Shortest Path Algorithm

The Shortest Path algorithm calculates the shortest (weighted) path between a pair of nodes. It's useful for user interactions and dynamic workflows because it works in real time. Pathfinding has a history dating back to the 19th century and is considered to be a classic graph problem. It gained prominence in the early 1950s in the context of alternate routing; that is, finding the second-shortest route if the shortest route is blocked. In 1956, Edsger Dijkstra created the best-known of these algorithms. Dijkstra's Shortest Path algorithm operates by first finding the lowest-weight relationship from the start node to directly connected nodes. It keeps track of those weights and moves to the "closest" node. It then performs the same calculation, but now as a cumulative total from the start node. The algorithm continues to do this, evaluating a "wave" of cumulative weights and always choosing the lowest weighted cumulative path to advance along, until it reaches the destination node.

Dijkstra’s Shortest Path Algorithm in Concept

Let distance of start vertex from start vertex = 0
Let distance of all other vertices from start = ∞
Repeat
    Visit the unvisited vertex with the smallest known distance from the start vertex
    For the current vertex, examine its unvisited neighbours
    For the current vertex, calculate distance of each neighbour from start vertex
    If the calculated distance of the vertex is less than the known distance, update the shortest distance
    Update the previous vertex for each of the updated distances
    Add the current vertex to the list of updated vertices
Until all vertices finished

Dijkstra's Shortest Path Algorithm Formalism

Let G be a directed network with vertices V = {1, . . . ,n} such that, for each arc ij, its length $d_{ij} > 0$.

The essence of Dijkstra's algorithm is a labelling procedure.

At a typical stage of the algorithm we have V = P ∪ S, where

- P is the set of permanently labelled vertices, and
- S is the set of temporarily (or tentatively) labelled vertices.

To begin with, S = V , then as the algorithm proceeds, S gets smaller, while P gets bigger until eventually

all vertices have received permanent labels and the algorithm terminates.

In fact, each vertex u will have two labels:

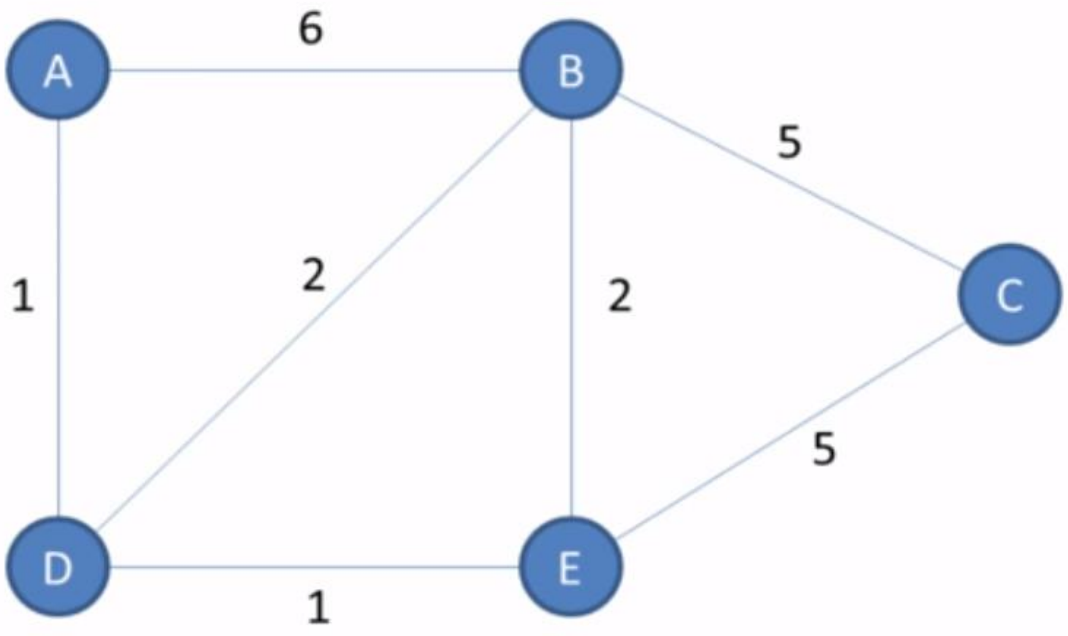
- D_u, our current guess at the distance from 1 to u;
- p_u, our current guess at the parent of u

