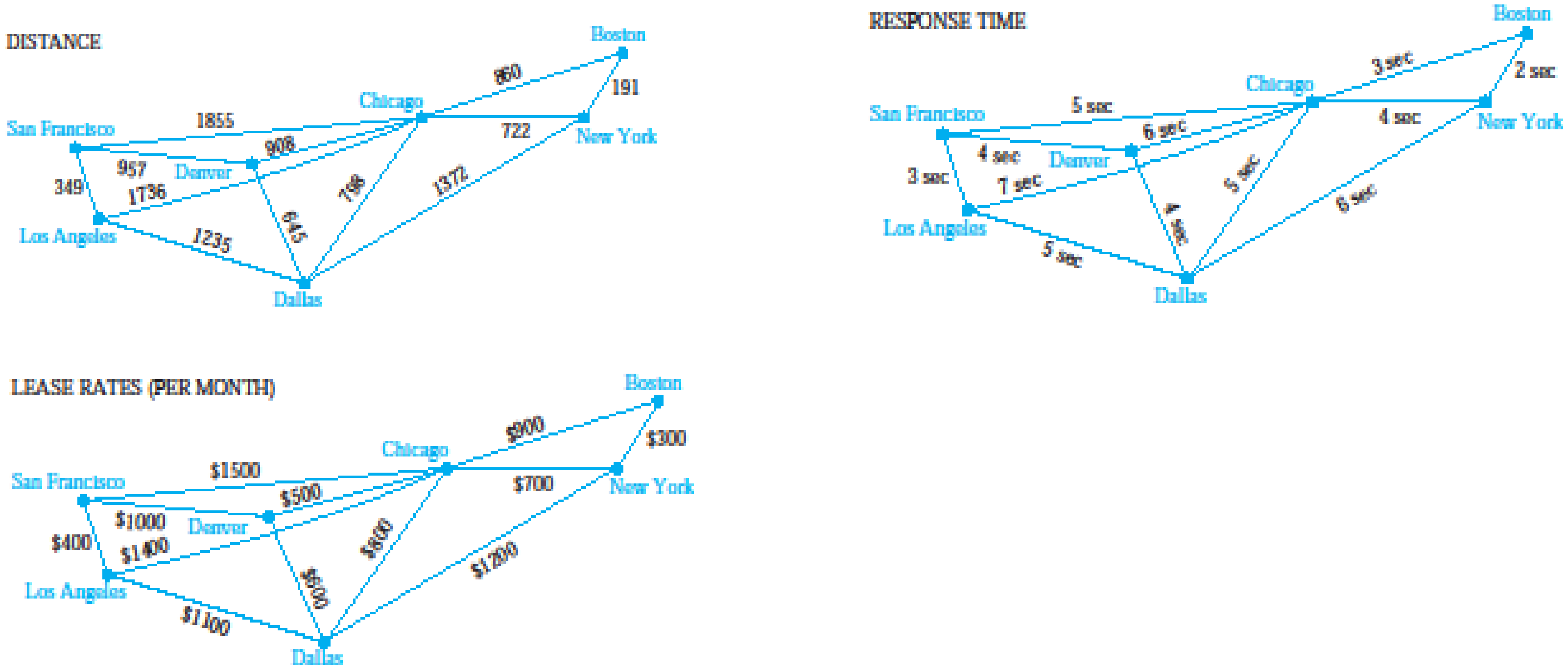
10.6 Shortest Path Problems

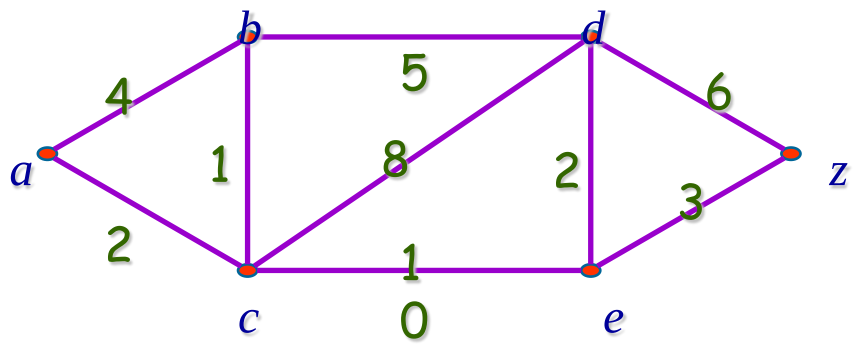
Introduction

- **Many problems can be modeled using graphs with weights assigned to their edges.**
  - **Weighted Graphs Modeling a Computer Network**



Some definitions

- **Weighted graph:** $G = (V, E, W)$
- **the length of a path in a weighted graph**
  - The sum of the weights of the edges of this path



