following simple graph:



- If **s** is the starting vertex, initially distance of u (**d[u]**) is 2 and distance of v (**d[v]**) is ∞ as shown:



Relaxation

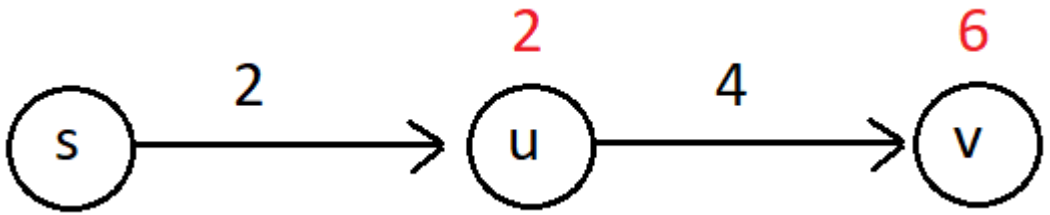
- While visiting vertex u, we will try to update the distance of v:

```
if(d[u] + c(u,v) < d[v])
{
        d[v] = d[u] + c(u,v)
}
```

- Where **c(u,v)** is the cost of the edge between u and v.

Relaxation

- In this case, we will update the d[v] to 6:



- This update is called **Relaxation**.
- Whenever we are visiting a vertex, we will try to relax the distance to other vertices that connected to this vertex.

Dijkstra’s Algorithm

- Dijkstra’s shortest path algorithm is an example of a greedy method.
- The algorithm chooses the unvisited vertex with the smallest known distance from the start vertex.

Dijkstra’s Algorithm
Time Complexity

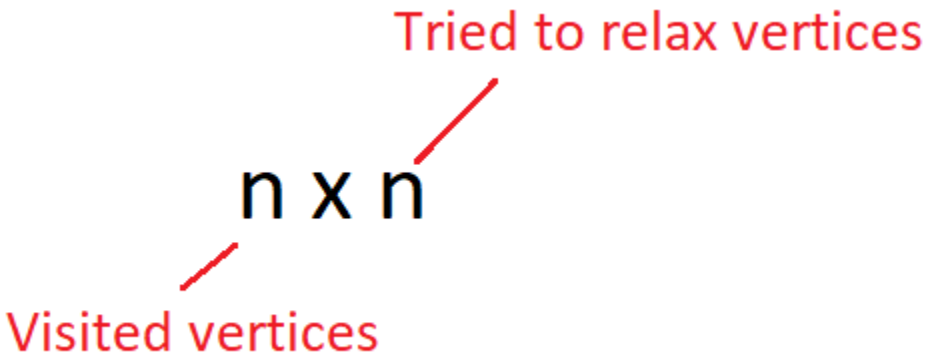
- Dijkstra's Algorithm purpose is to find the shortest path to all vertices, it will visit all vertices, if we suppose that **n** is the number of vertices.

$$n = |v|$$

- While visiting each vertex, it is trying to relax the path of all other connected vertices to the currently visited vertex.
- The total number of the connected vertices to a vertex is depending on the graph.

Dijkstra's Algorithm
Time Complexity

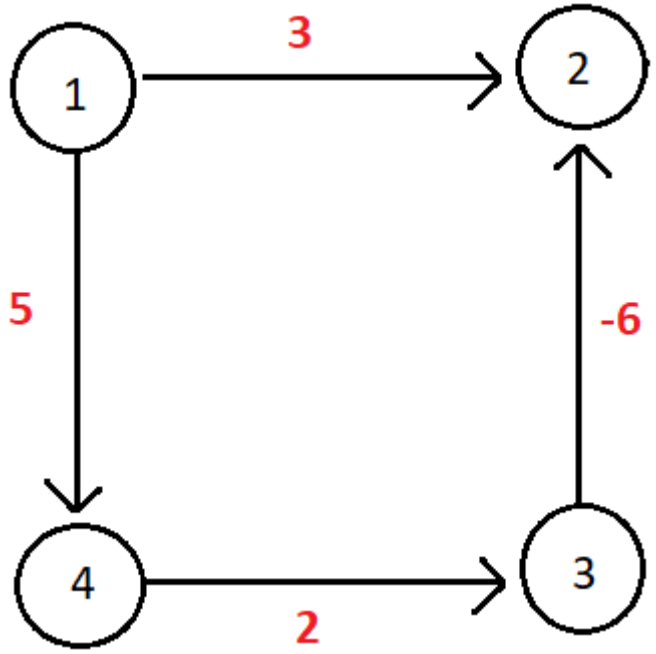
- At most a vertex can be connected to all other vertices.
- So, while visiting a vertex, at most it will try to relax **n** vertices.
- It will take $n \times n$ where:



- So, the time compacity is for this algorithm is **Quadratic time,** or $O(n^2)$
- Also, We can say $O(|V|^2)$

Dijkstra's Algorithm Drawback

- The drawback of this algorithm is cannot handle negative weighted edges, it may work or may not work in this case.



Bellman–Ford Algorithm

- If there are negative edges in a graph, we cannot use Dijkstra's algorithm and confirms that the result is correct.
- In this case, we want an algorithm which confirms that the result is correct, this algorithm is Bellman-Ford Algorithm.
- It doesn't follow greedy method strategy; it follows a different strategy.

Bellman–Ford Algorithm

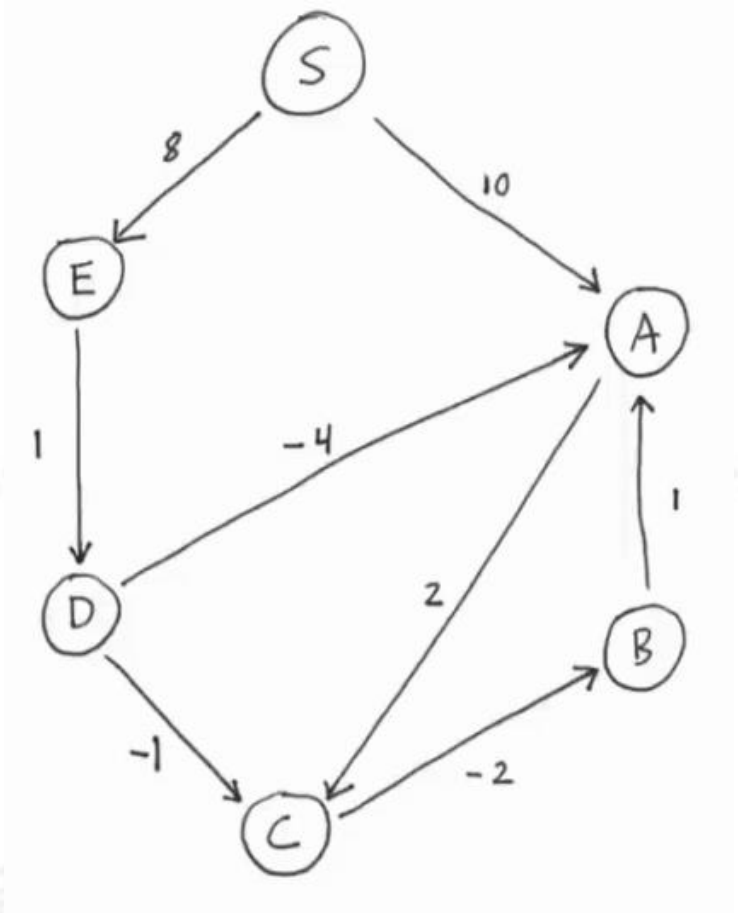
- The Bellman–Ford algorithm and Dijkstra's algorithm have the same result.
- They both give you the shortest path from one vertex to all other vertices in a graph.
- The most important difference between the two algorithms is:
  - Bellman-Ford works on graphs with negative edge weights, while Dijkstra's does not.

Bellman–Ford Algorithm

- The algorithm says that go on repeatedly relaxing all the edges for (|V| - 1) times.
- For example, if the number of vertices in a graph is 5, we should relax all the edges 4 times.

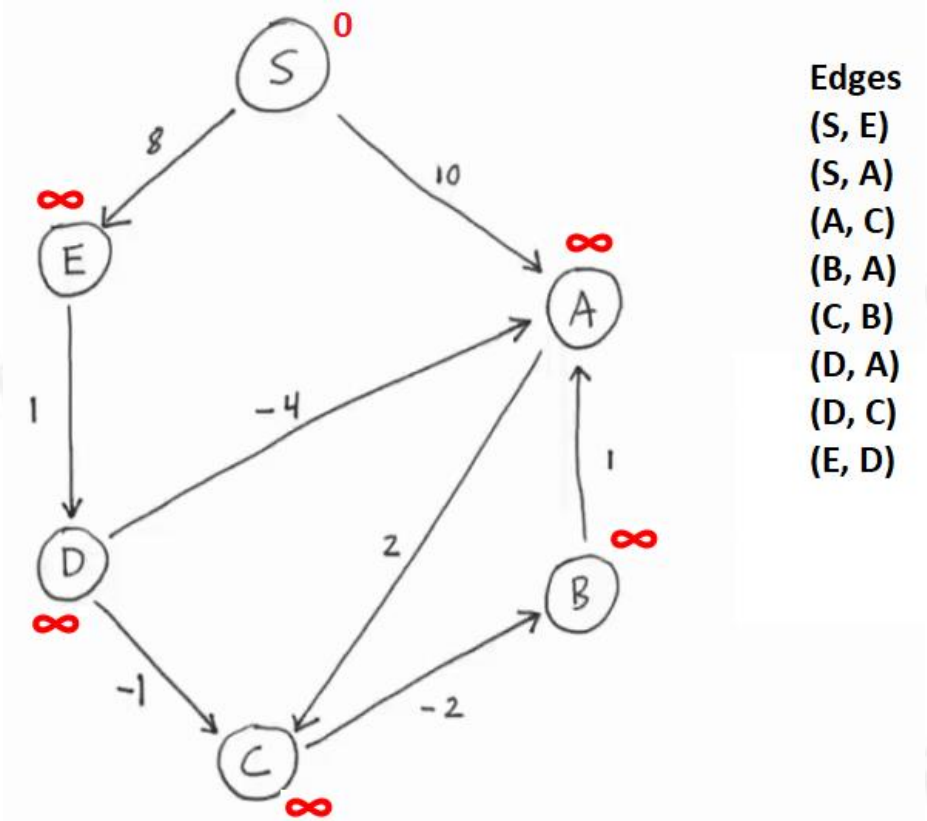
Bellman–Ford Algorithm
Example 1

- We are given the following graph, and S is our starting vertex.



Bellman–Ford Algorithm
Example 1

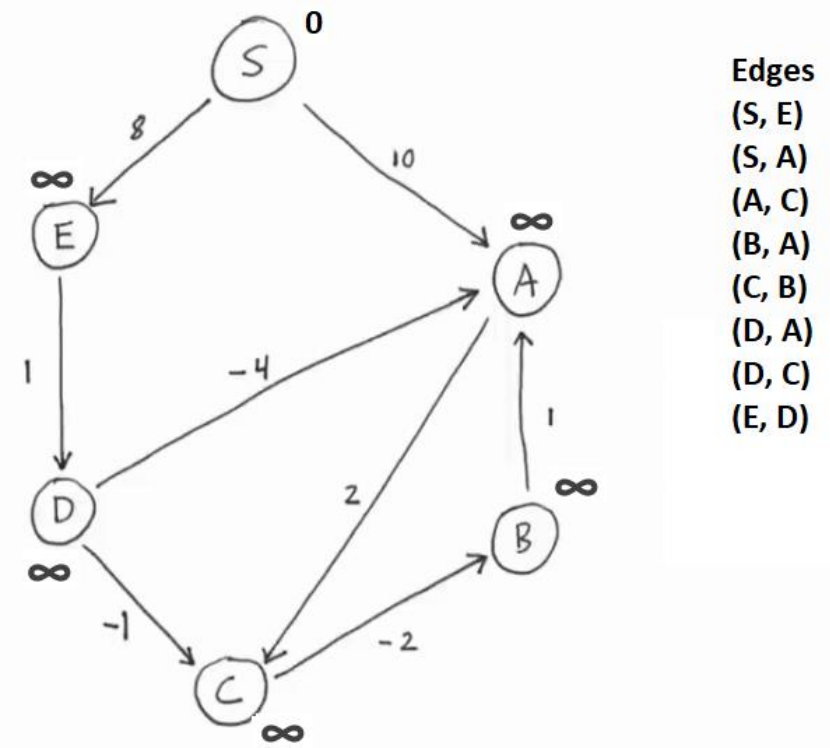
- Step 1: **prepare** a list of all edges in the graph and **set** the distance of the starting vertex to 0 and all other vertices to ∞.



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

- Step 2: **relax** all the edges (|V| - 1) times.
- We have 6 vertices, then we should relax all the edges 5 times.

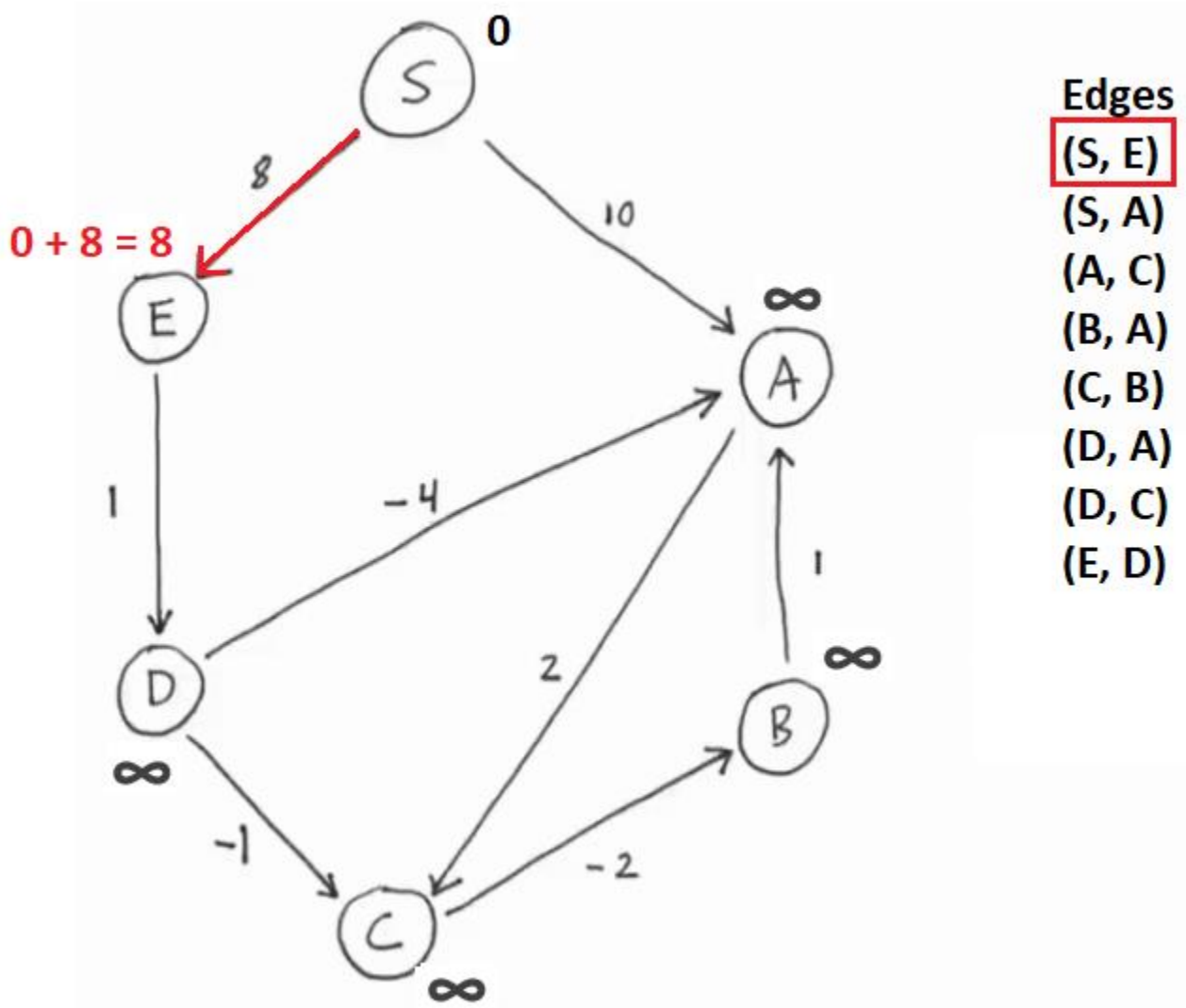


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

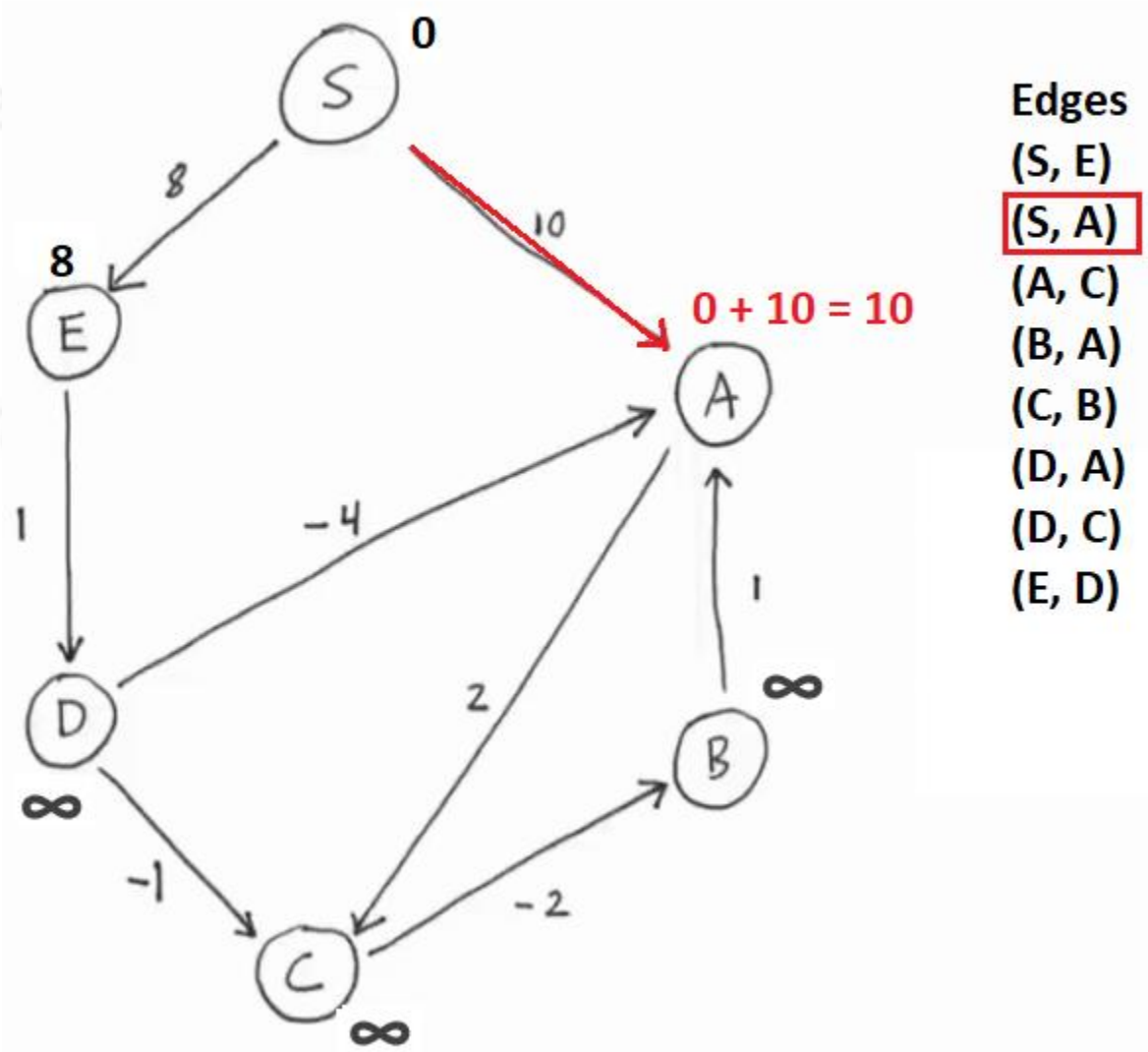
1st iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