Dijkstra's Shortest Path Algorithm Pseudo Code

Algorithm 5.2: *Dijkstra*(r, G, T)

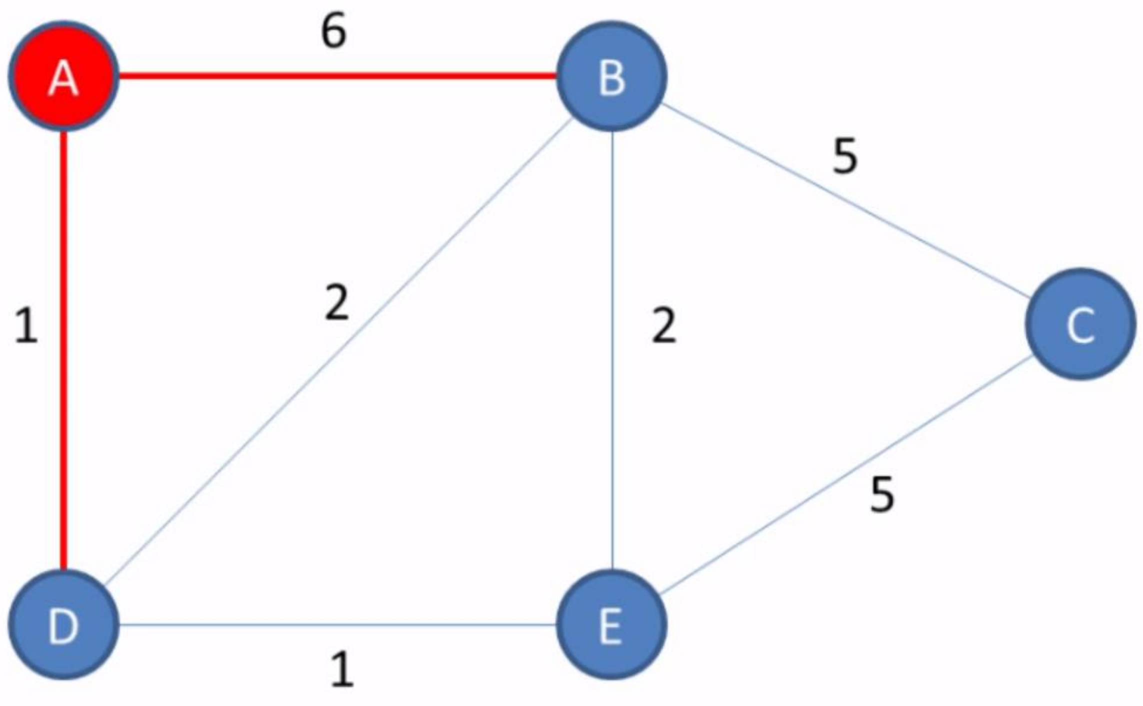
```
P := {1};   S := V ∖ P = {2, ..., n};   A := ∅;        // initialise P, S and A
D_1 := 0;
for each vertex u ∈ S do
    D_u := d_1u;   p_u := 1;                          // initialise D and p labels
end for
while S ≠ ∅ do
    Select u ∈ S with D_u = min_{v∈S} D_v;
    P := P ∪ {u};   S := S ∖ {u};                     // move u from S to P
    A := A ∪ {(p_u, u)};
    for all v ∈ S do                                  // note u is no longer in S
        if D_v > D_u + d_uv then
            D_v := D_u + d_uv;                        // relax inequality
            p_v := u;                                 // set the parent of v to be u
        end if
    end for
end while
```