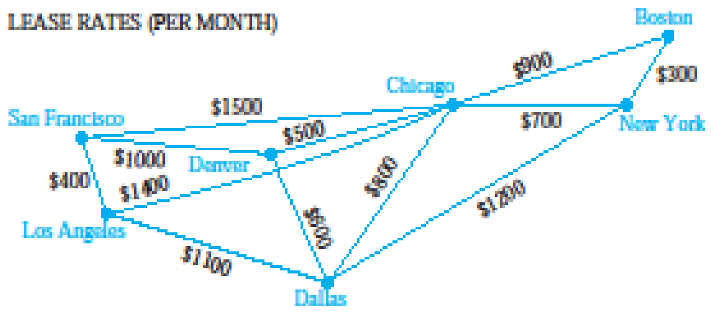
Shortest Path Problem

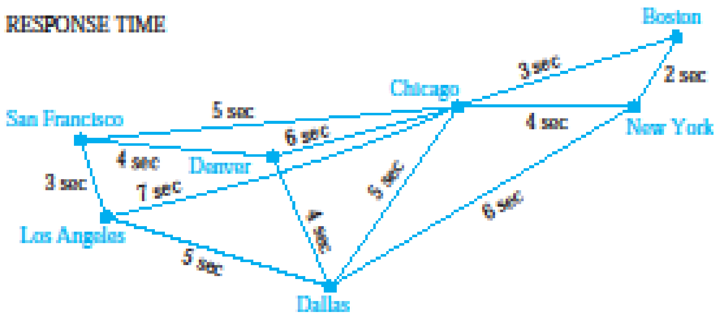
◆**Some problems involving weighted graphs arise frequently.**

The weighted Graph of Computer Network:

- **What is a least expensive set of telephone lines needed to connect the computers in San Francisco with those in New York?**
- **Which set of telephone lines gives a fastest response time for communications between San Francisco and New York?**

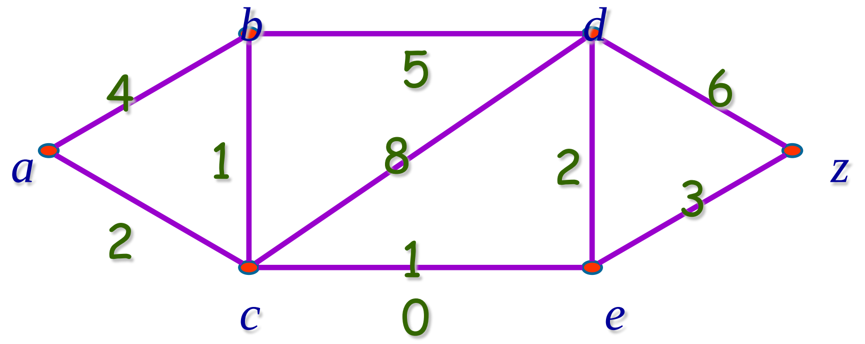
What is a shortest path between two given vertices?





The Description of a Shortest Path Problem

$G = (V, E, W)$ **is a weighted graph, where** $w(x, y)$ **is the weight of edge associated vertices *x* and *y* (if** $(x, y) \notin E, w(x, y) = \infty$**),** $a, z \in V$**, find the shortest path between *a* and *z*.**



More problems:

- The shortest path from a to all other vertices of the graph
- The shortest path between all pairs of vertices in a weighted connected simple graph
- Weights: all are 1, positive, or arbitrary real numbers

A Shortest-path Algorithm

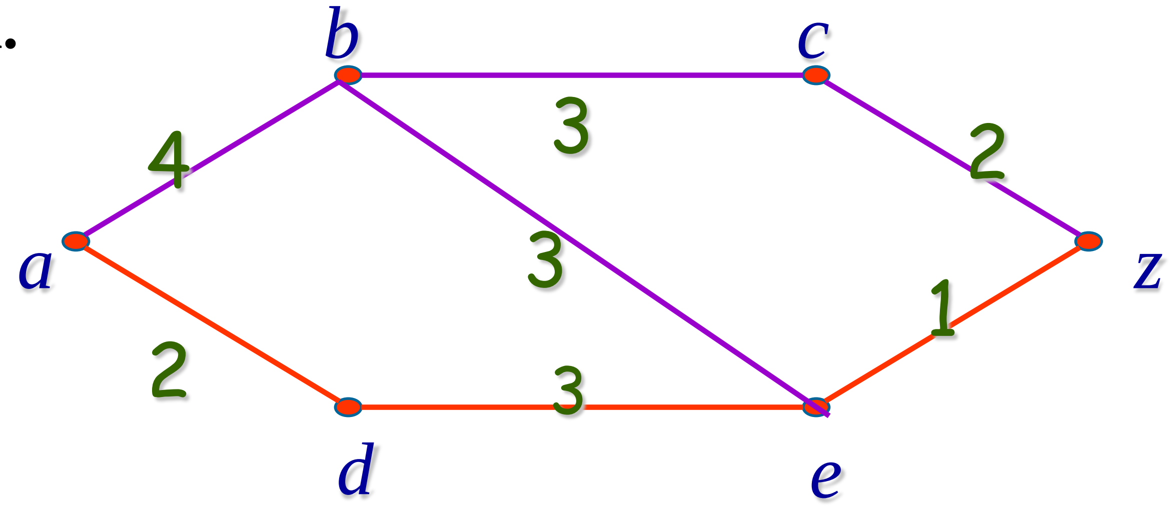
Dijkstra's Algorithm

- A greedy algorithm discovered by the Dutch mathematician E. Dijkstra in 1959.
- To solve the problem in undirected weighted graphs **where all the weights are positive**.
- Main idea:

  Proceed by **finding the length of the shortest path from *a* to** a first vertex, the length of the shortest path from *a* to a second vertex, and so on, until the length of the shortest path from *a* to *z*.