1st iteration

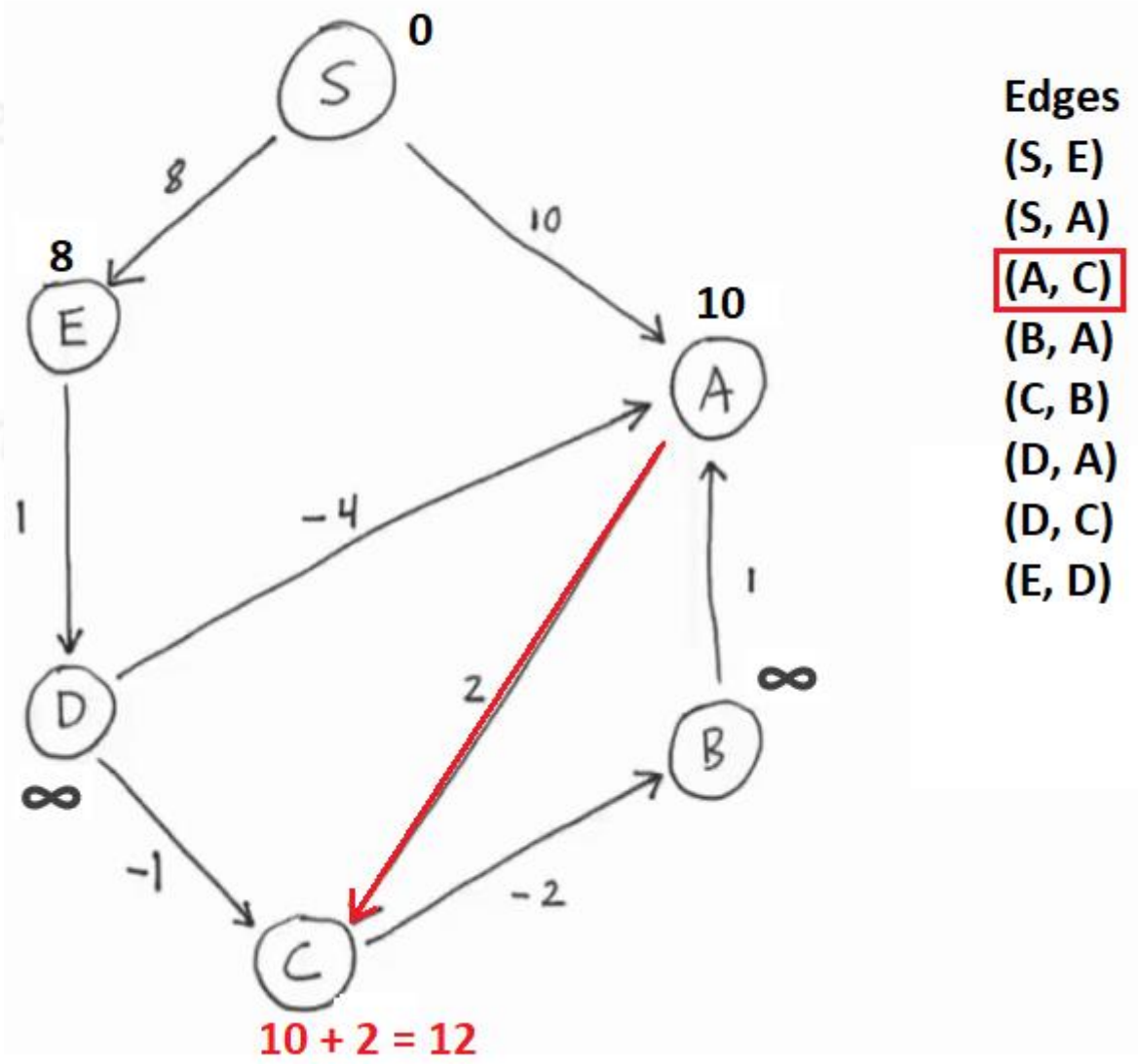


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

1st iteration



Edges

(S, E)

(S, A)

(A, C)

(B, A)

(C, B)

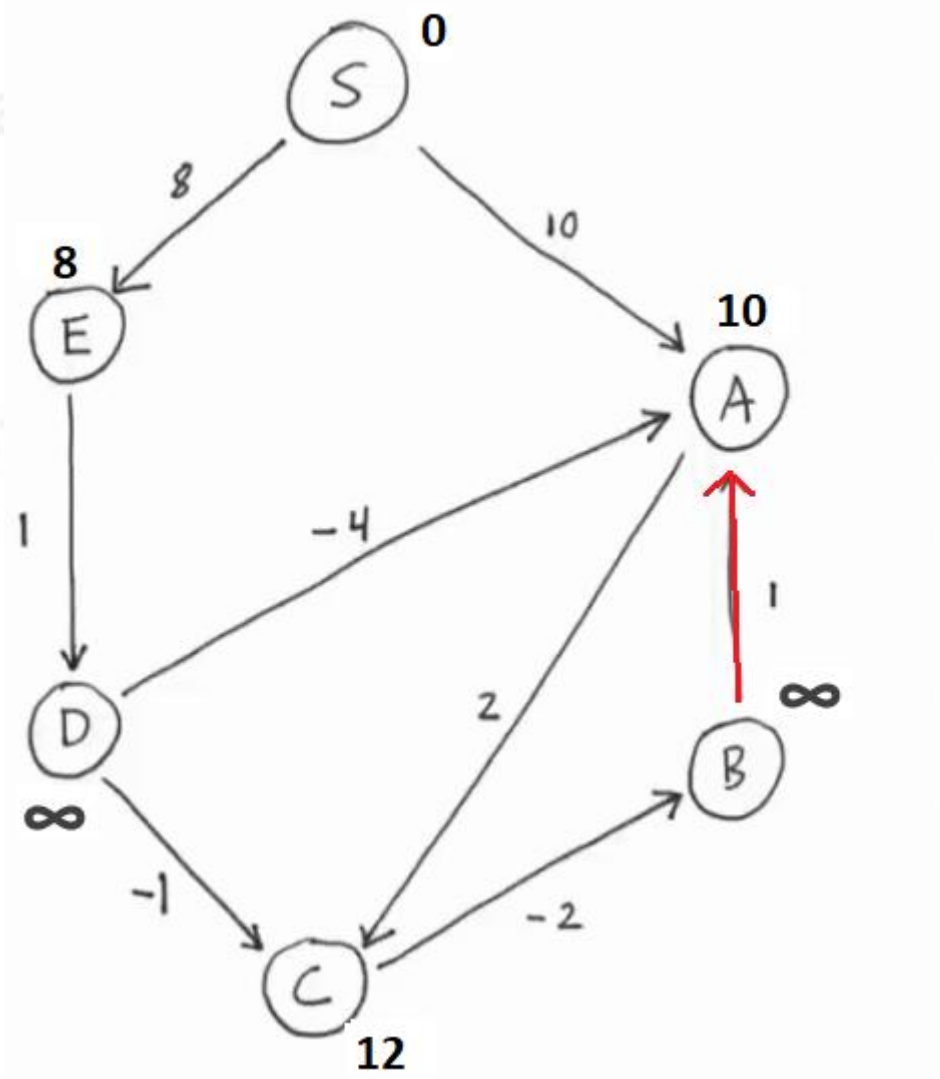
(D, A)

(D, C)

(E, D)

Bellman–Ford Algorithm
Example 1

1st iteration



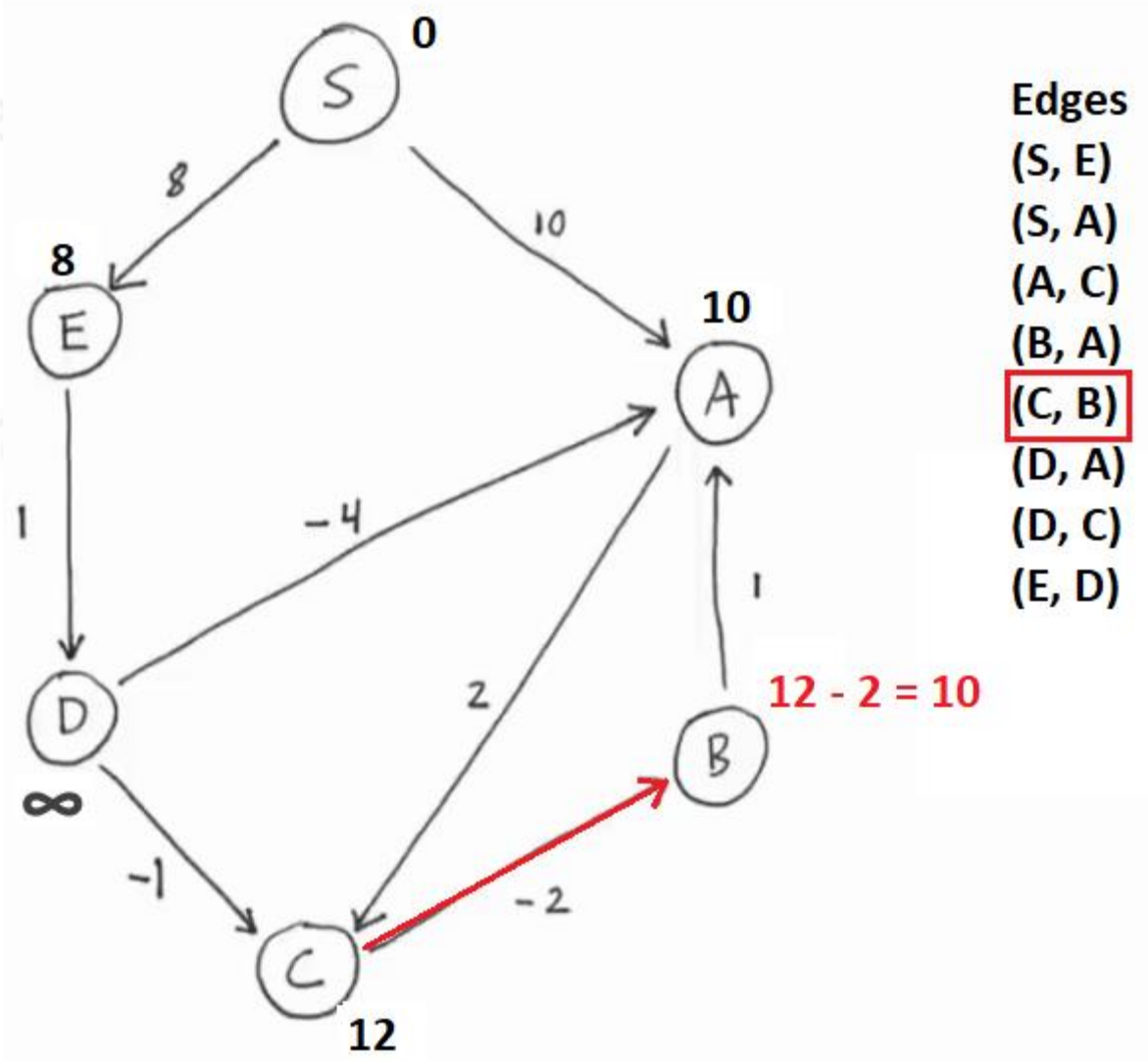
Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

1st iteration

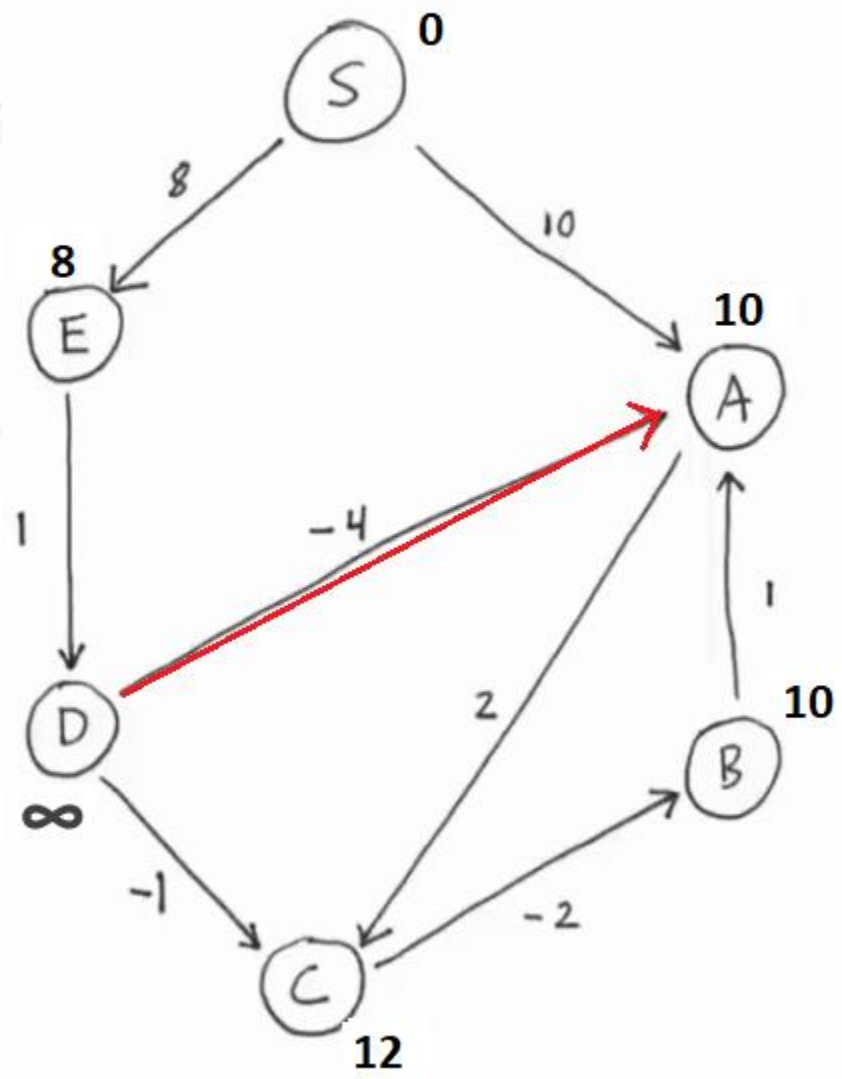


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

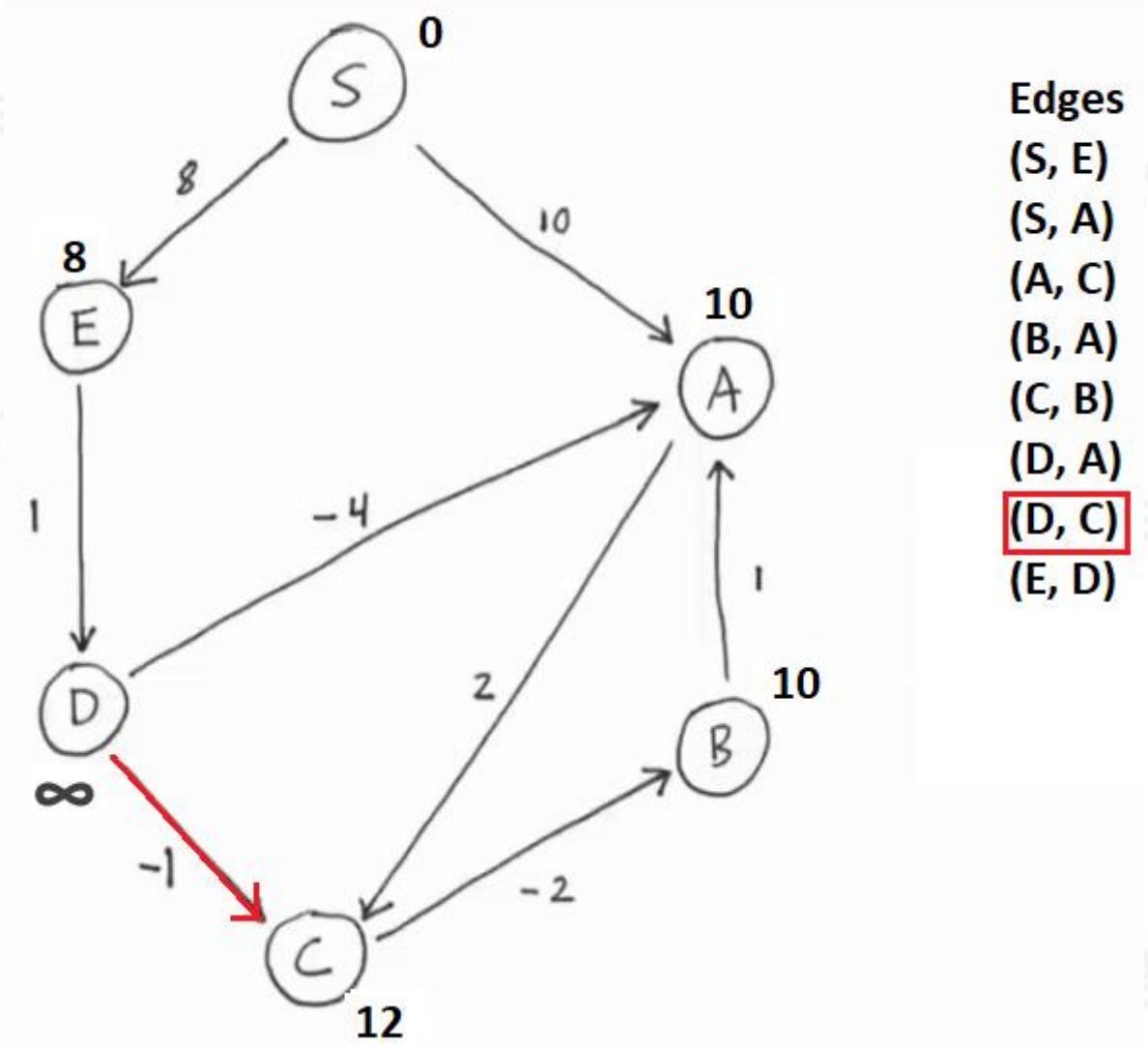
1st iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