Dijkstra’s Shortest Path Algorithm Solution



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

Dijkstra’s Shortest Path Algorithm Worked Example

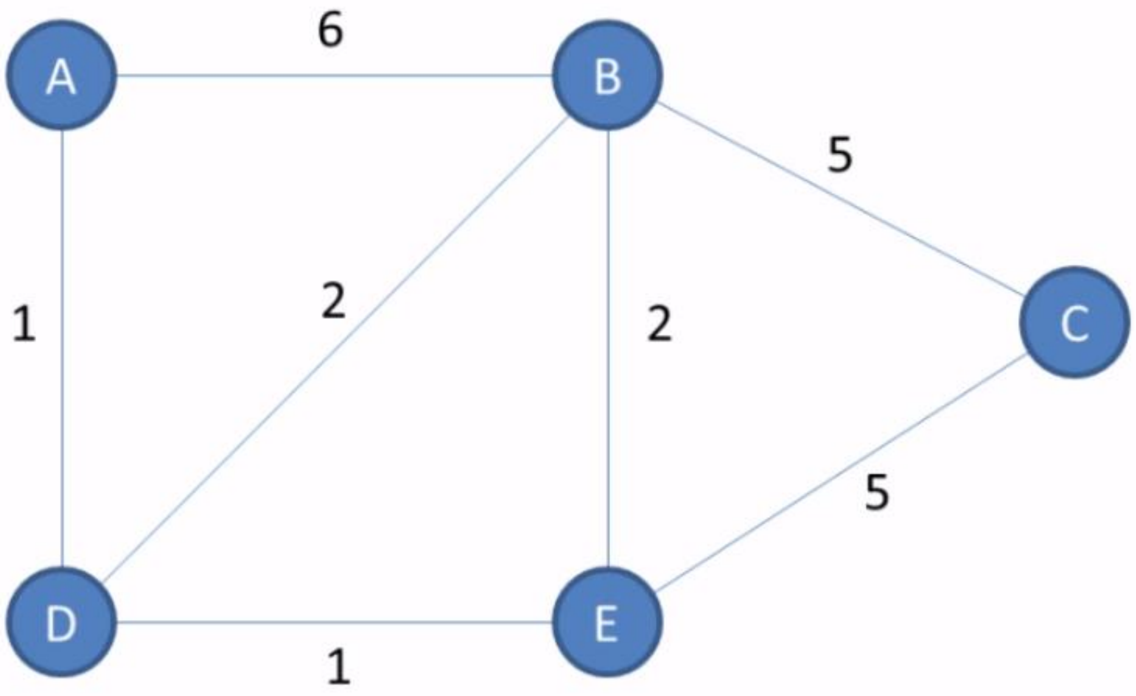


Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

Visited = [ ]

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

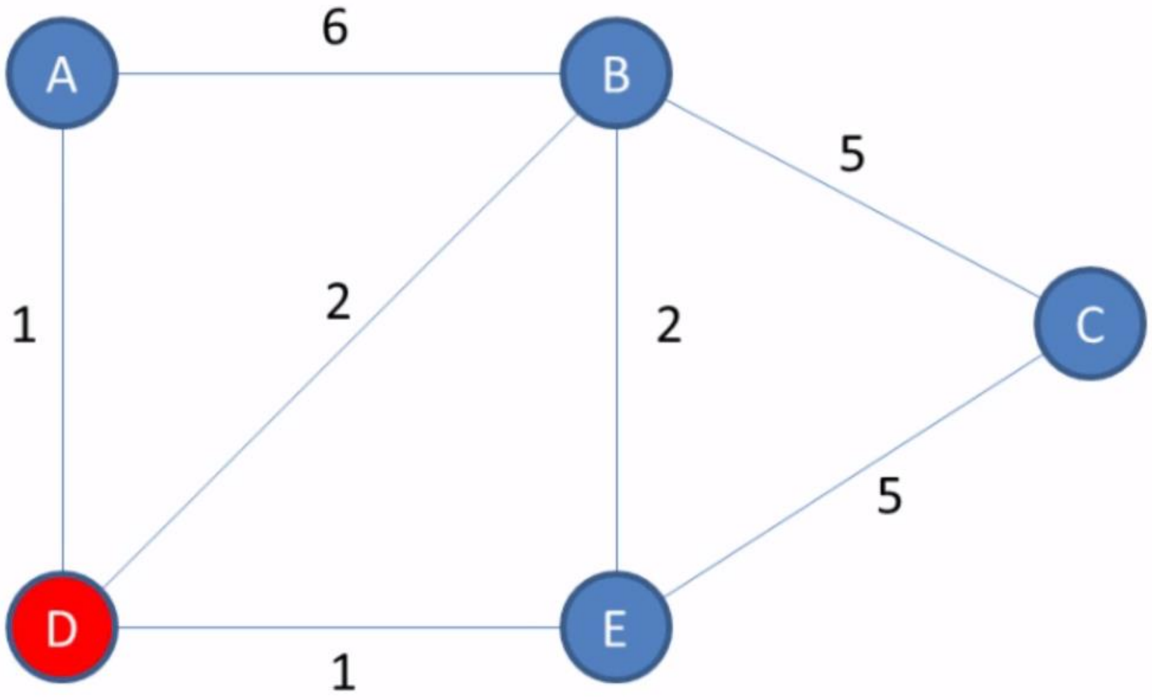
Dijkstra's Shortest Path Algorithm Worked Example