The General Principles Used in Dijkstra's Algorithm

〖Example 1〗 **What is the length of the shortest path between *a* and *z* in the weighted graph.**



Solution:

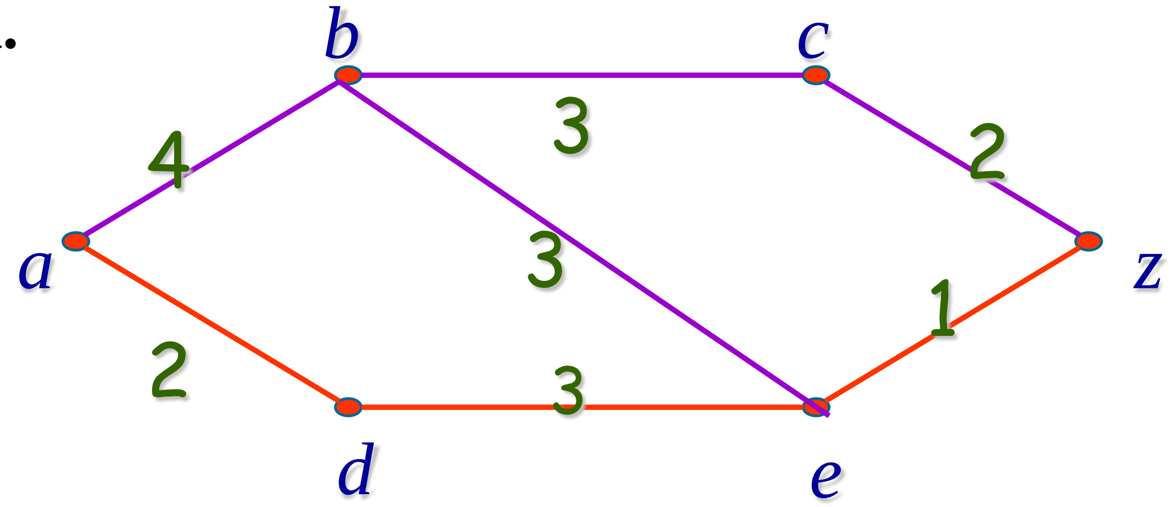
We will solve this problem by finding the length of a shortest path from *a* to successive vertices, until *z* is reached.

1) The first closest vertex: *d*

The only paths starting at *a* that contain no vertex other than *a* are *a*, *b* and *a*, *d*.

The General Principles Used in Dijkstra's Algorithm

〖Example 1〗What is the length of the shortest path between *a* and *z* in the weighted graph.



Solution:

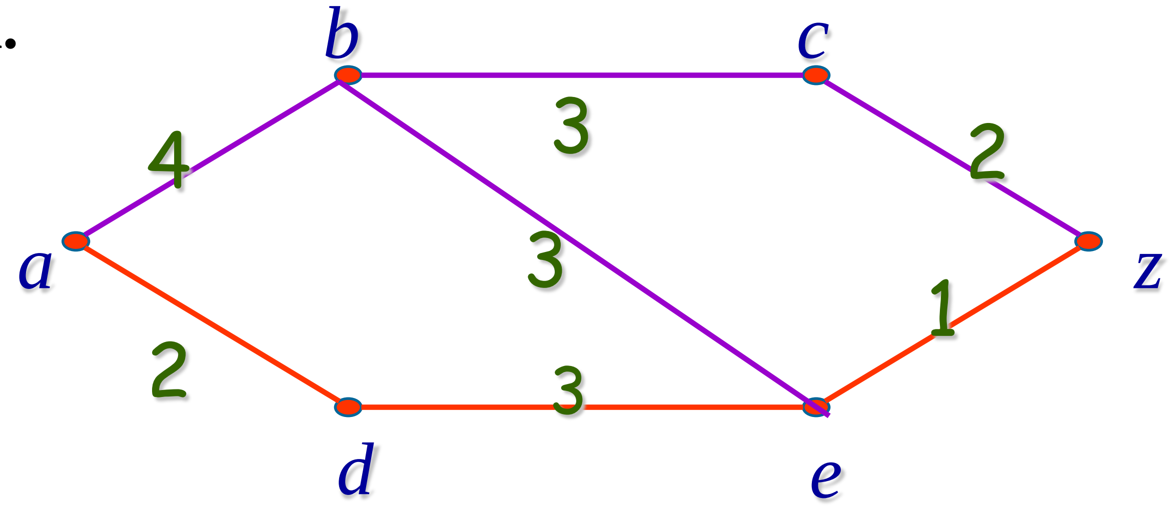
1) The first closest vertex: *d*

2) The second closest vertex: *b*

Looking at all paths that go through only *a* and *d*.

The General Principles Used in Dijkstra's Algorithm

〖Example 1〗 What is the length of the shortest path between *a* and *z* in the weighted graph.



Solution:

1) The first closest vertex: *d*

2) The second closest vertex: *b*

3) The third closest vertex: *e*

Examine only paths go through only *a*, *d* and *b*.