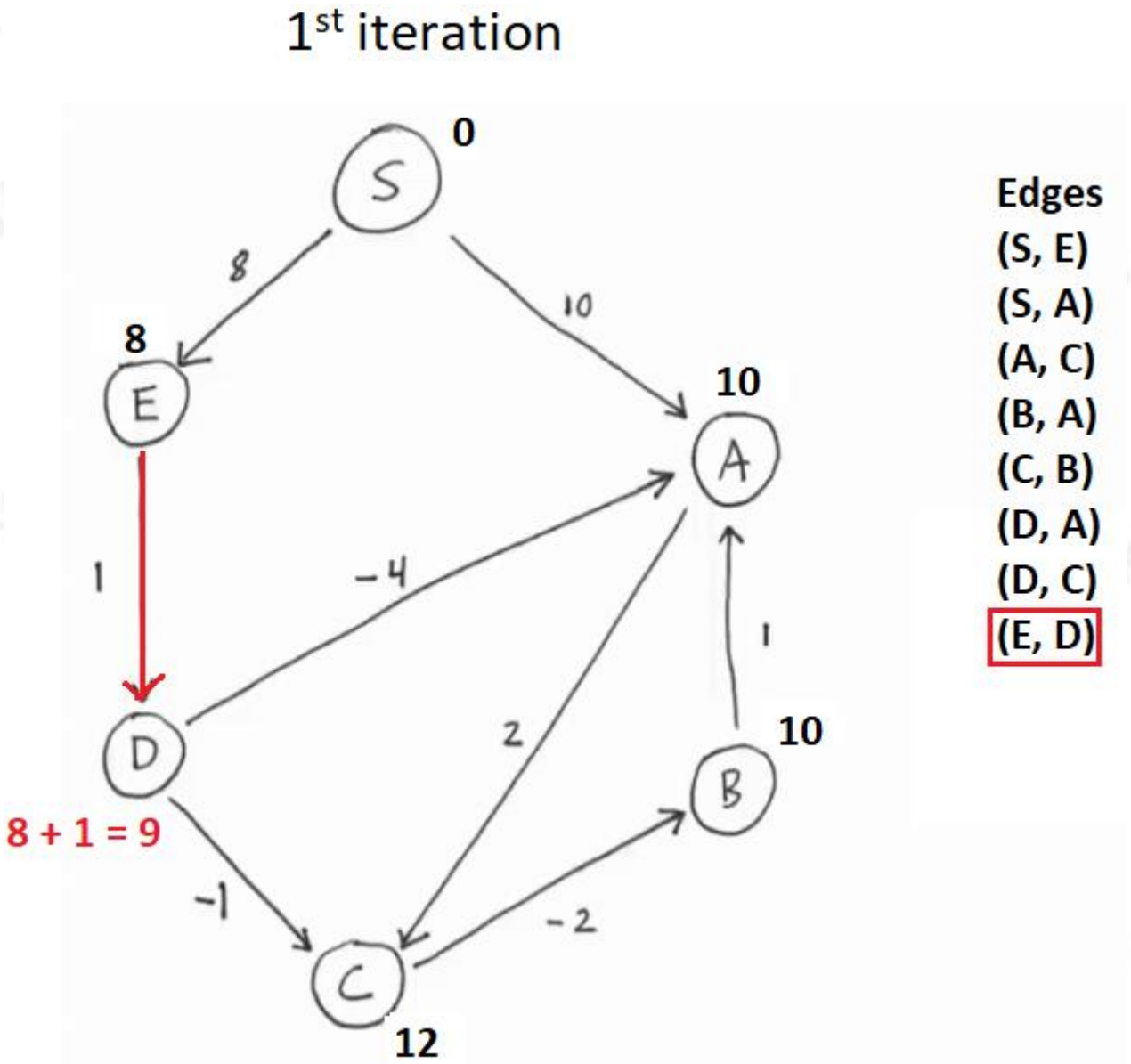
1st iteration



Edges

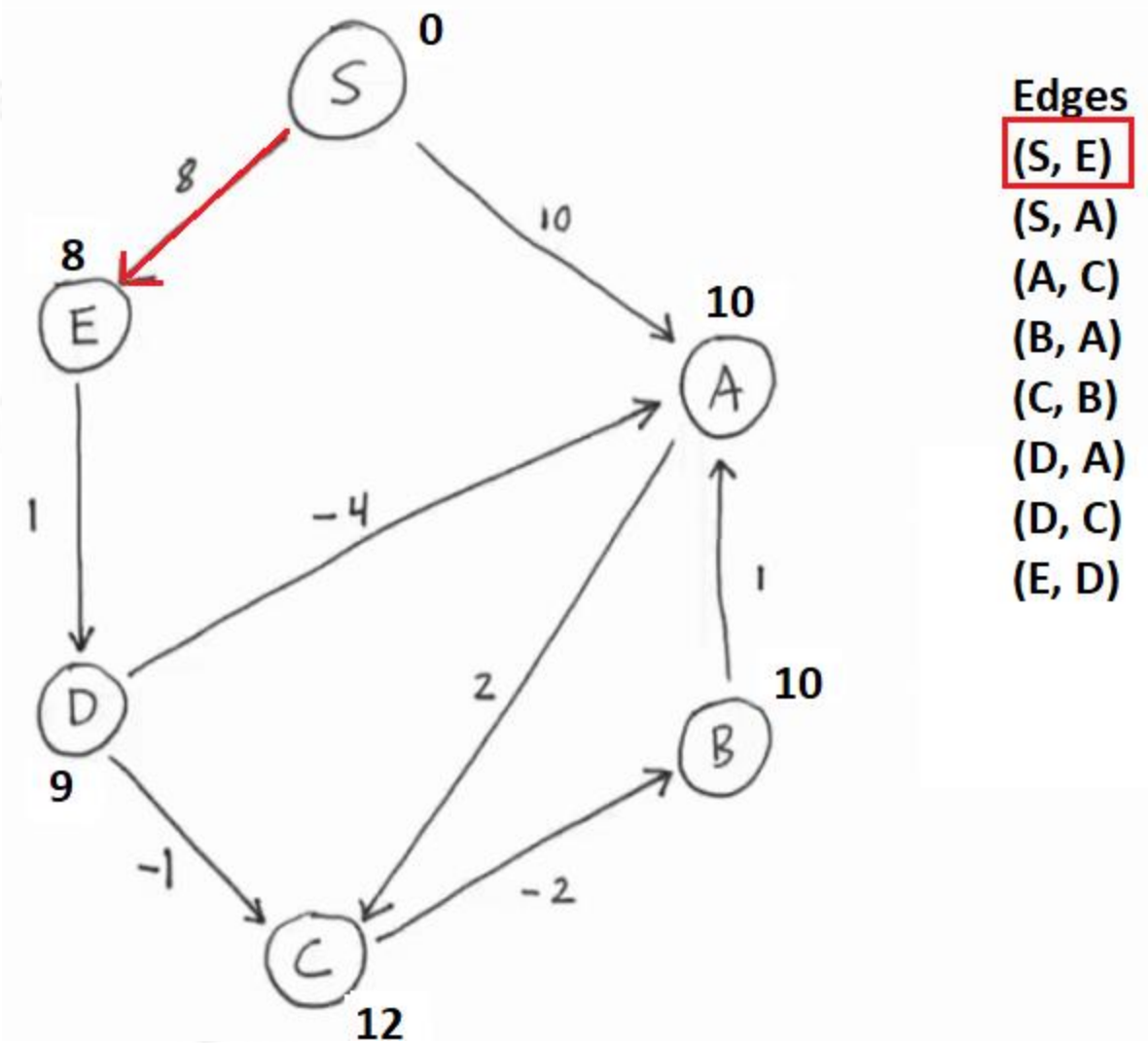
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1



Bellman–Ford Algorithm
Example 1

2nd iteration



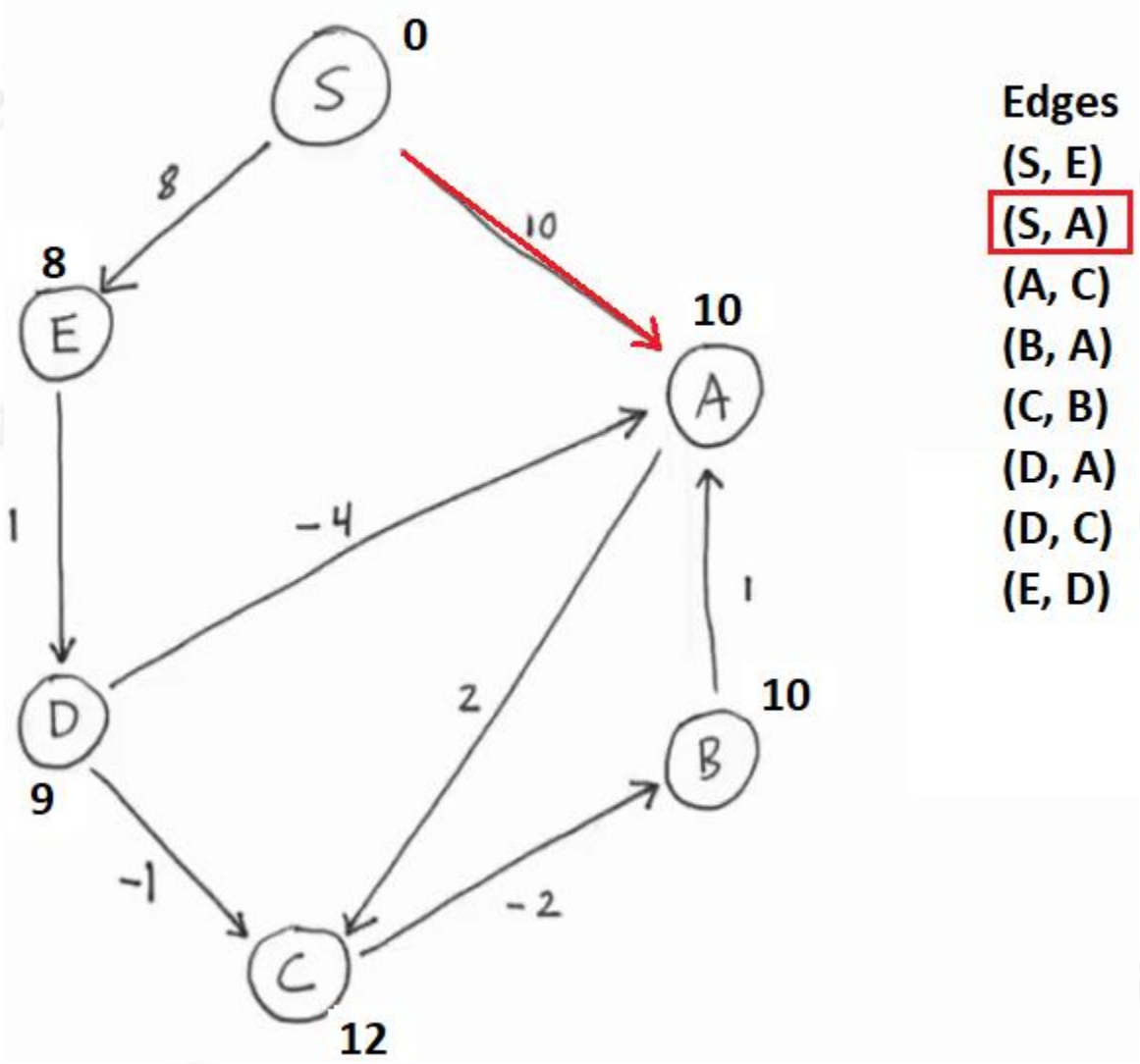
Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

2nd iteration

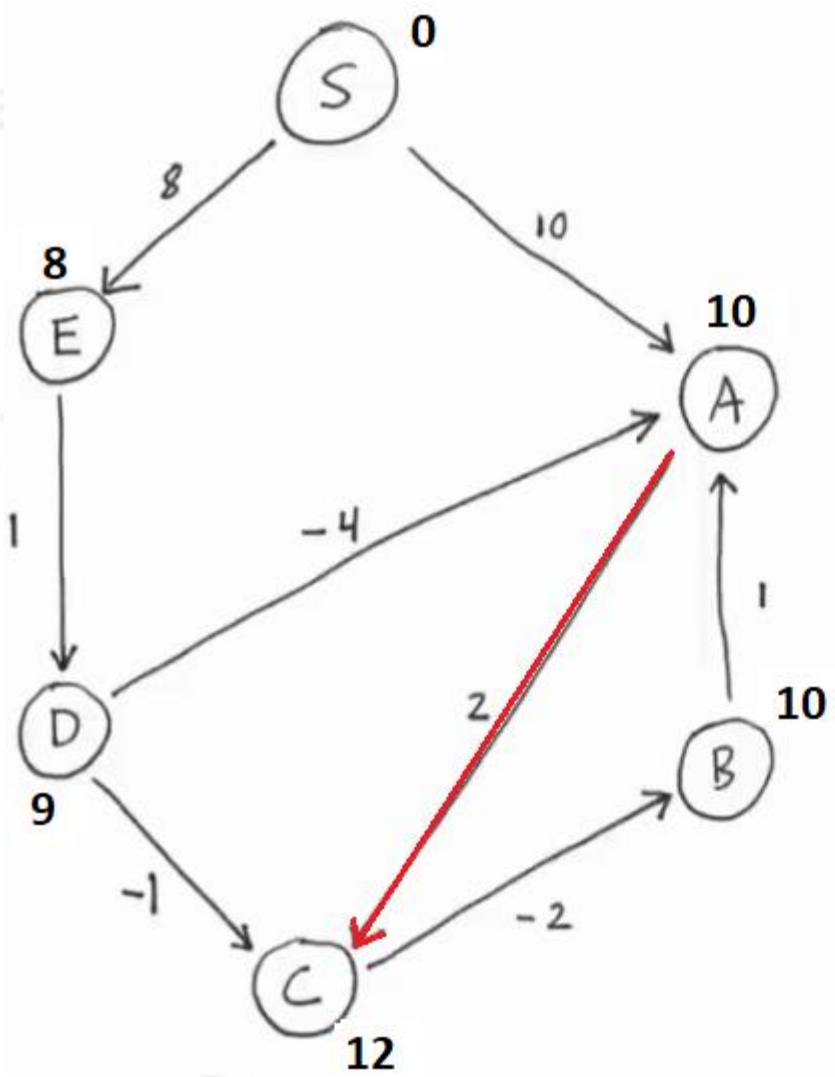


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

2nd iteration



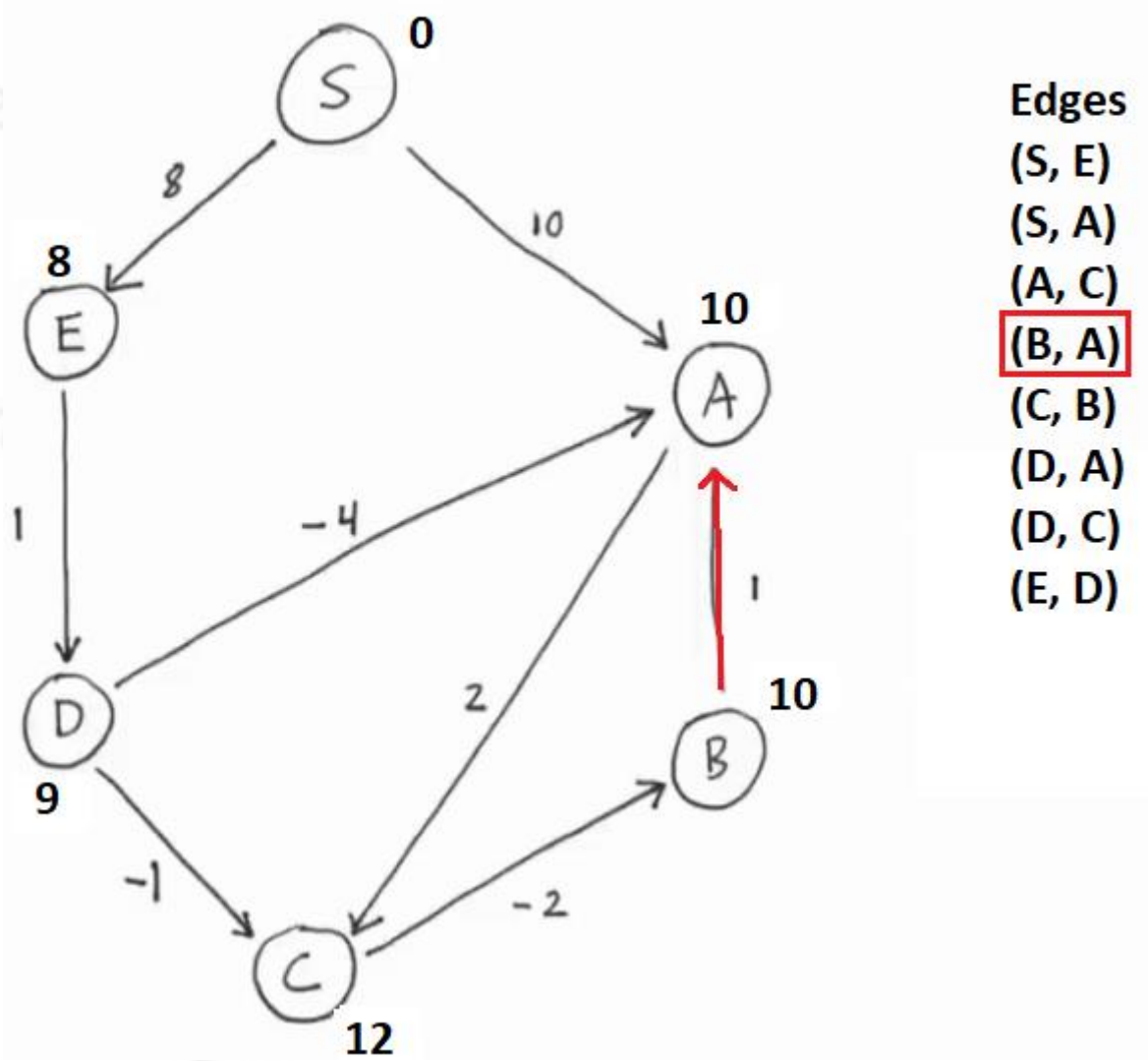
Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

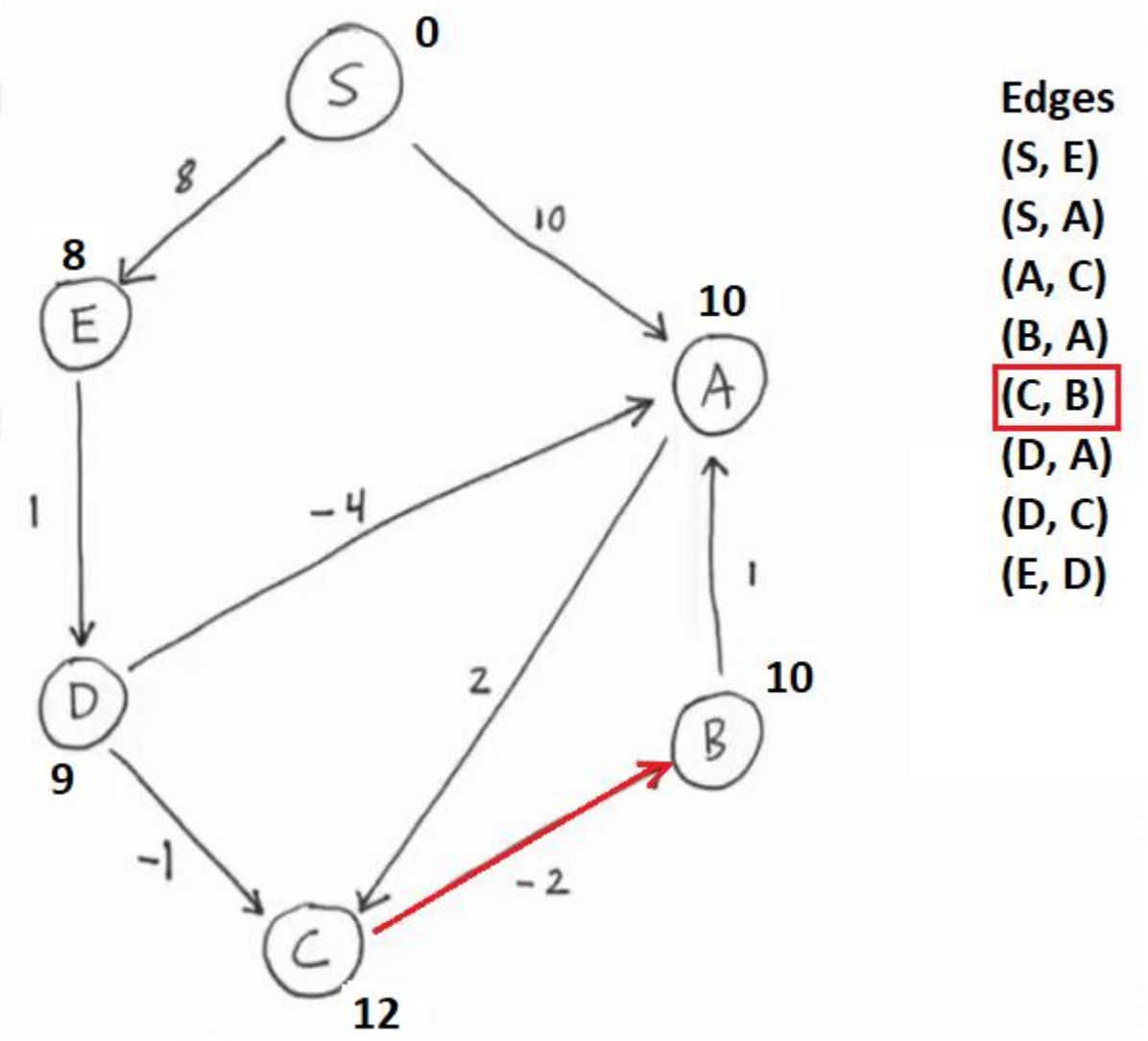
2nd iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