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 Shortest Path Algorithm Worked Example

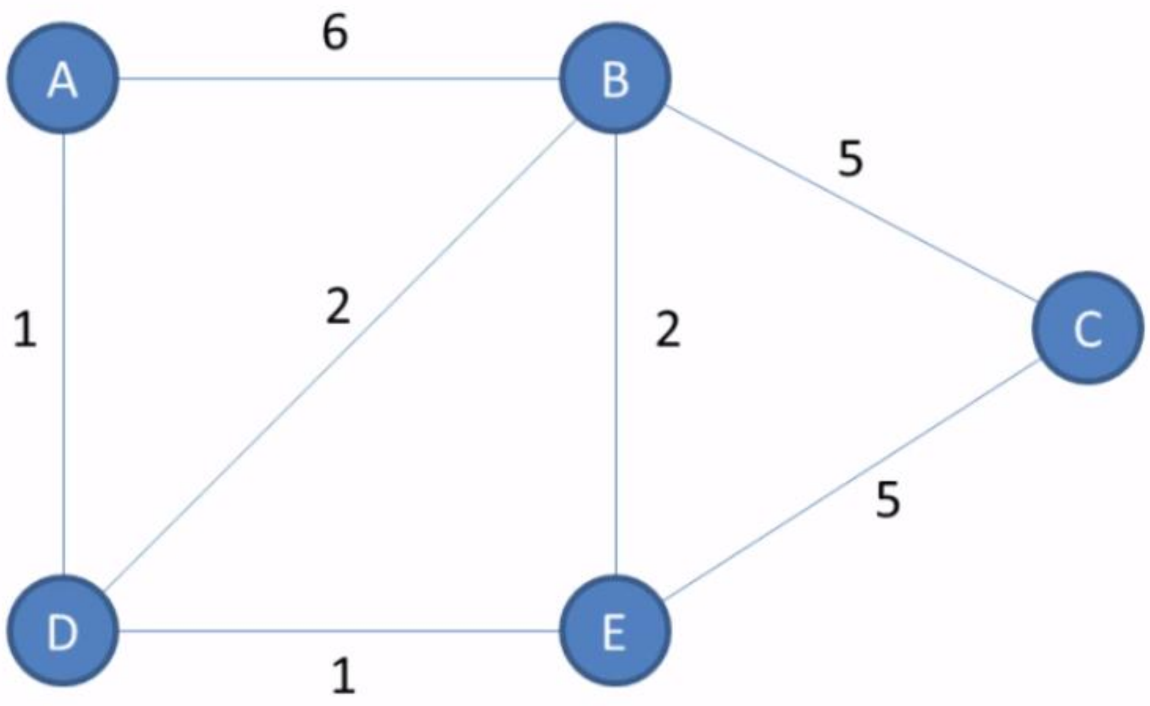


Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Visited = [A]

Unvisited = [B, C, D, E]