4) The forth closest vertex: *z*

The Details of Dijkstra's algorithm

Let S_k denote the set of vertices after k iterations of labeling procedure.

1. Initialization. Label a with 0 and other with ∞, i.e. $L_0(a)=0$, and $L_0(v)= \infty$ and $S_0=\phi$.
2. Form S_k. The set S_k is formed from S_{k-1} by adding a vertex u not in S_{k-1} with the smallest label.
3. Update the labels of all vertices not in S_k , so that $L_k(v)$, the label of the vertex v at the kth stage, is the length of the shortest path from a to v that containing vertices only in S_k .
4. Step 2 and 3 is iterated by successively adding vertices to the distinguished set the until z is added.

The Details of Dijkstra's algorithm

- **Update the labels of all vertices not in S_k**

  Let v be a vertex not in S_k , $L_k(v)$ is the shortest path from a to v containing only vertices in S_k

  This shortest path is either

  ✓ the shortest path from a to v containing only elements of S_{k-1}

  Or

  ✓ it is the shortest path from a to u at the $(k-1)$st stage with the edge (u,v) added.

$$L_k(v)=\min\{L_{k-1}(v),\ L_{k-1}(u)+w(u,v)\}$$

Pseudocode for Dijkstra's algorithm

Algorithm 1 Dijkstra's Algorithm.

Procedure Dijkstra(G: weighted connected simple graph, with all weights positive)

{G has vertices $a = v_0, v_1, \cdot \cdot \cdot, v_n = z$ and weights $w(v_i, v_j)$, where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}

For $i := 1$ to n

    $L(v_i) := \infty$

$L(a) := 0$

$S := \emptyset$

{the labels are now initialized so that the label of a is zero and all other labels are ∞, and S is the empty set }

Pseudocode for Dijkstra's algorithm

While $z \notin S$

Begin

u :=a vertex not in S with $L(u)$ minimal

$S := S \cup \{u\}$

for all vertices v not in S

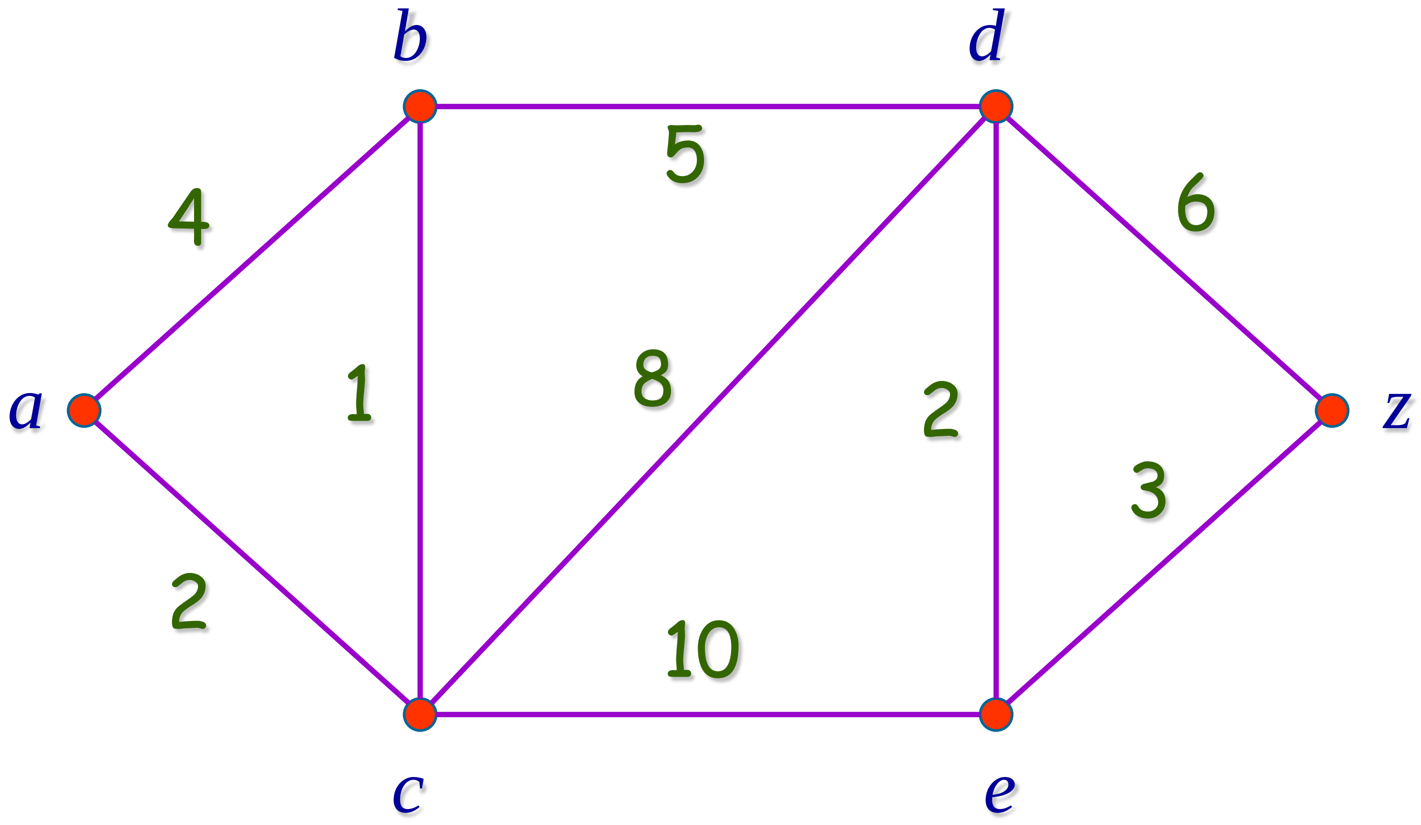
if $L(u) + w(u,v) < L(v)$ $L(v) := L(u) + w(u,v)$