2nd iteration

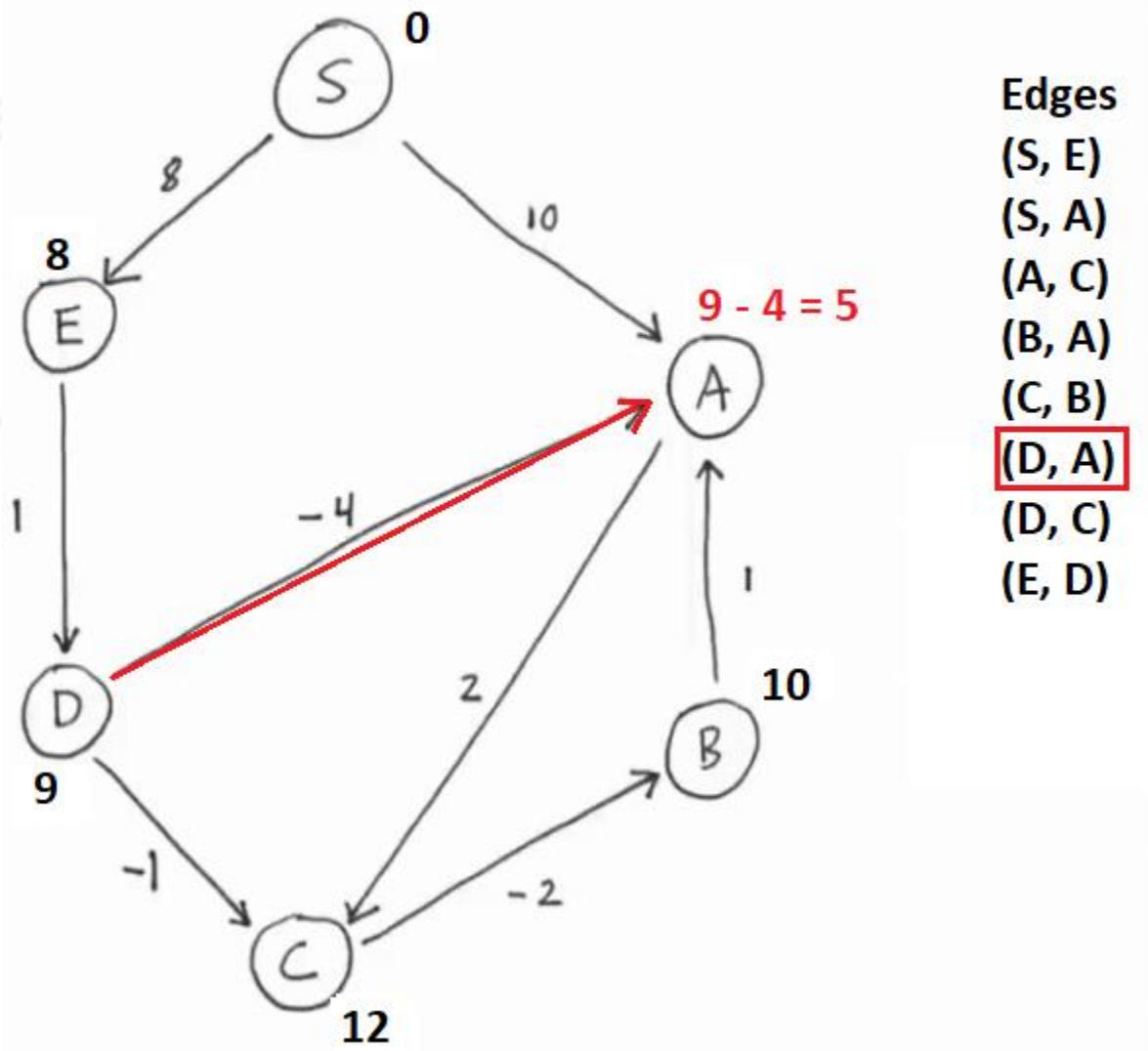


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

2nd iteration

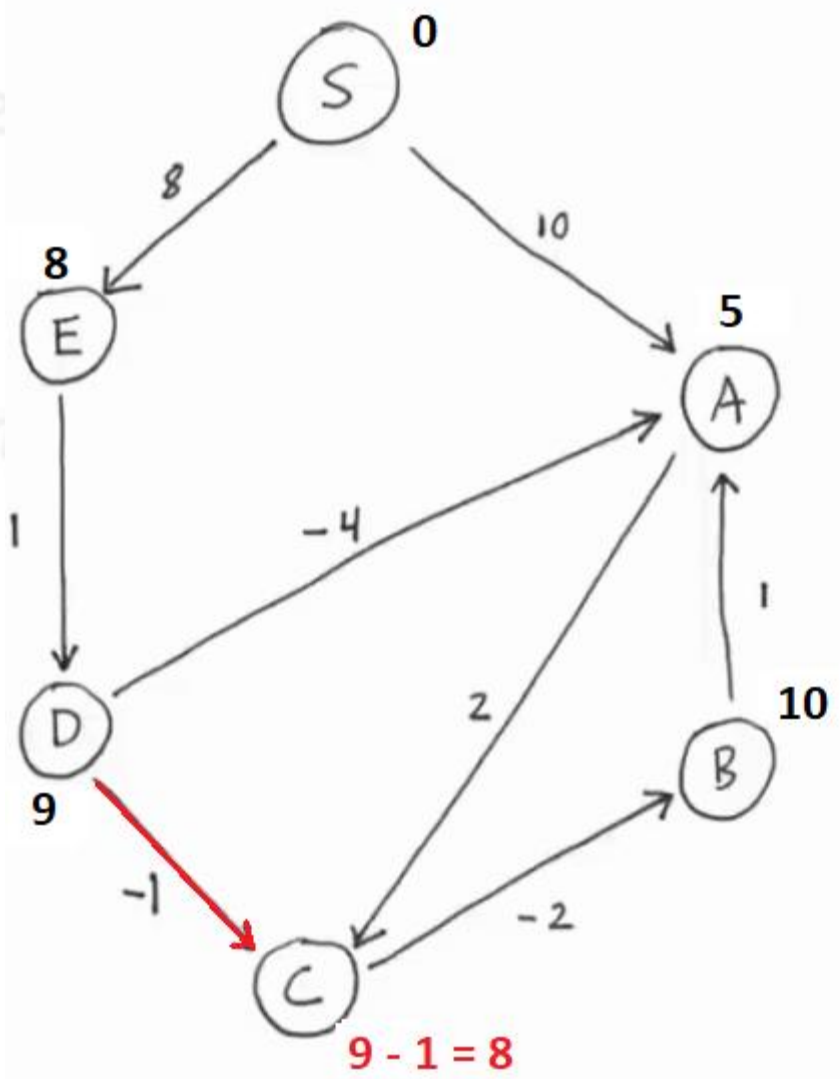


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

2nd iteration



Edges

(S, E)

(S, A)

(A, C)

(B, A)

(C, B)

(D, A)

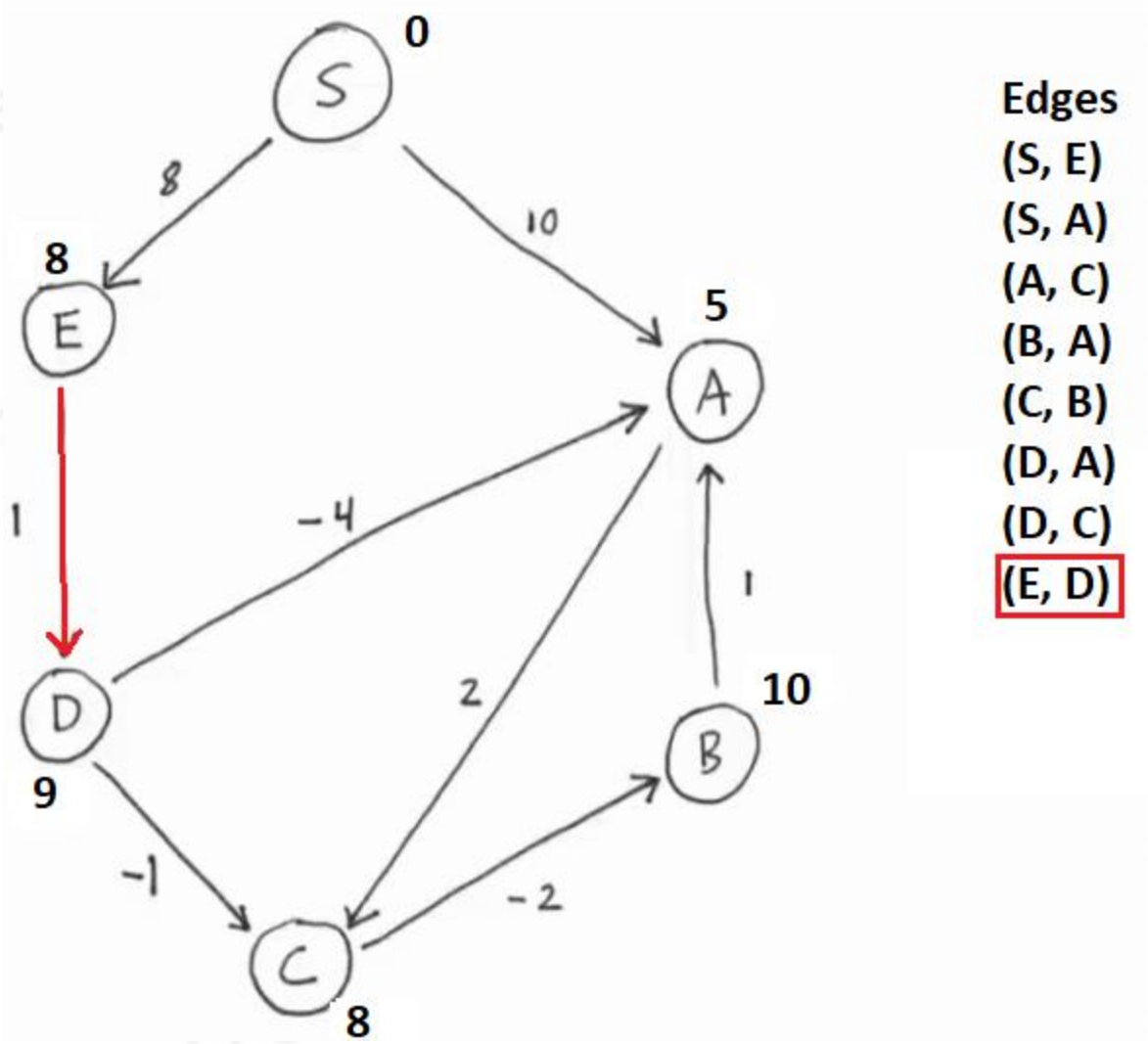
(D, C)

(E, D)

Bellman–Ford Algorithm

Example 1

2nd iteration



Edges

(S, E)

(S, A)

(A, C)

(B, A)

(C, B)

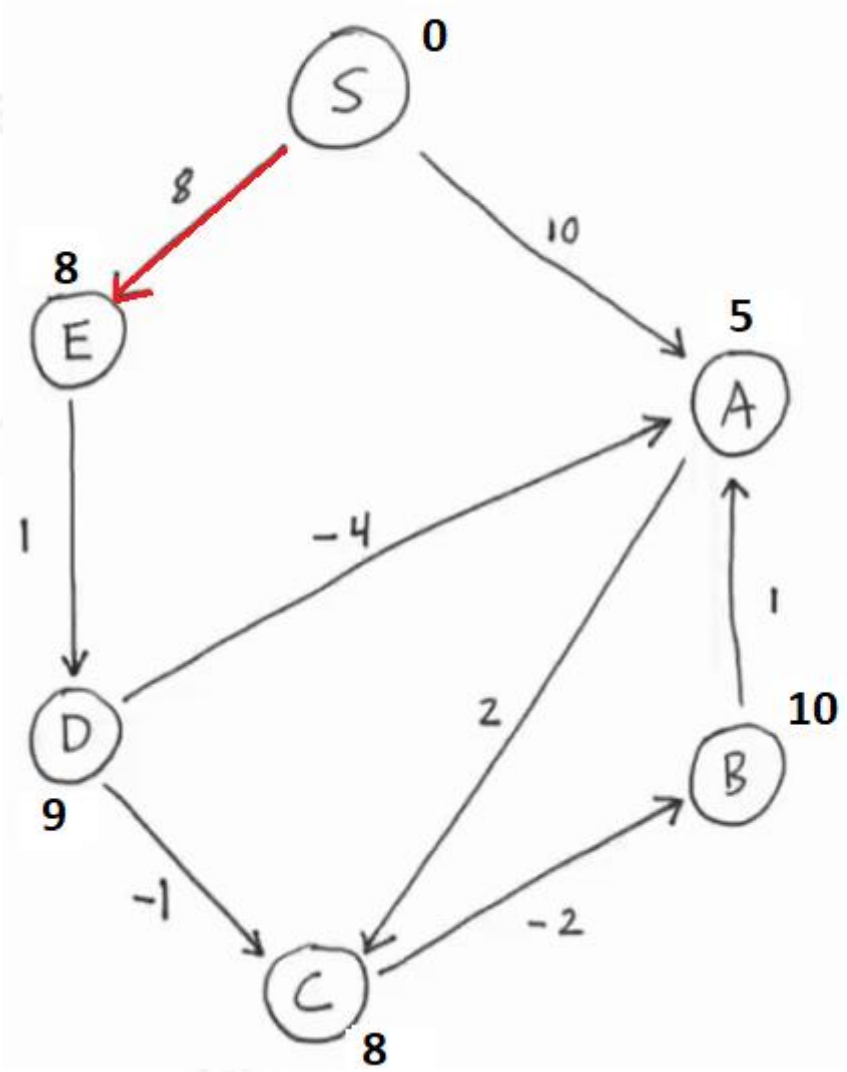
(D, A)

(D, C)

(E, D)