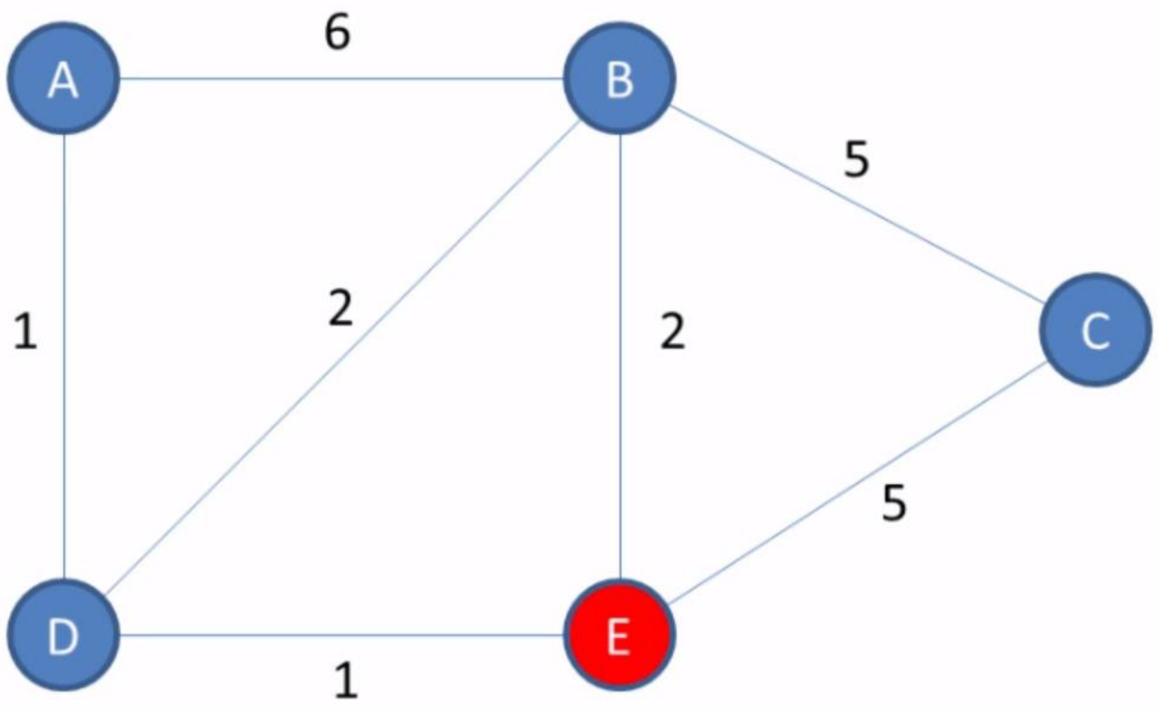
Dijkstra’s Shortest Path Algorithm Worked Example



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]

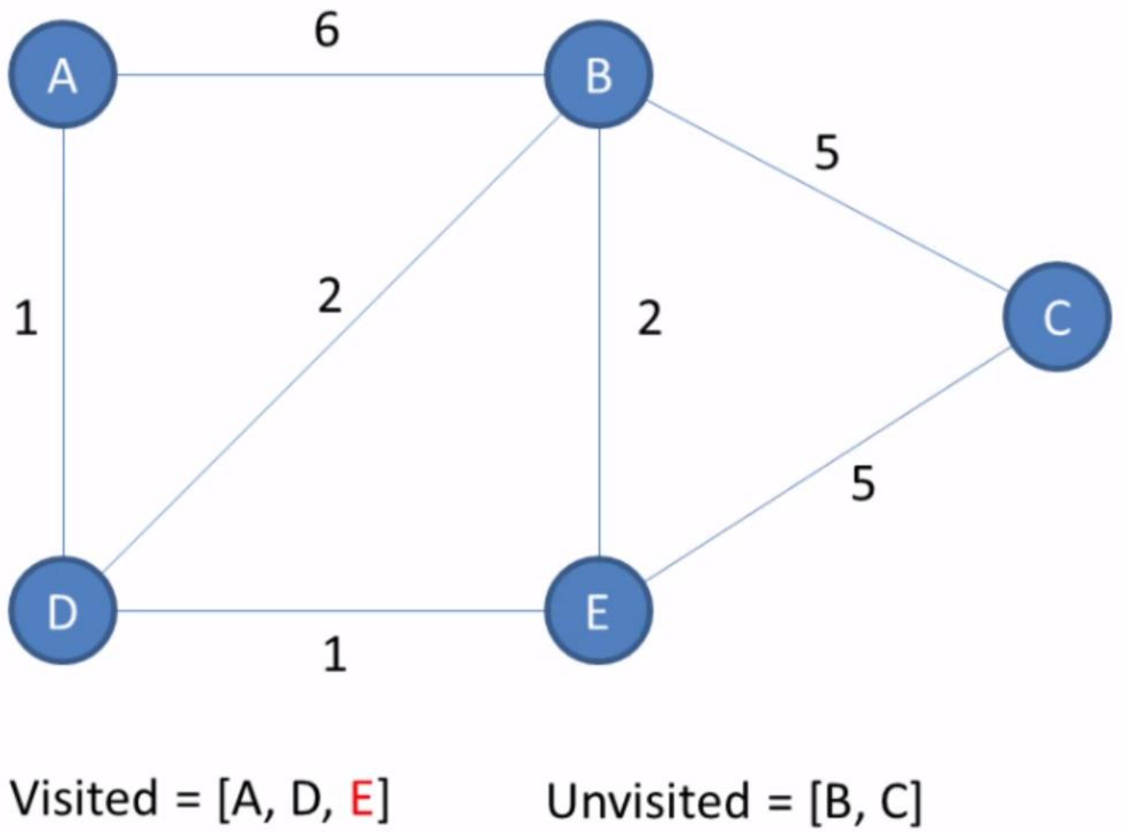
Dijkstra’s Shortest Path Algorithm Worked Example