{this adds a vertex to S with minimal label and updates the labels of vertices not in S}

End {$L(z)$=length of shortest path from a to z }

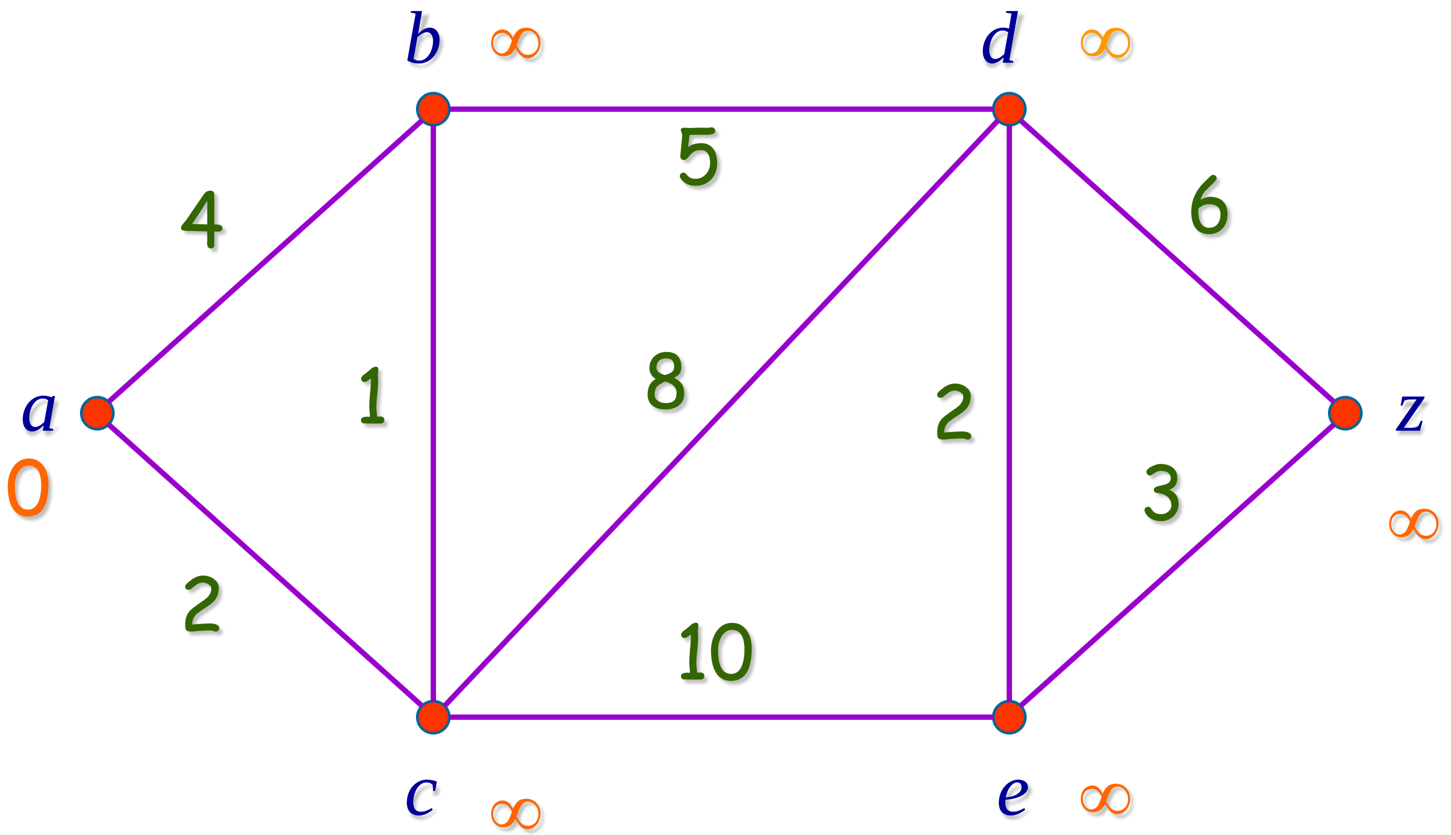
How Does Dijkstra's Algorithm Work?

〖Example 2〗 Find the length of the shortest path between *a* and *z* in the given weighted graph.