Bellman–Ford Algorithm
Example 1

3rd iteration

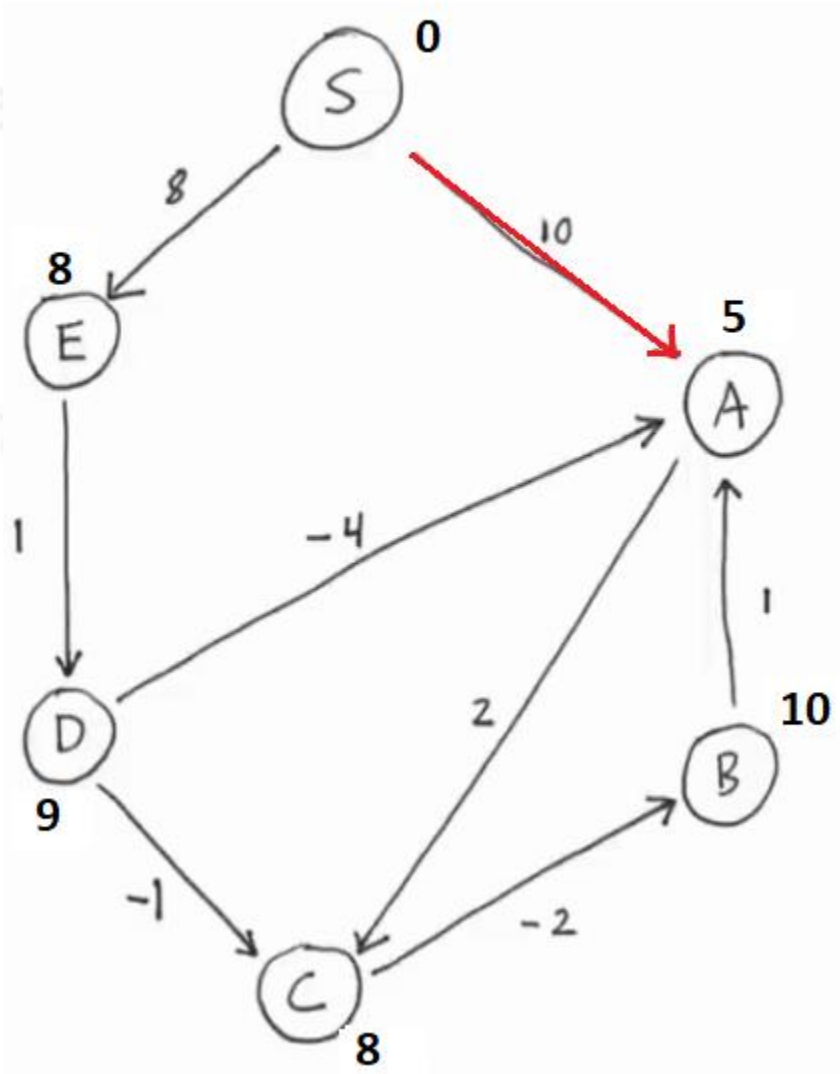


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

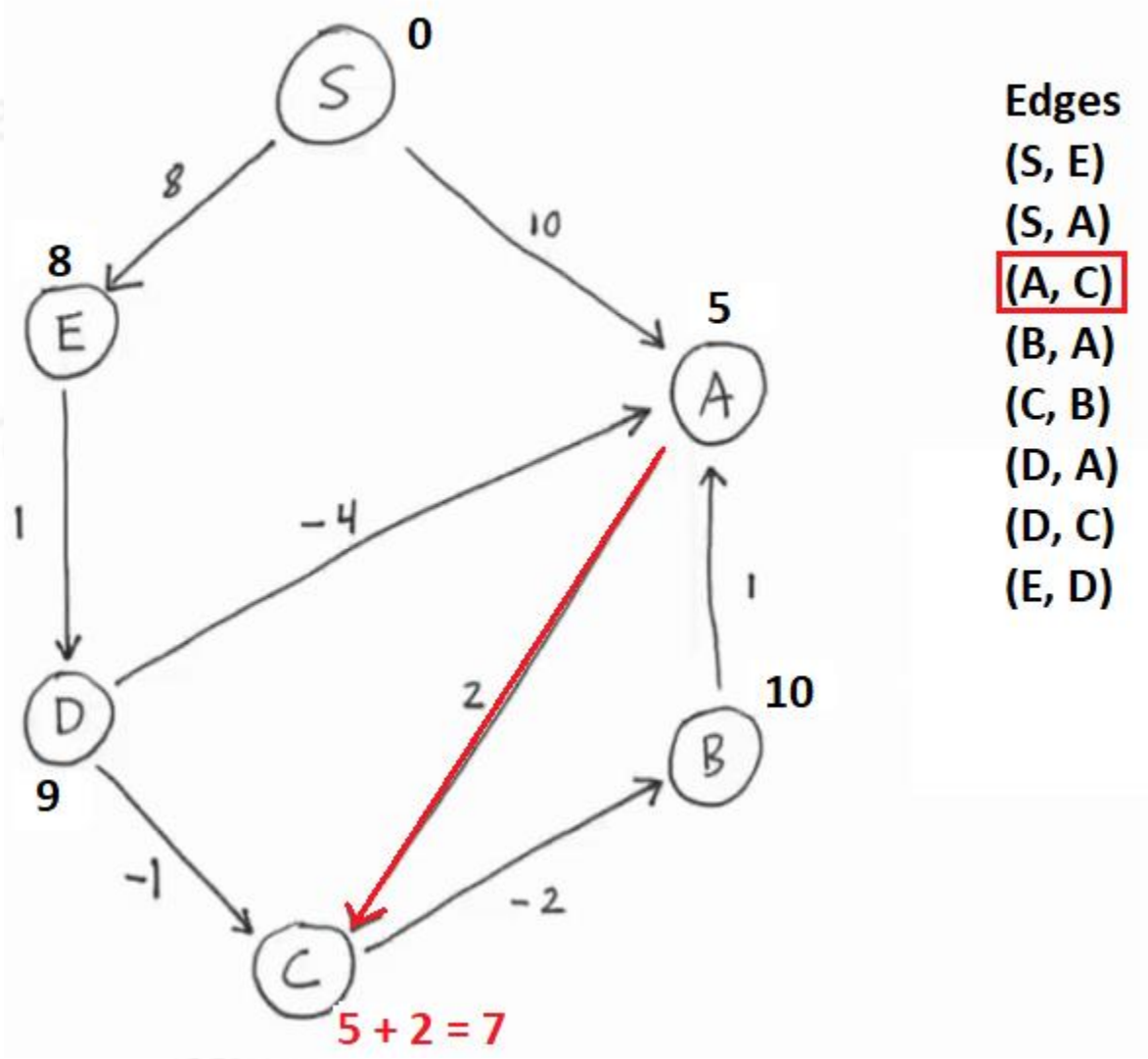
3rd iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

3rd iteration

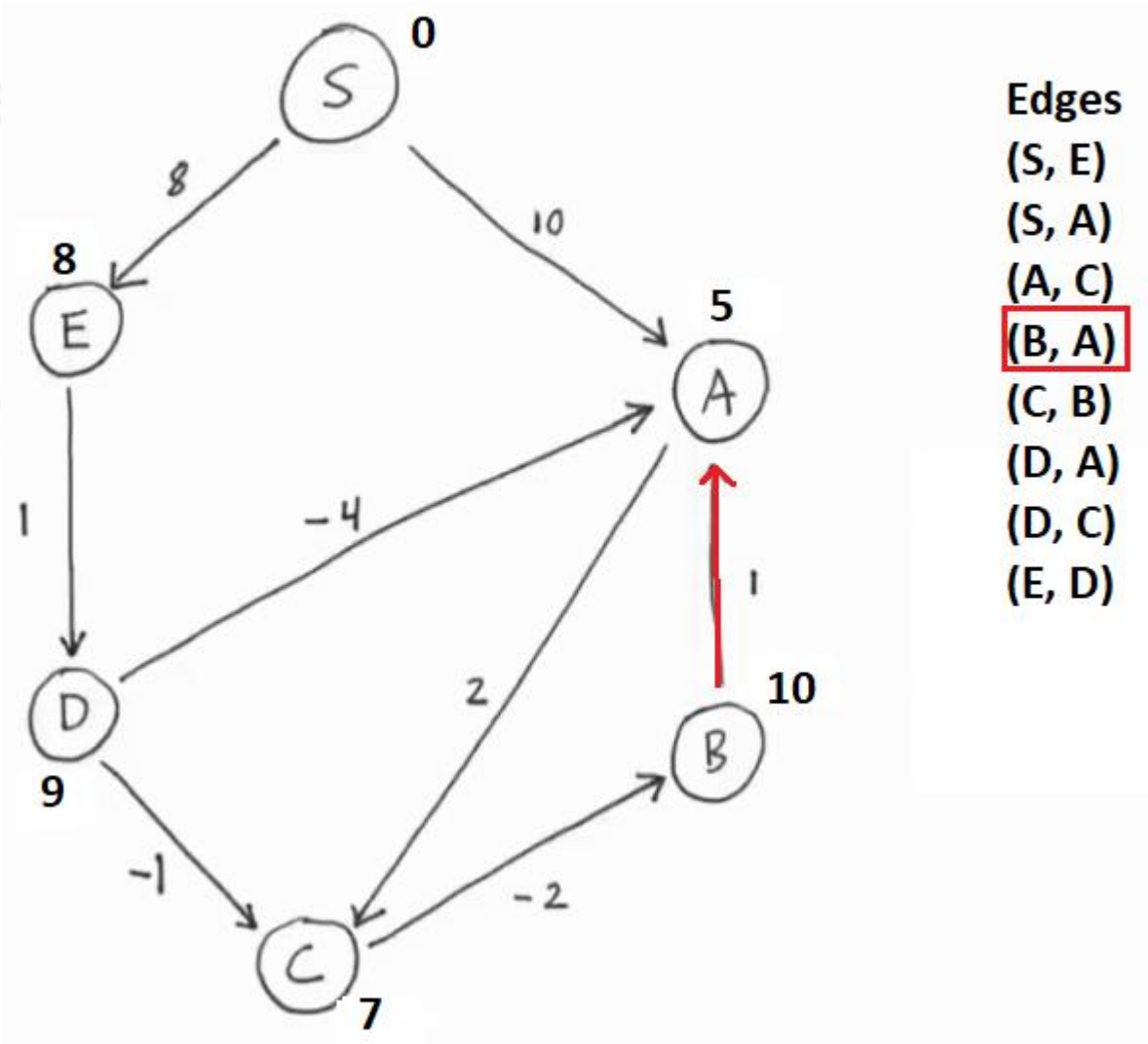


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

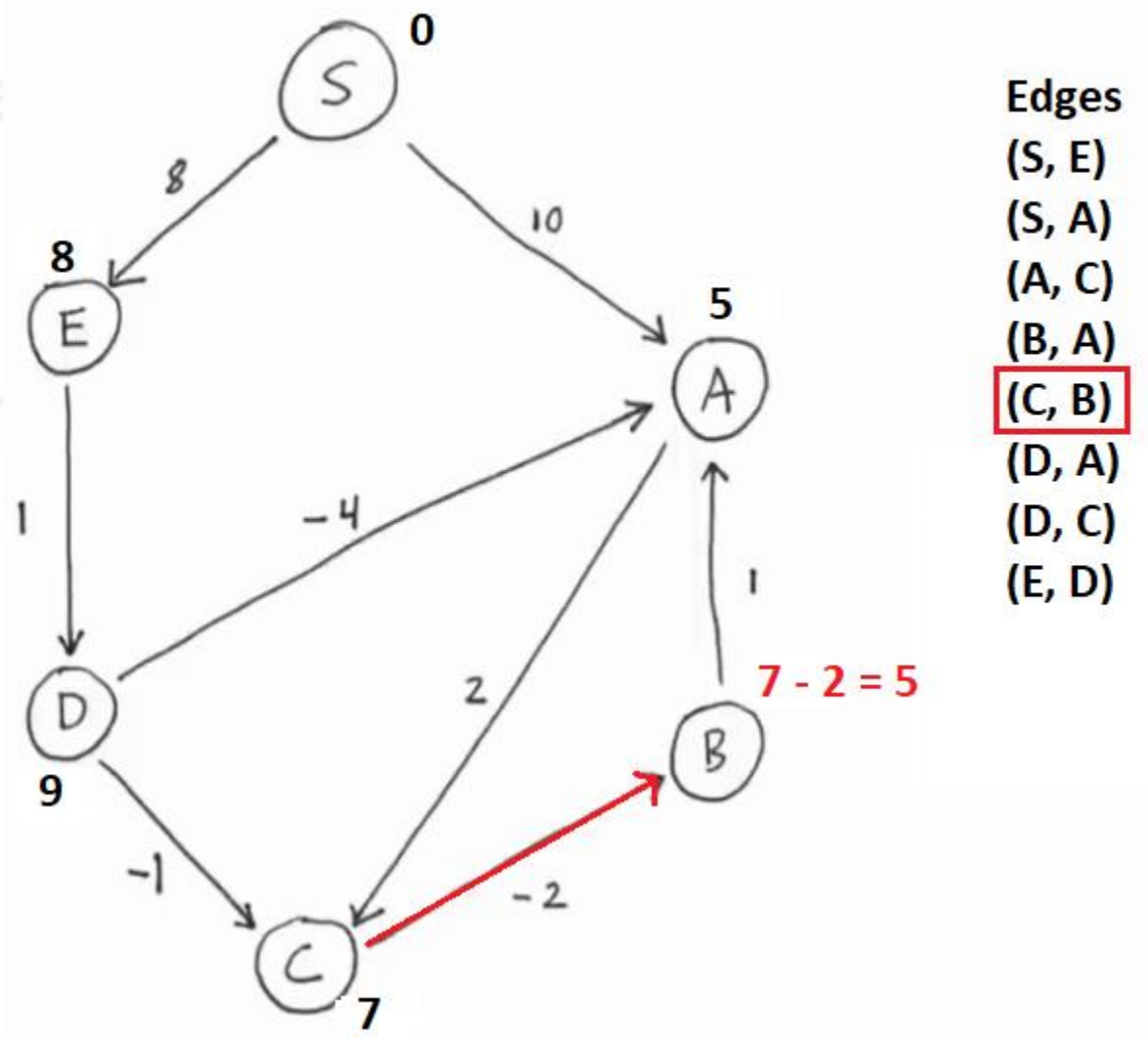
3rd iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

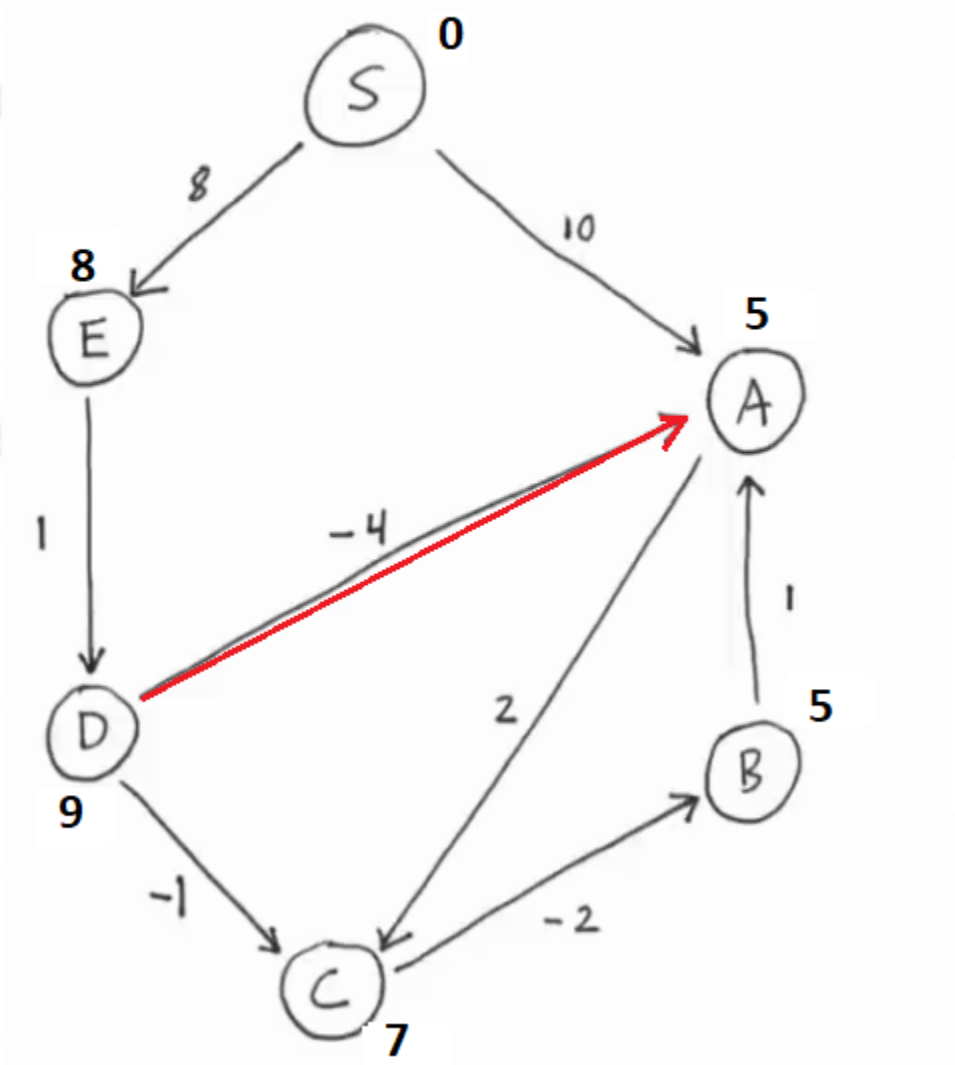
3rd iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