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	$\boxed{2}$	D

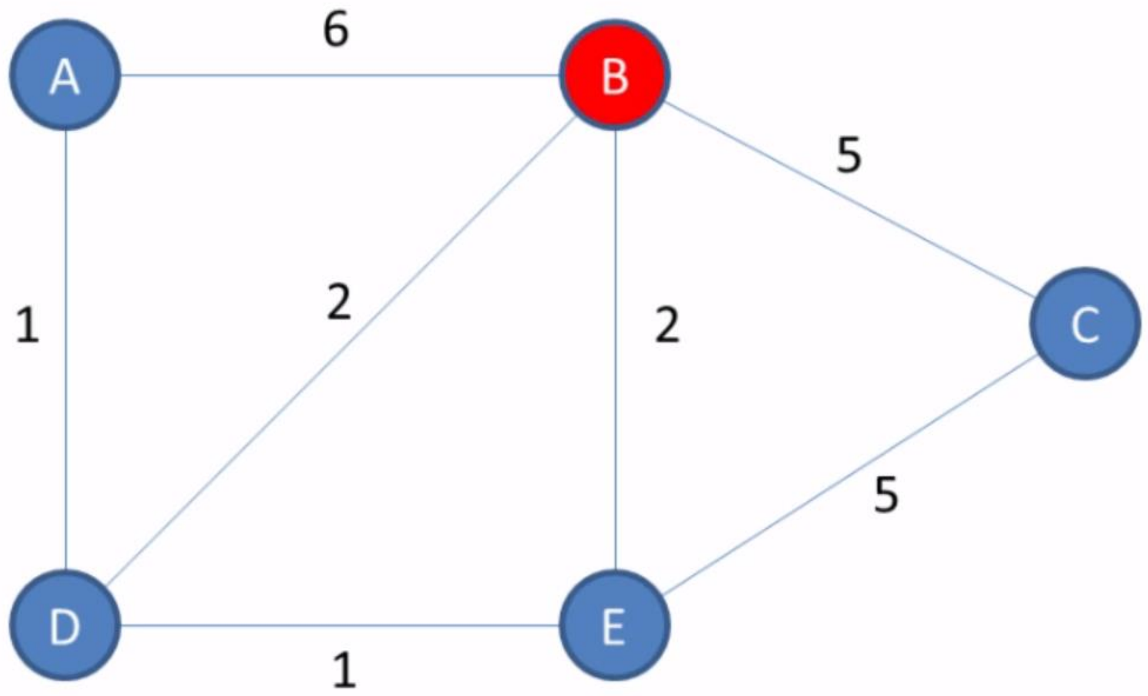
Visited = [A, D]   Unvisited = [B, C, E]