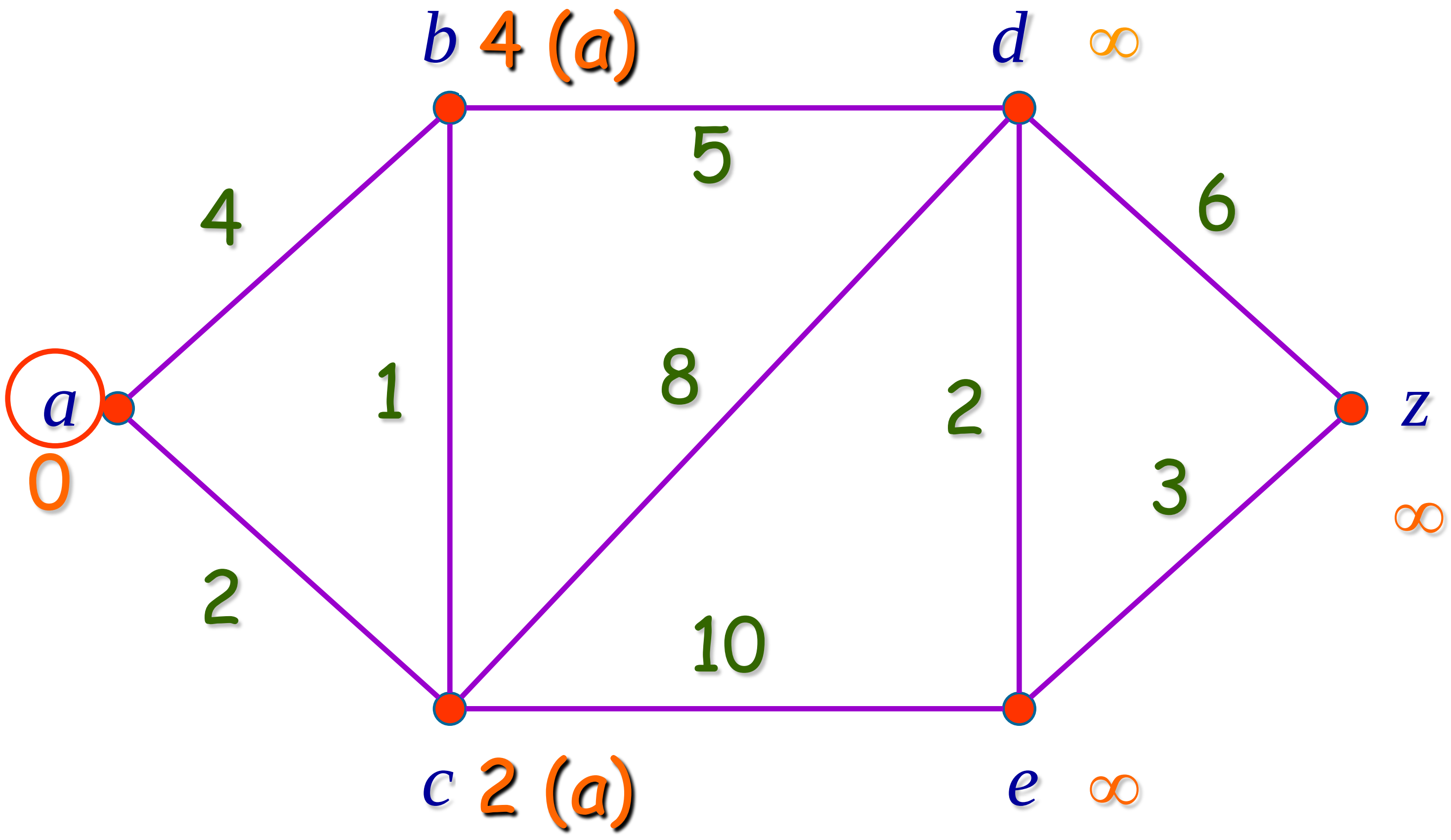
How Does Dijkstra's Algorithm Work?