3rd iteration

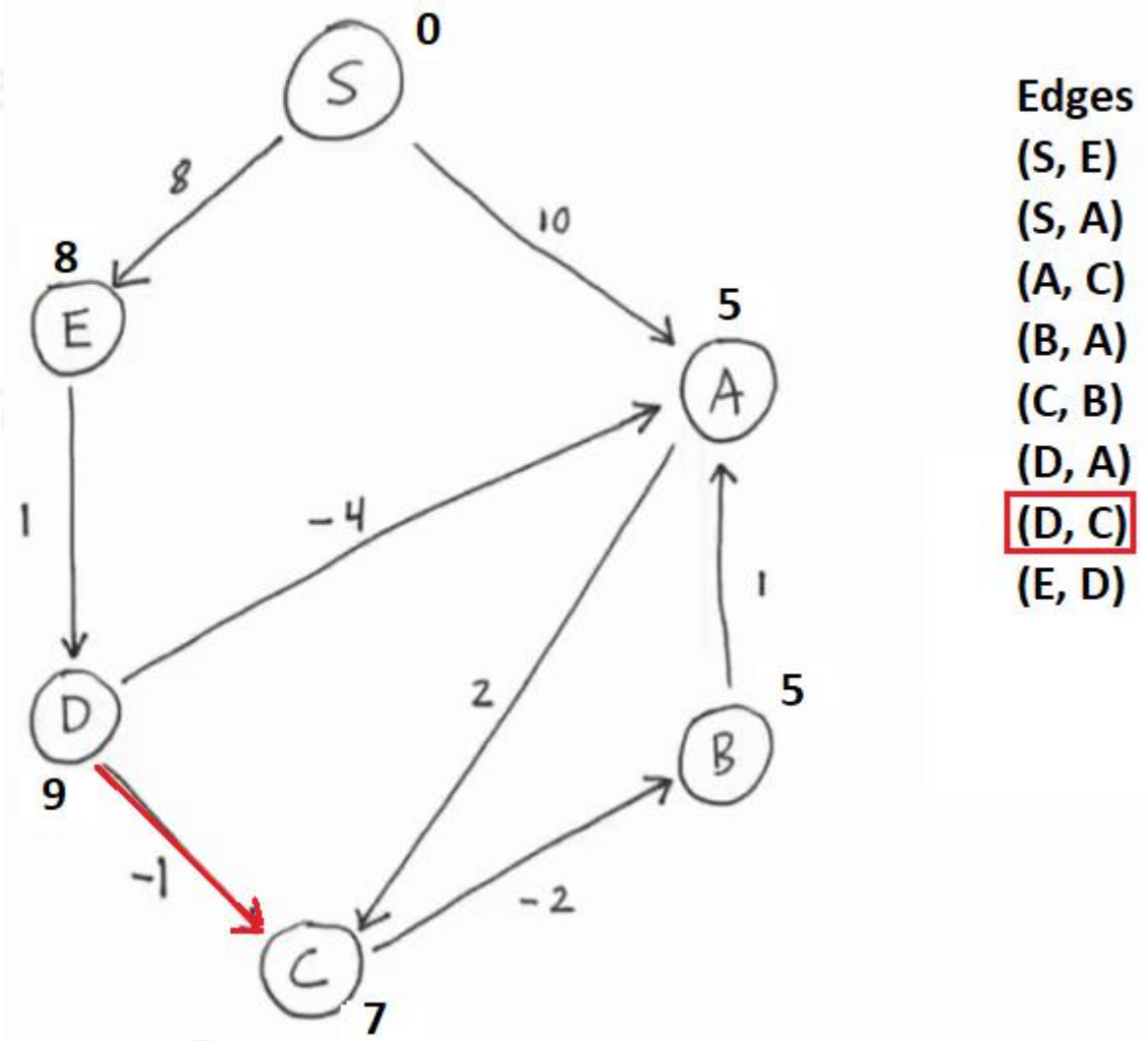


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

3rd iteration

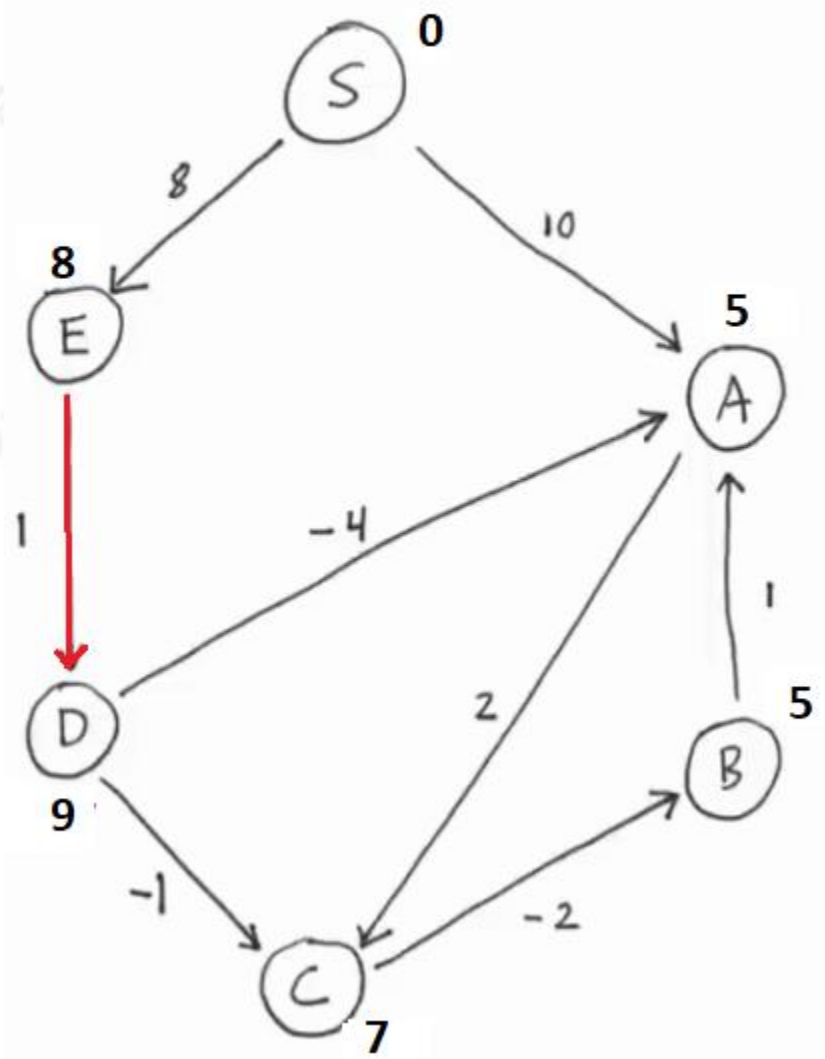


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

3rd iteration

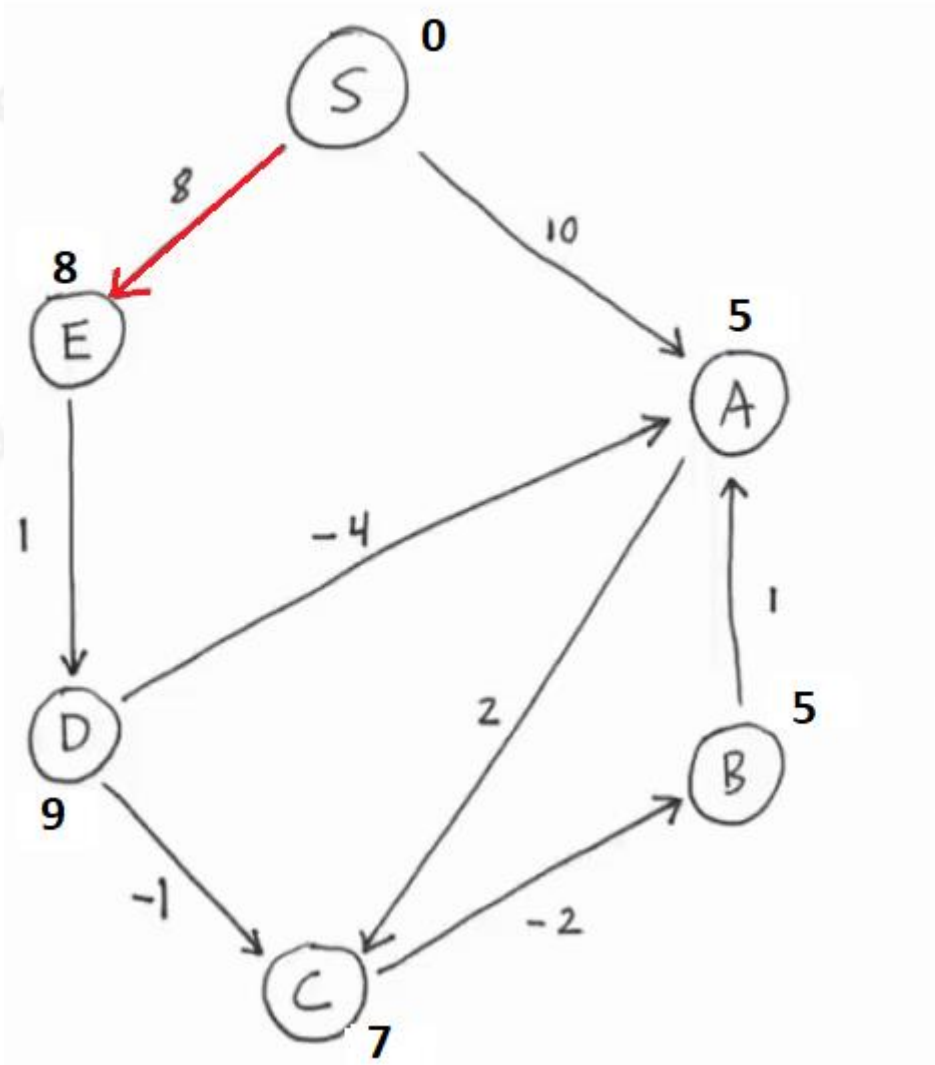


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

4th iteration

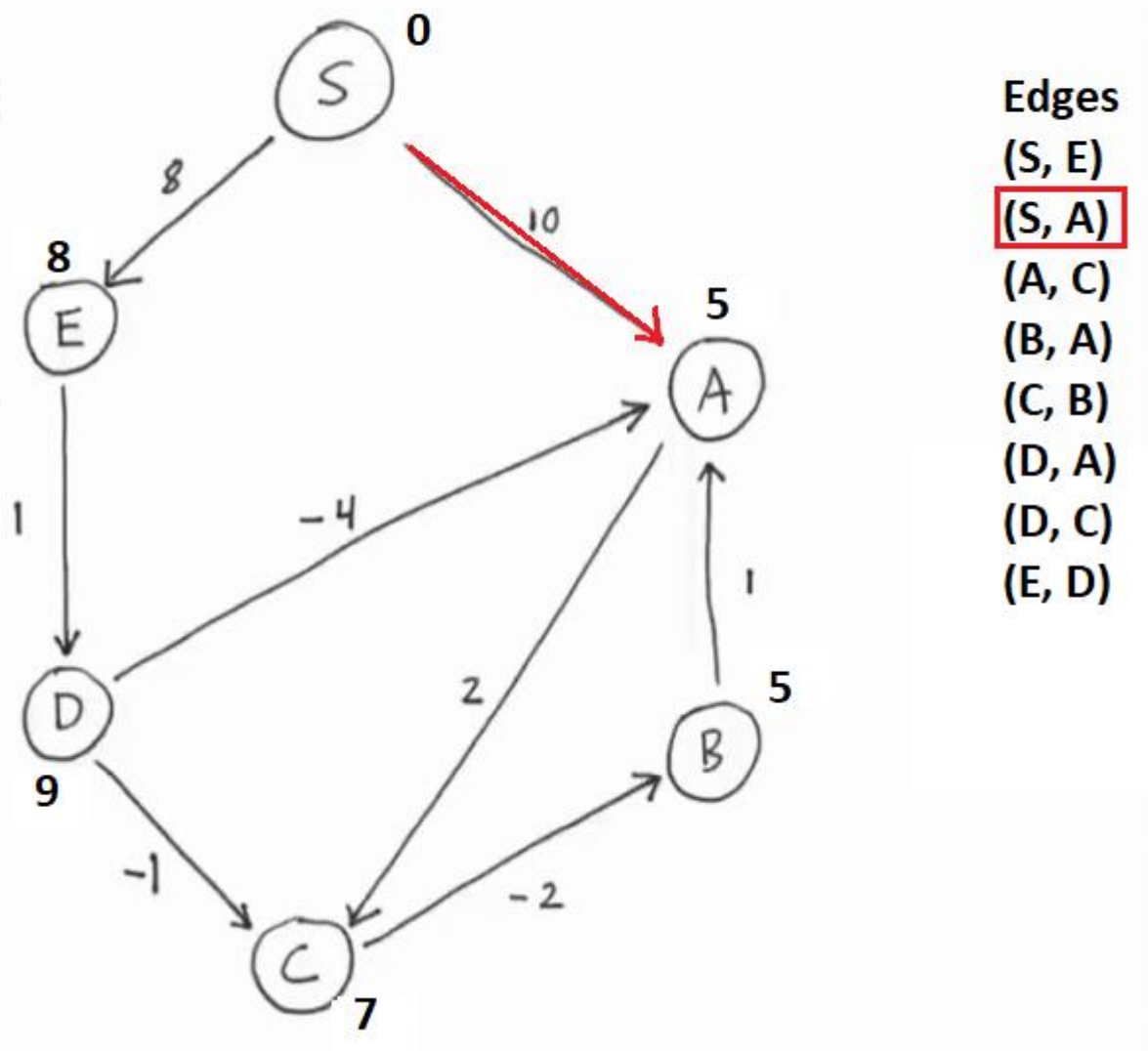


Edges

- **(S, E)**
- **(S, A)**
- **(A, C)**
- **(B, A)**
- **(C, B)**
- **(D, A)**
- **(D, C)**
- **(E, D)**

Bellman–Ford Algorithm
Example 1

4th iteration

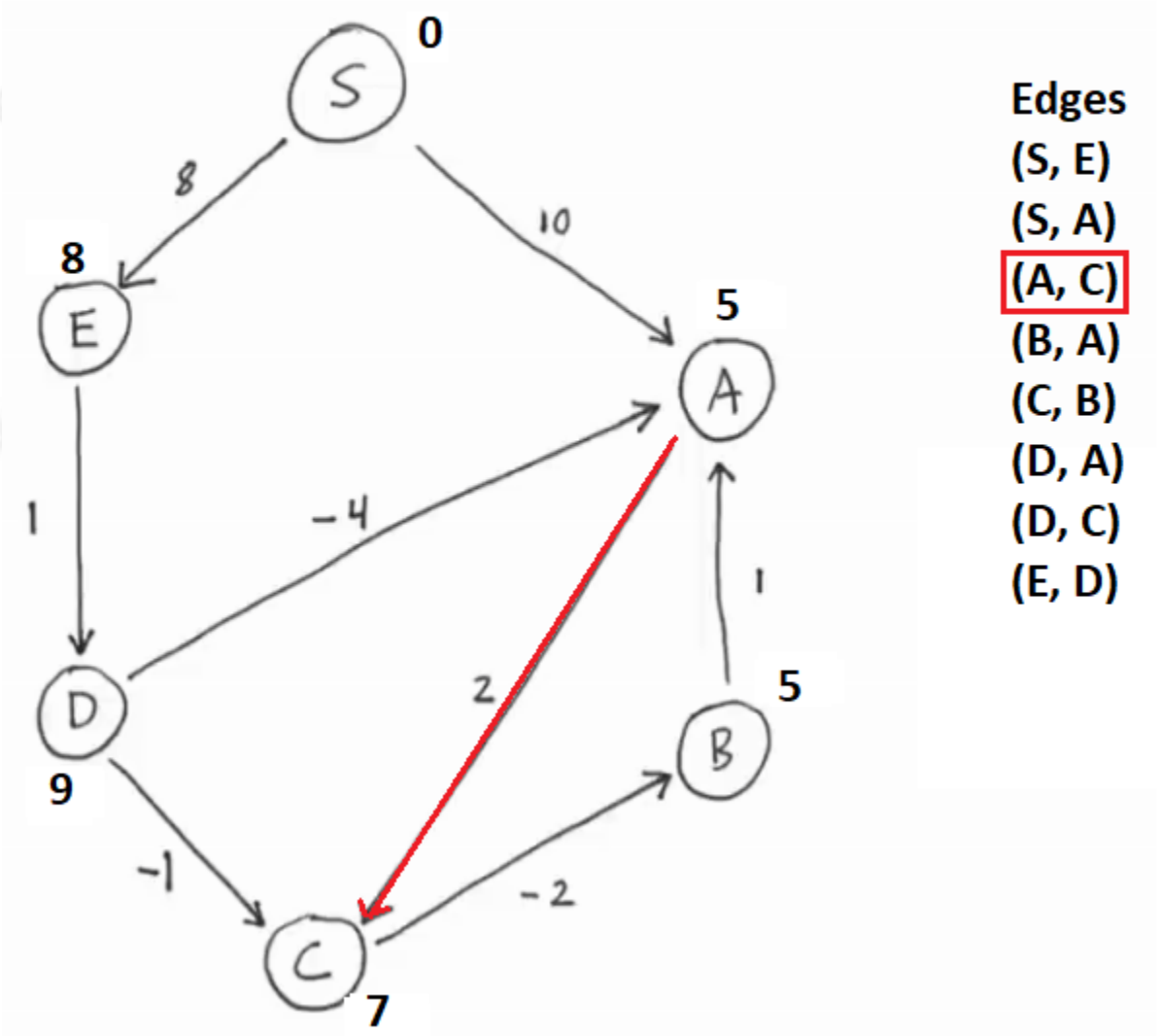


Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm

Example 1

4th iteration

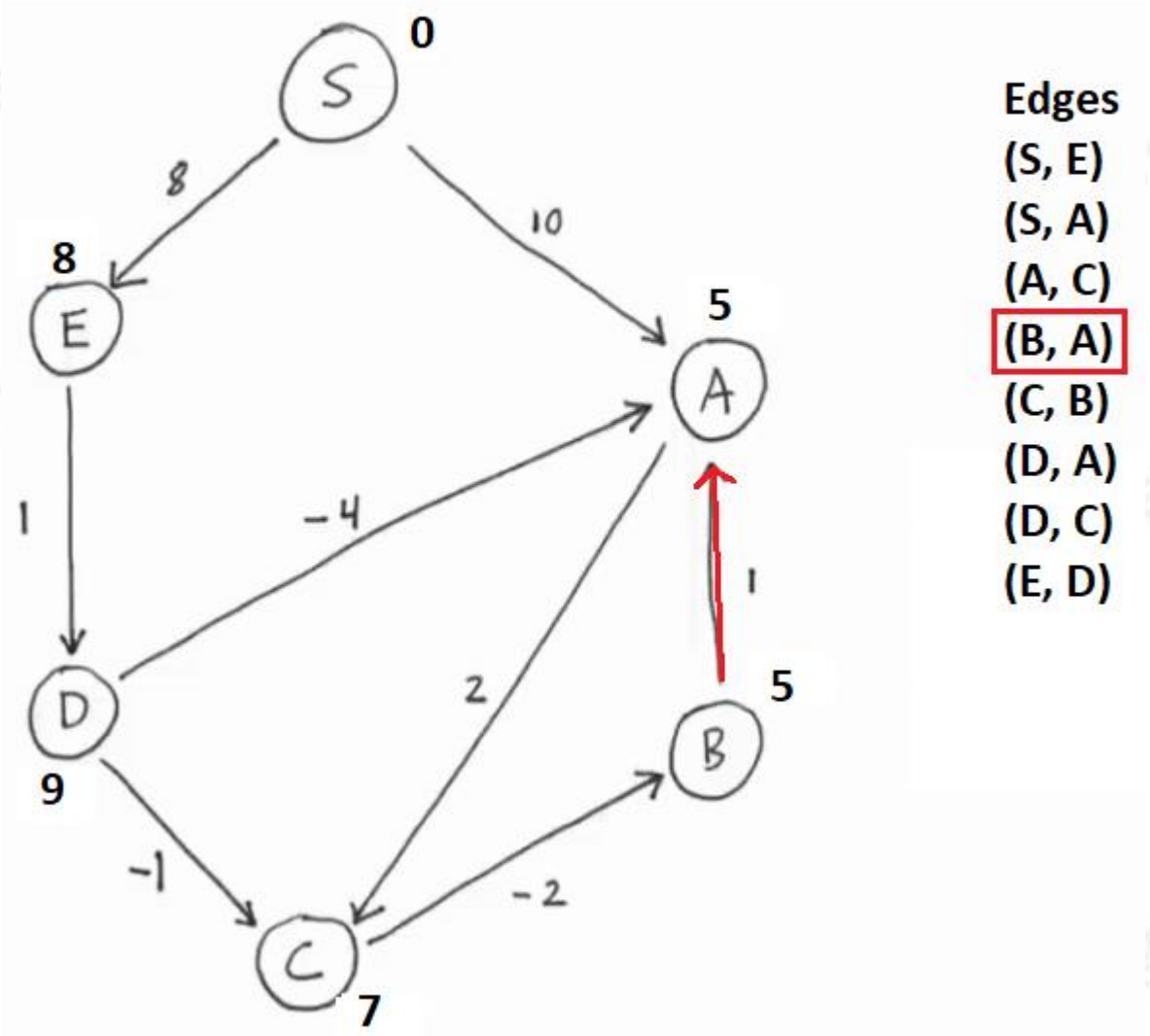


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

4th iteration

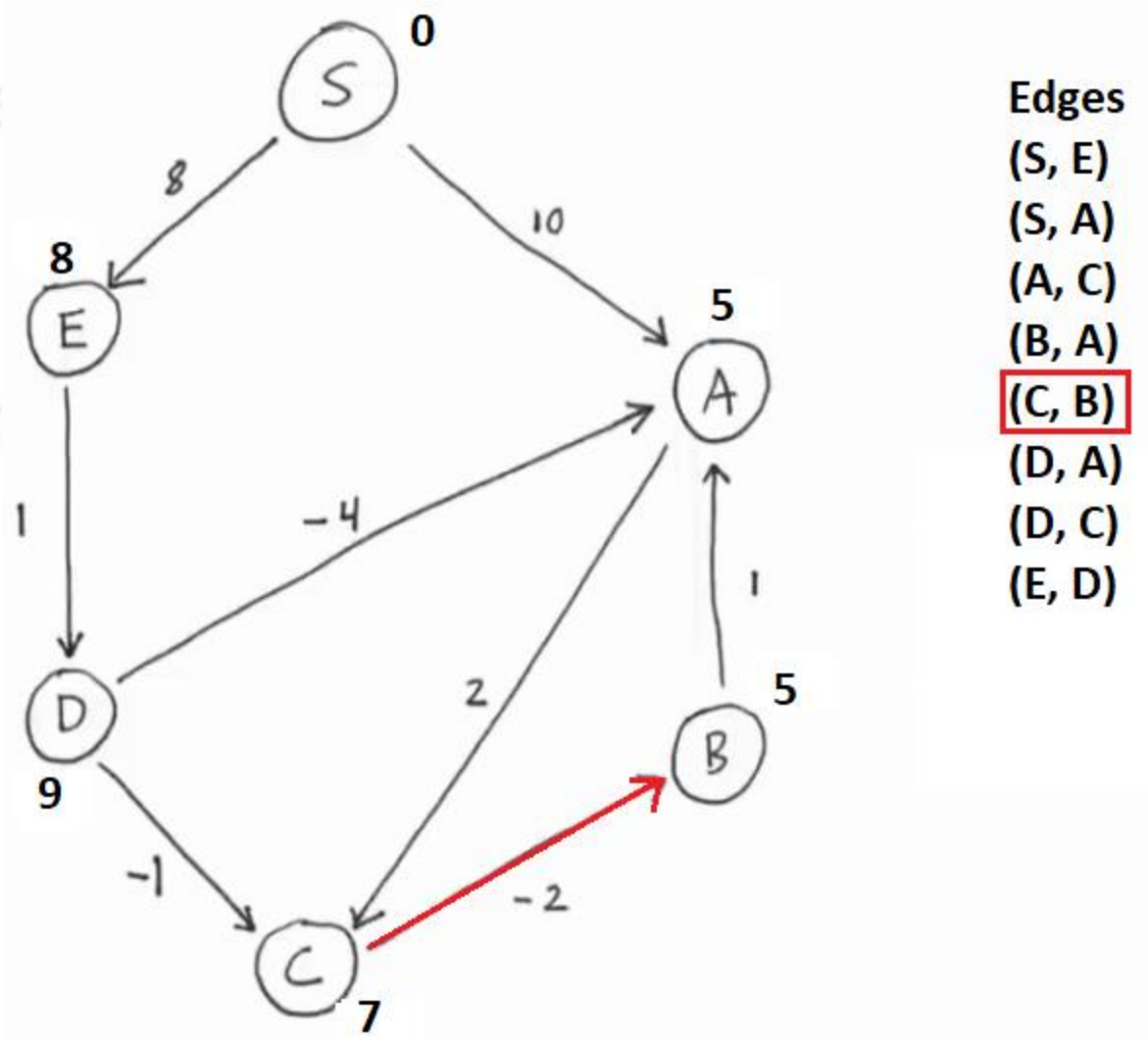


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

4th iteration

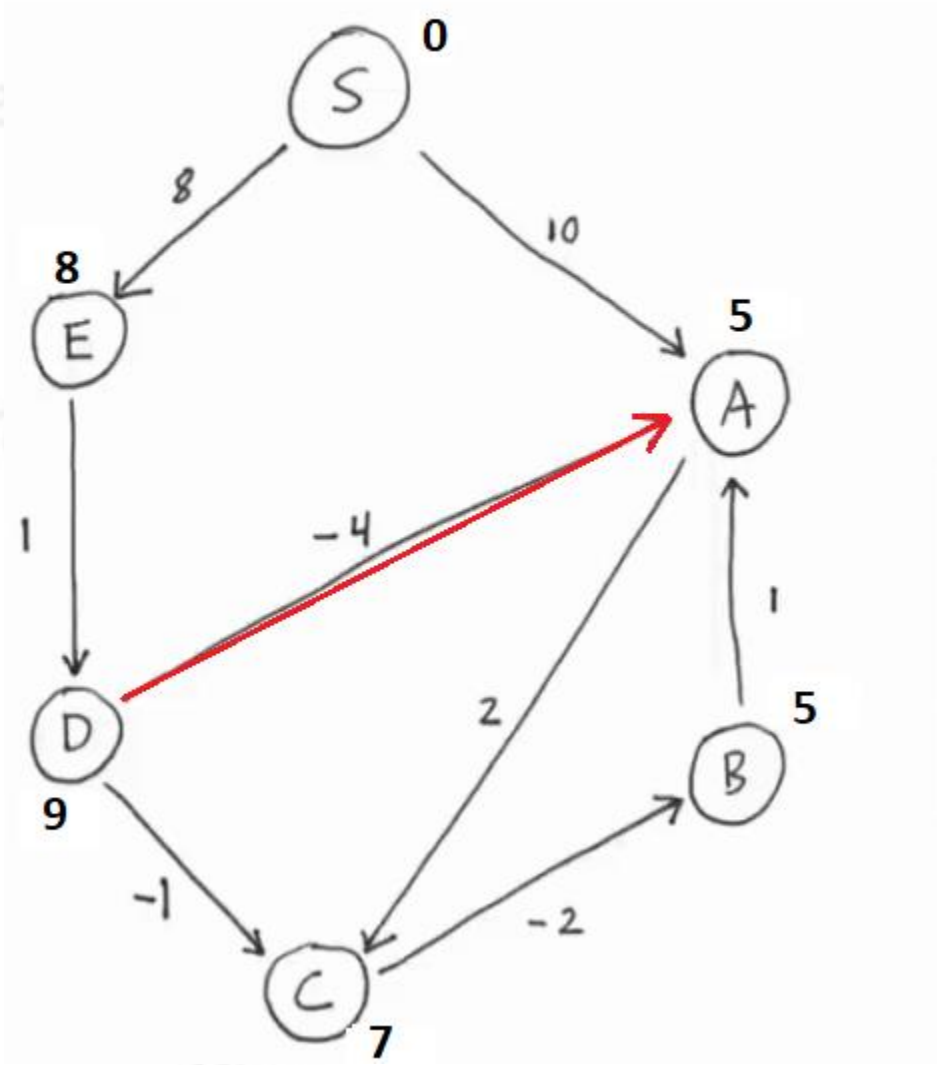


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

4th iteration

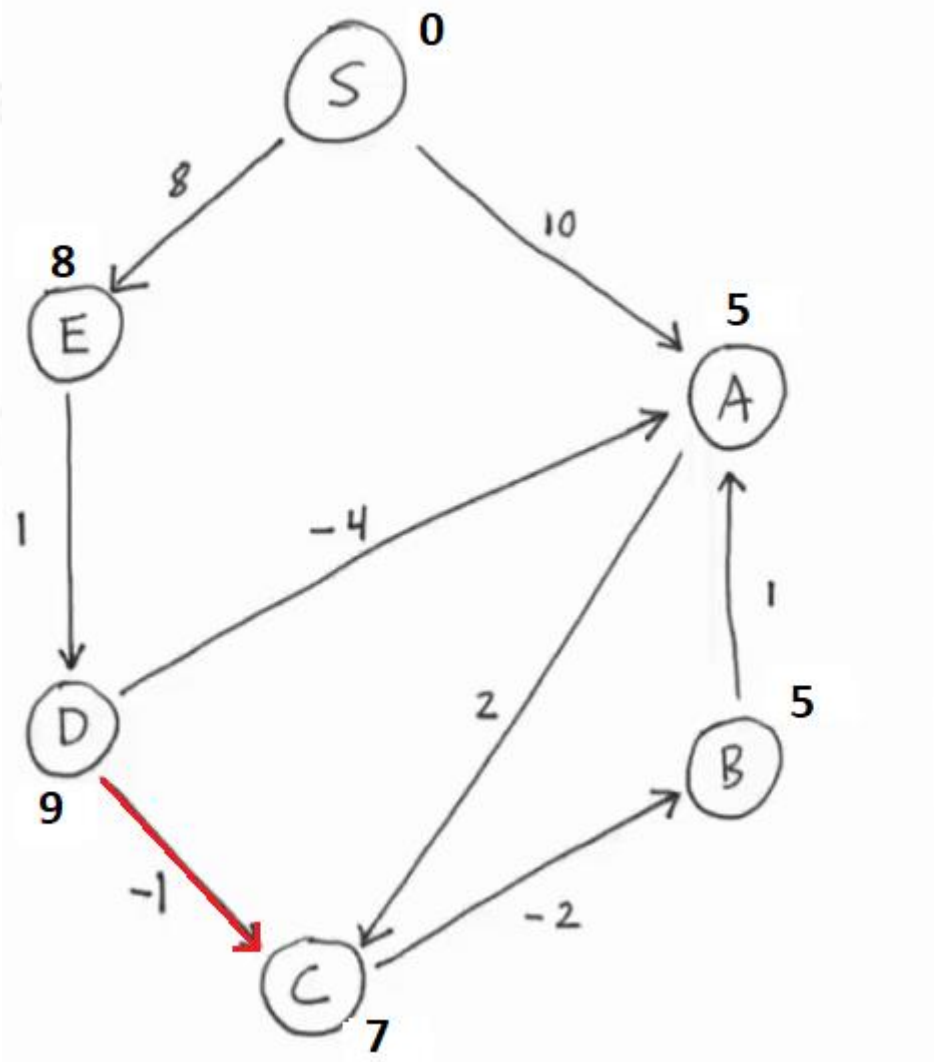


Edges

(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

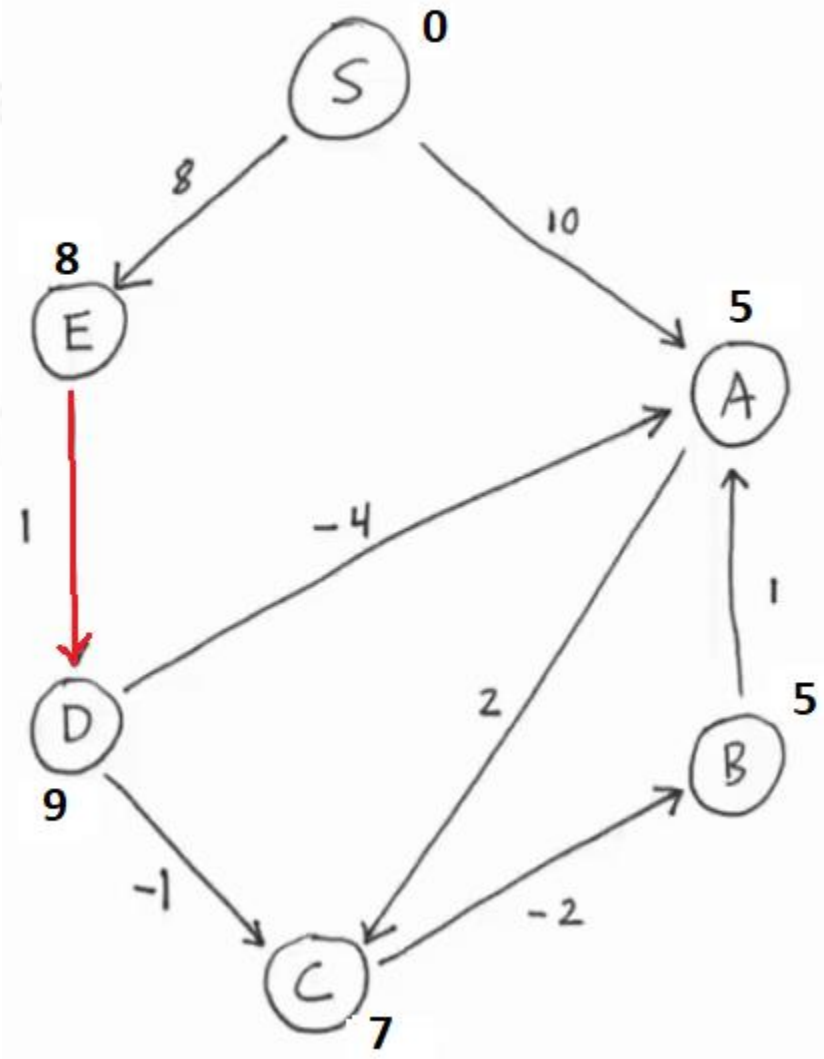
4th iteration



Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