Dijkstra’s Shortest Path Algorithm Worked Example



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

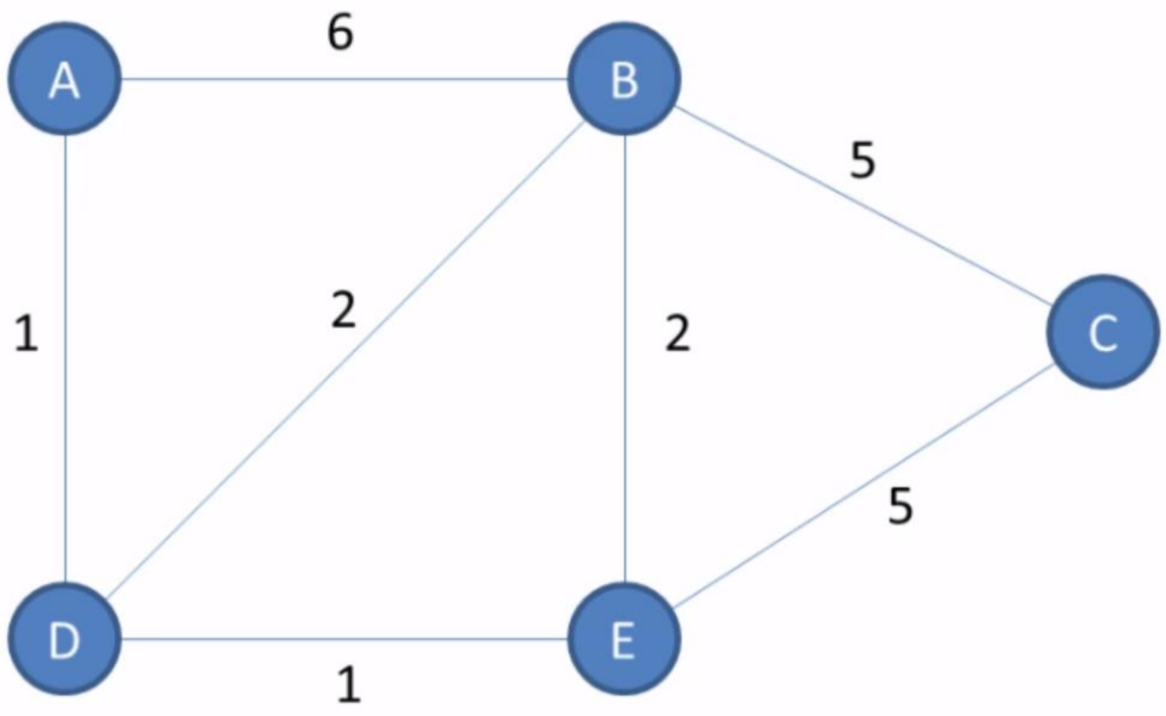
Dijkstra’s Shortest Path Algorithm Worked Example