Step 0



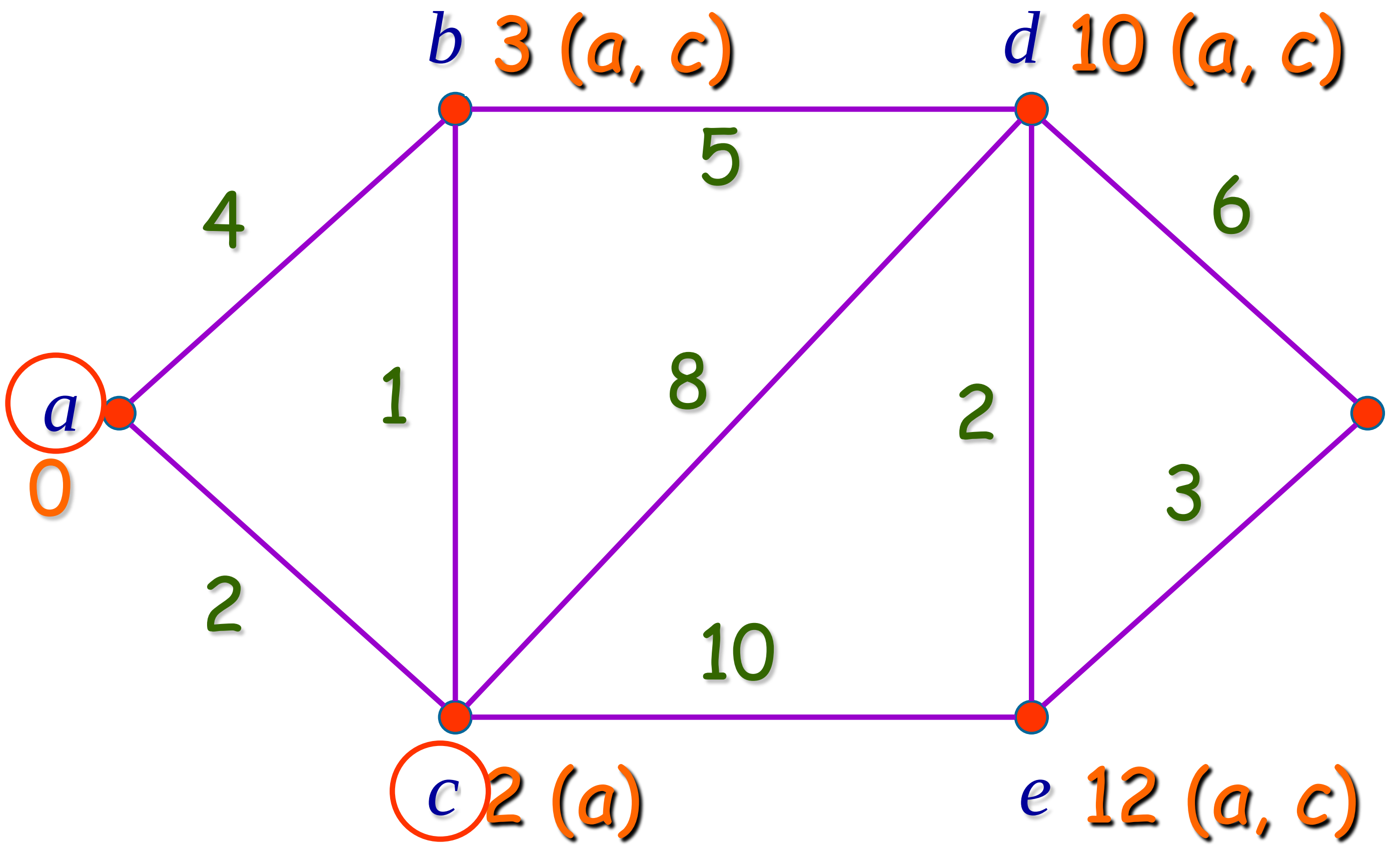
How Does Dijkstra's Algorithm Work?

Step 1



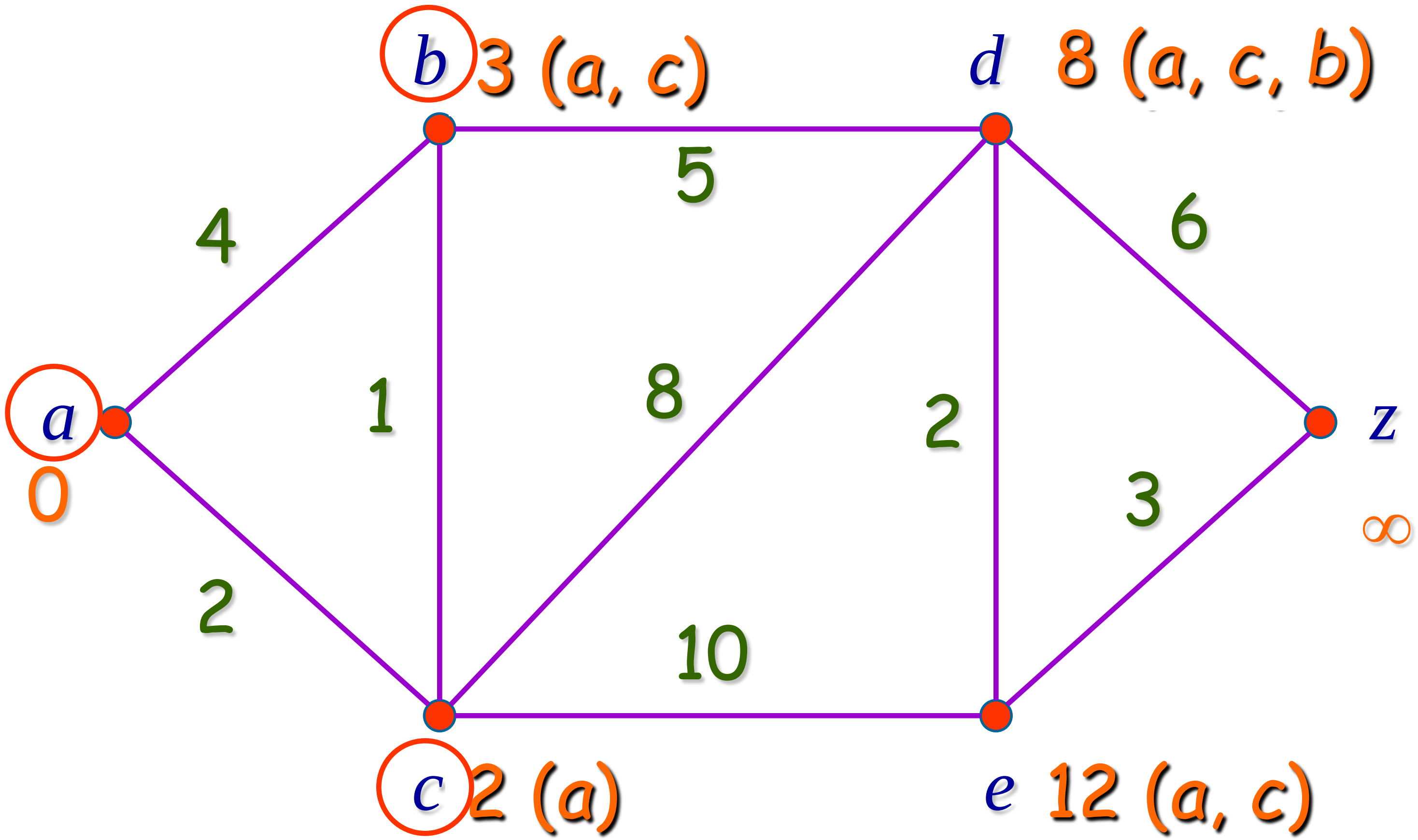
How Does Dijkstra's Algorithm Work?