4th iteration



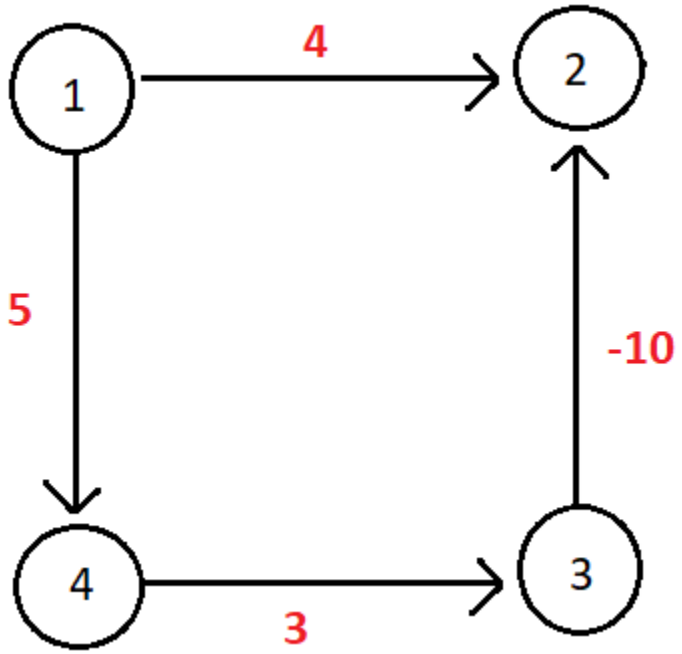
Edges
(S, E)
(S, A)
(A, C)
(B, A)
(C, B)
(D, A)
(D, C)
(E, D)

Bellman–Ford Algorithm
Example 1

- We finished iteration four and all the values are the same! Why?
- The algorithm takes at most (|V| - 1) iterations, we can make it more efficient by stopping earlier if nothing improves during an iteration.
- So, in this example, after the fourth iteration we are finished.
- Don’t be worried if the values after each iteration are different, it depends on the order in which the edges are visited.
- You may need five iterations; the algorithm guarantees that you will end up with the same result.

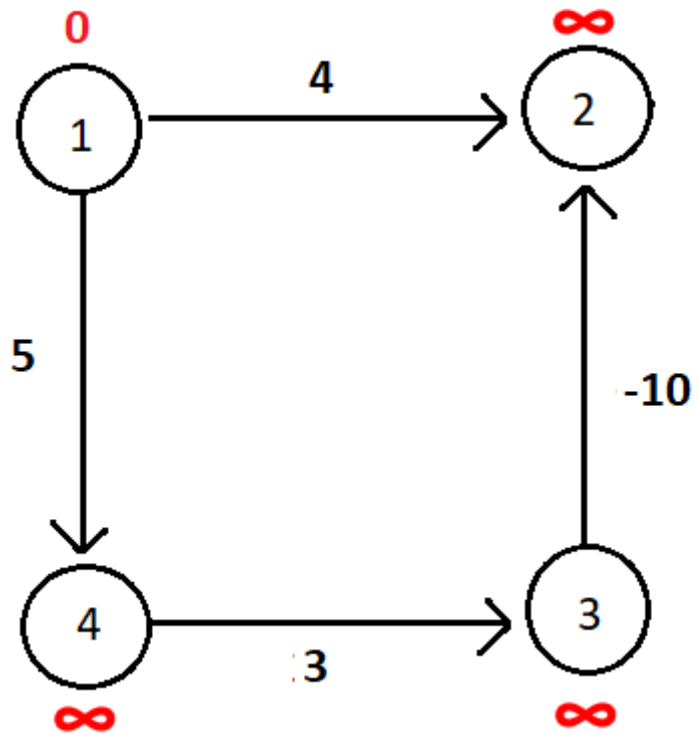
Bellman–Ford Algorithm
Example 2

- We are given the following graph, and 1 is our starting node.



Bellman–Ford Algorithm
Example 2

- Step 1: **prepare** a list of all edges in the graph and **set** the distance of the starting vertex to 0 and all other vertices to ∞.



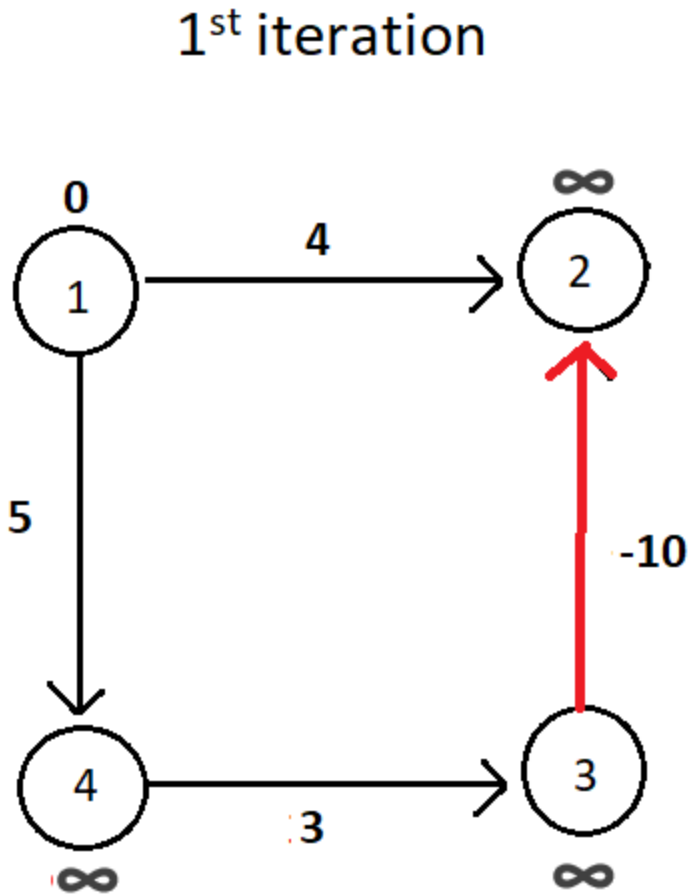
Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

Bellman–Ford Algorithm
Example 2

- Step 2: **relax** all the edges (|V| - 1) times.
- We have 4 vertices, then we should relax all the edges 3 times.