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

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

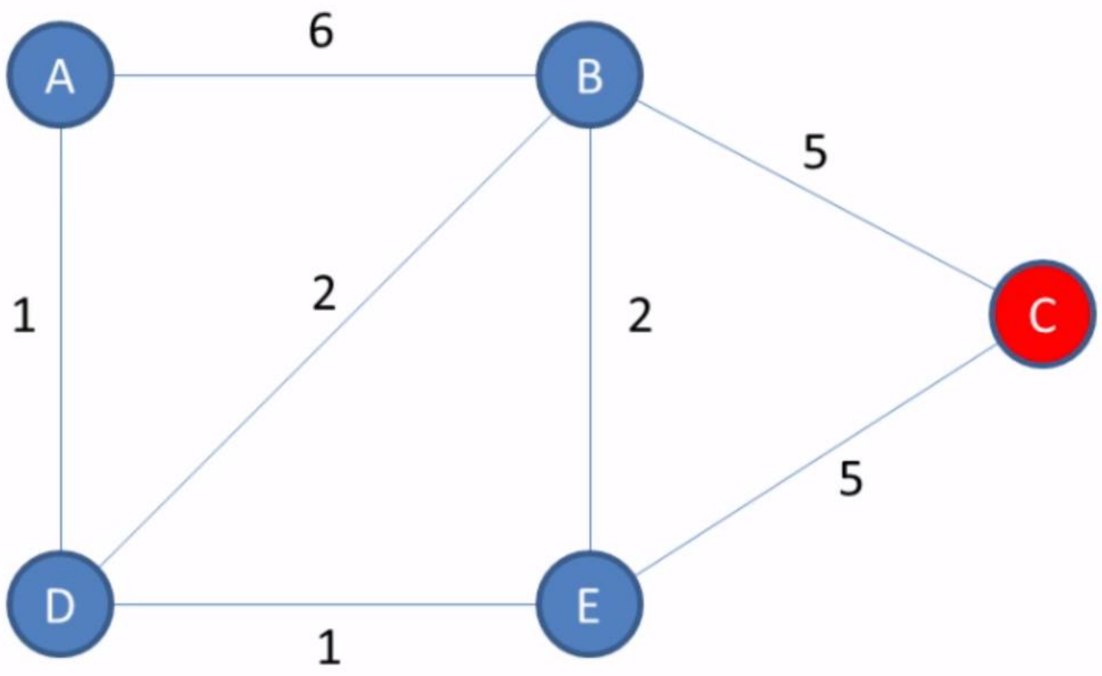
Dijkstra’s Shortest Path Algorithm Worked Example



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]

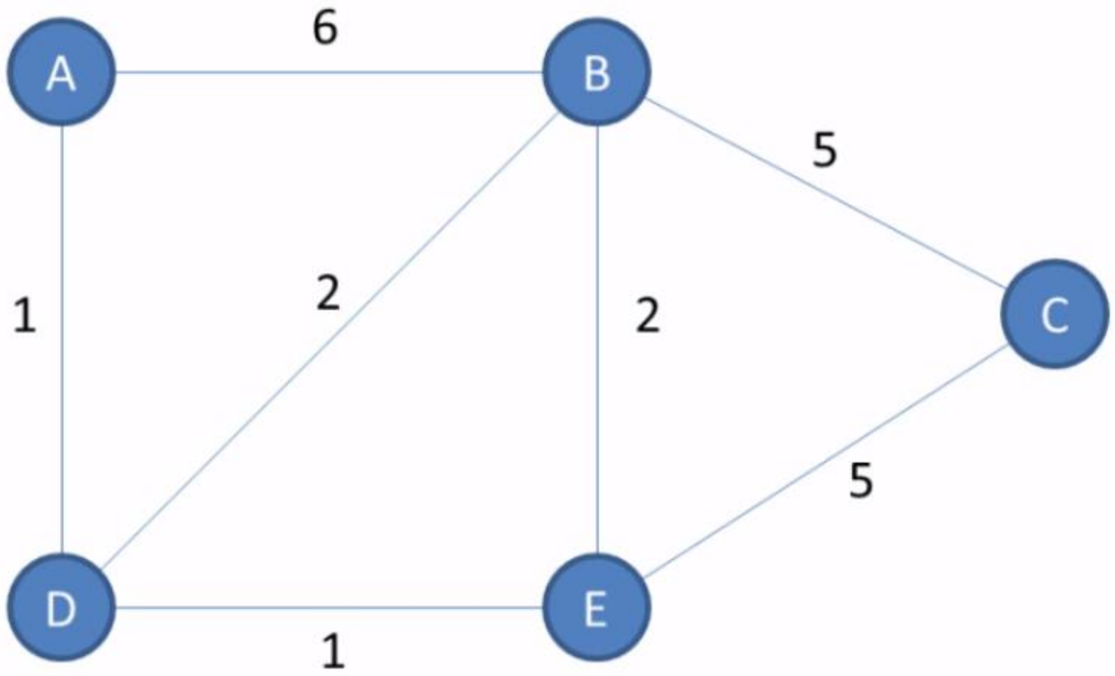
Dijkstra’s Shortest Path Algorithm Worked Example



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 Shortest Path Algorithm Worked Example



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 = []