Bellman–Ford Algorithm
Example 2

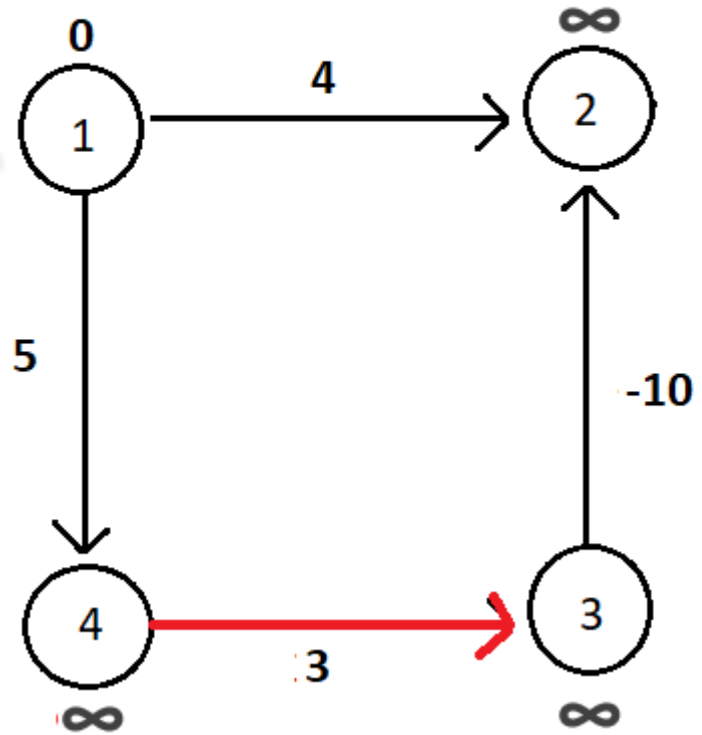


Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

Bellman–Ford Algorithm
Example 2

1st iteration

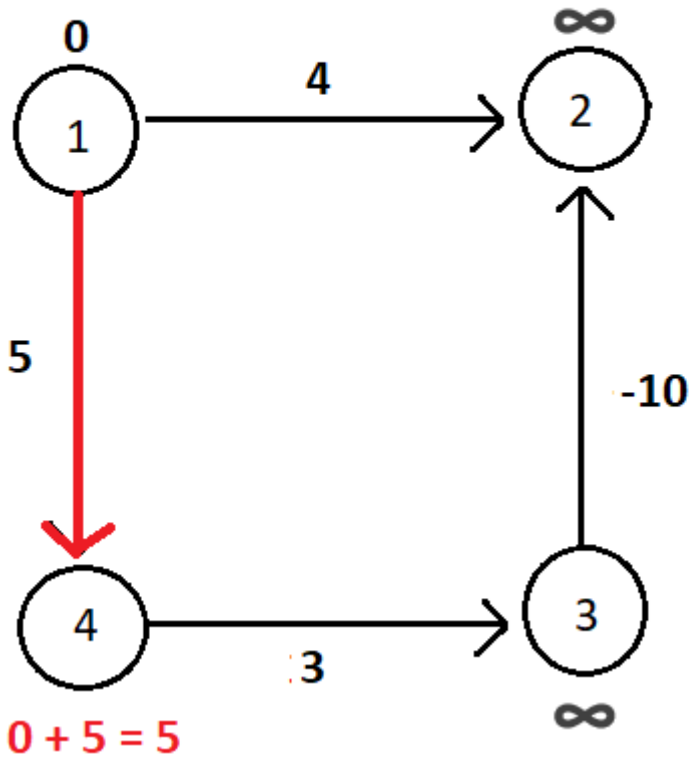


Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

Bellman–Ford Algorithm
Example 2

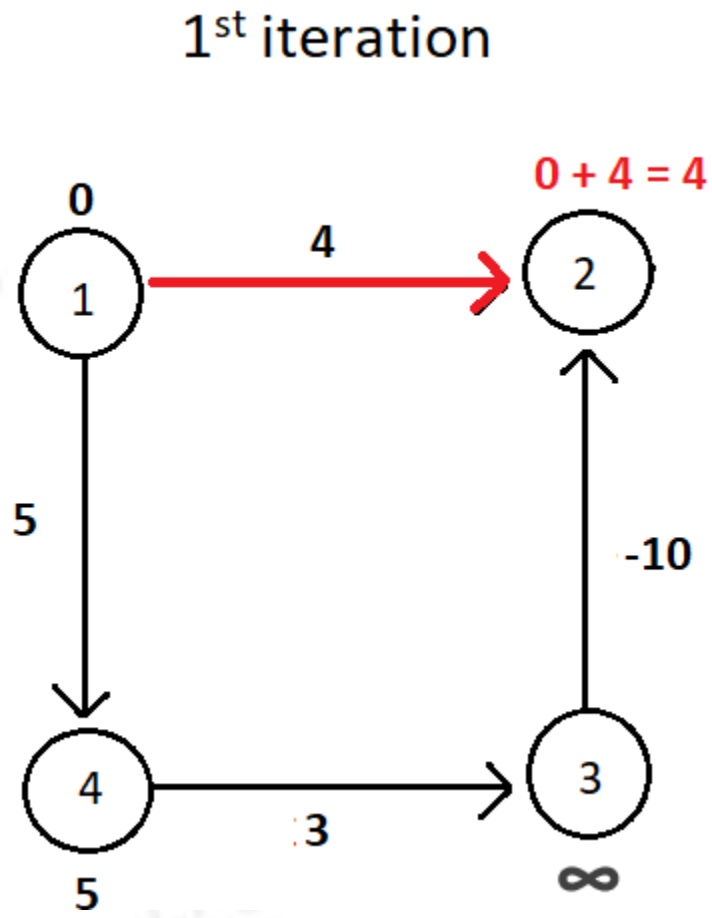
1st iteration



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

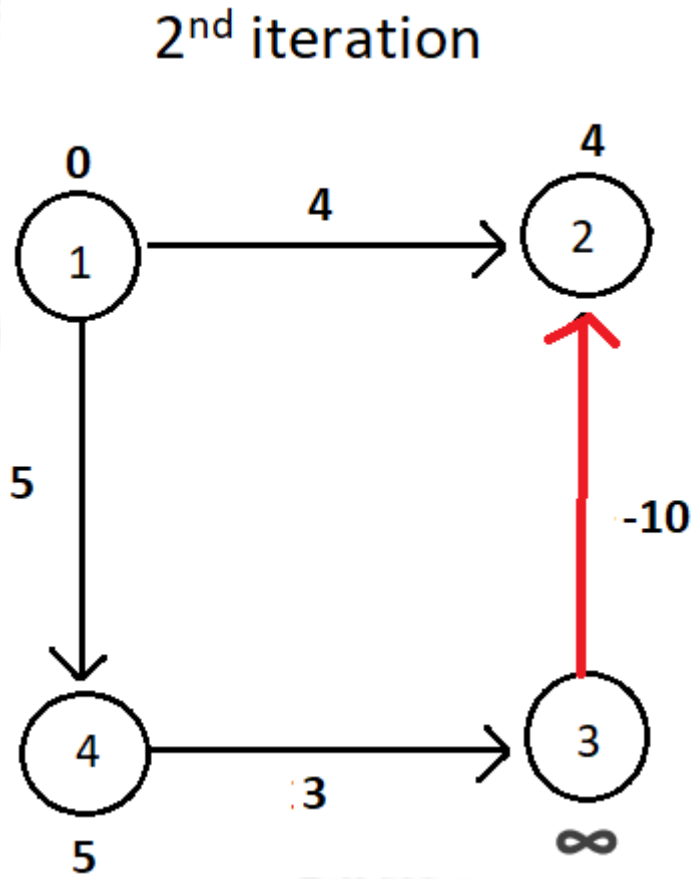
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

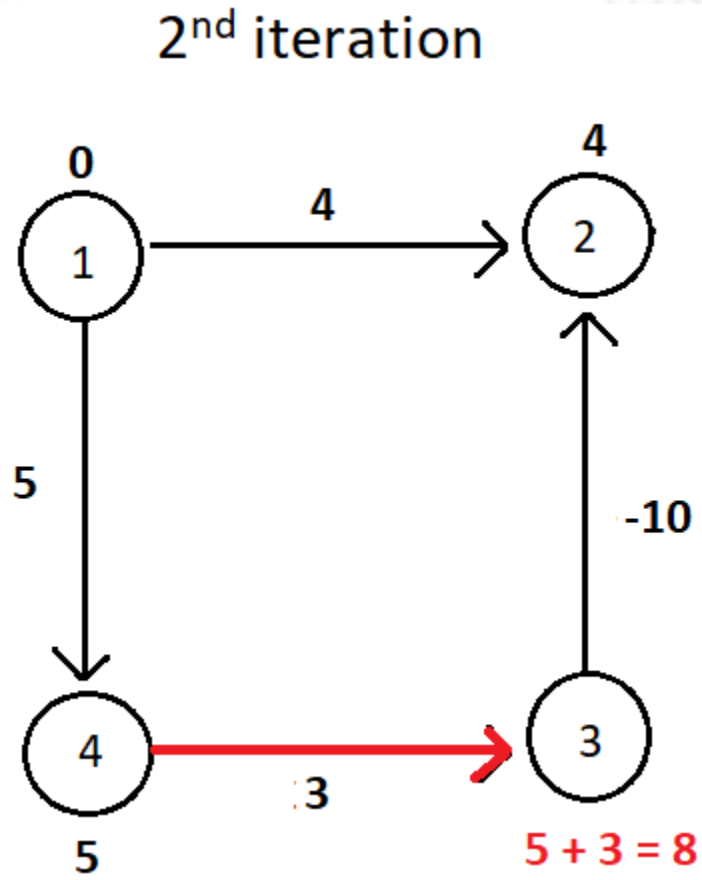
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

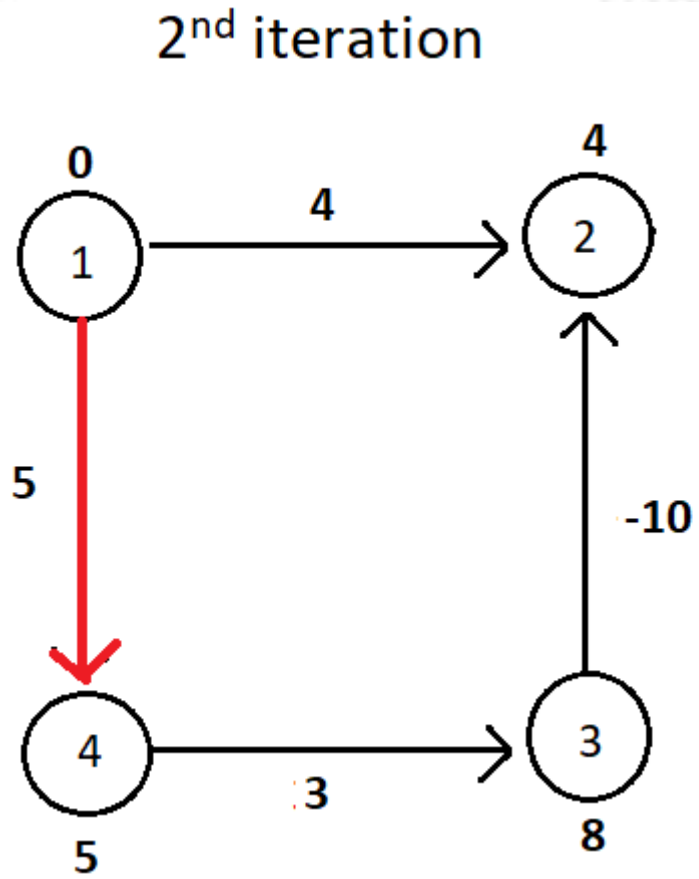
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

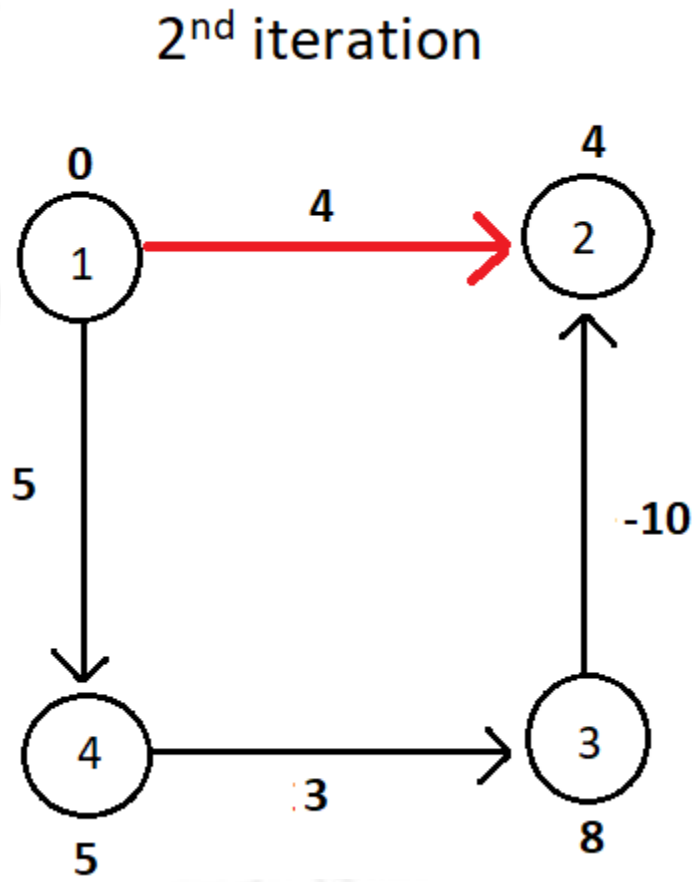
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

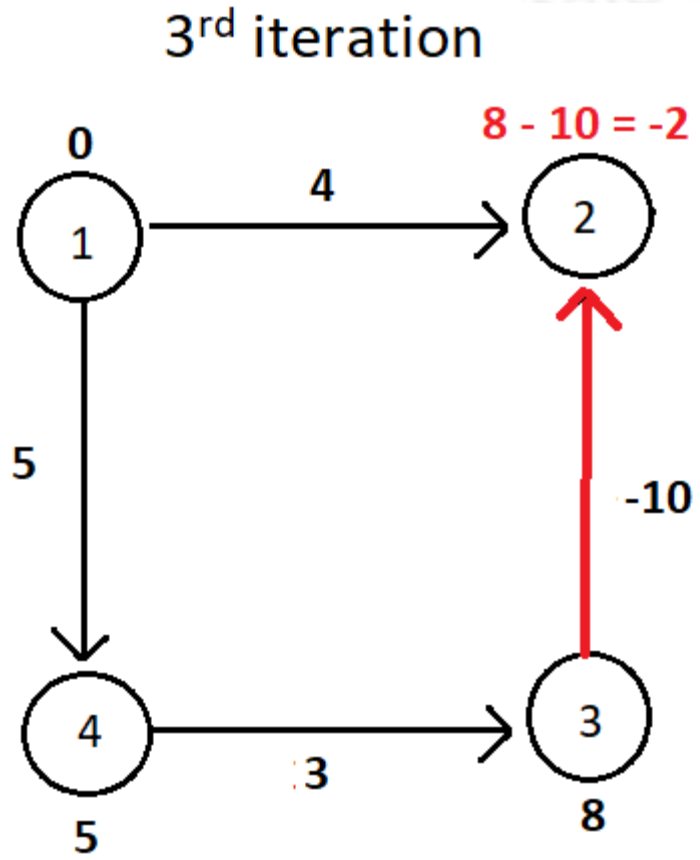
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

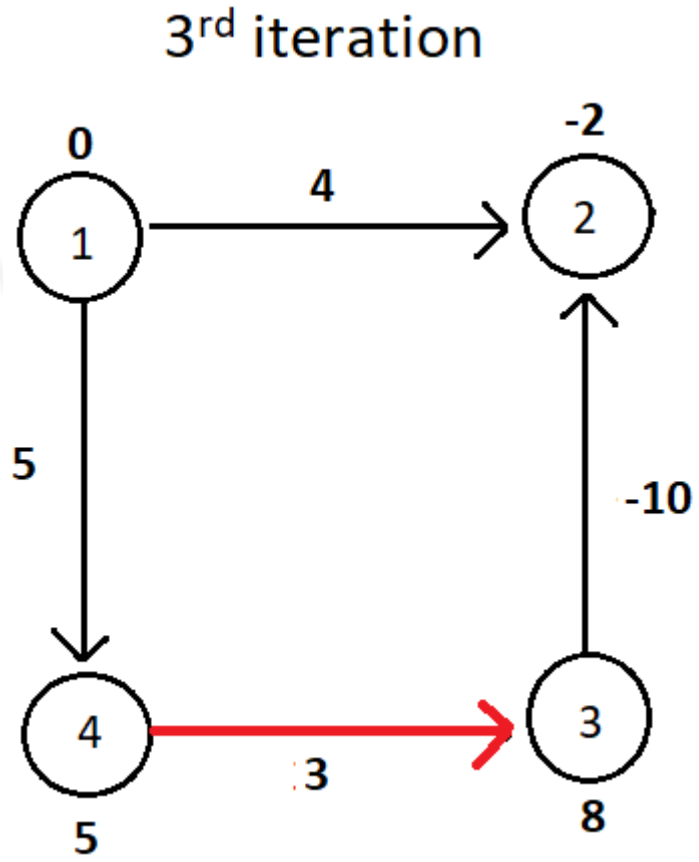
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

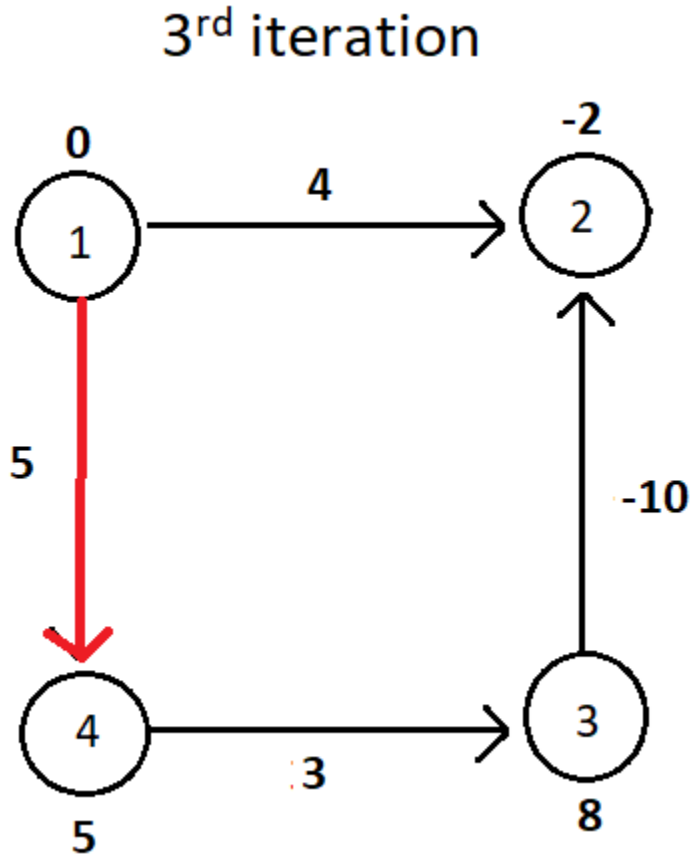
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

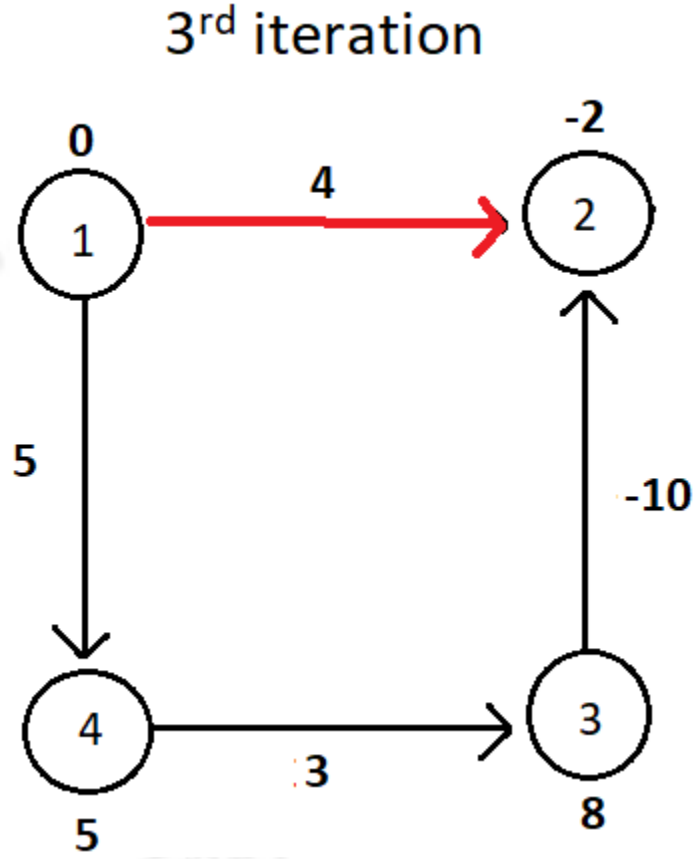
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

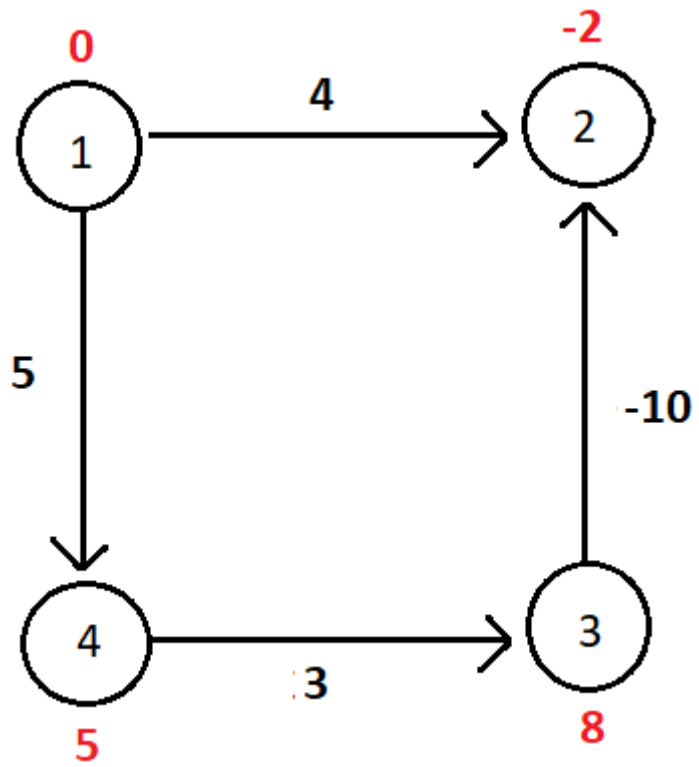
Bellman–Ford Algorithm
Example 2



Edges

(3, 2)
(4, 3)
(1, 4)
(1, 2)

Bellman–Ford Algorithm
Example 2



Any Questions???…