Step 2



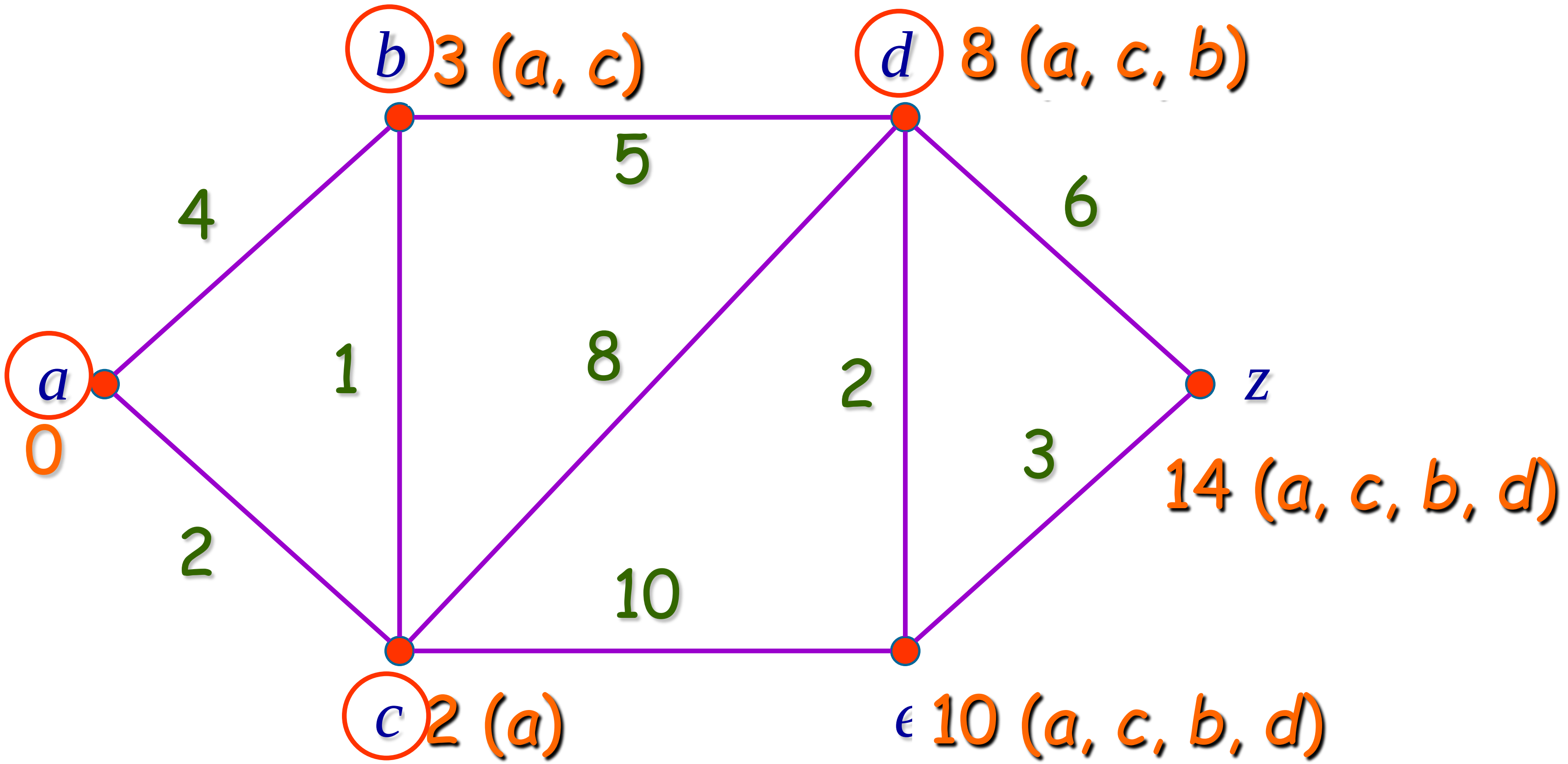
How Does Dijkstra's Algorithm Work?

Step 3



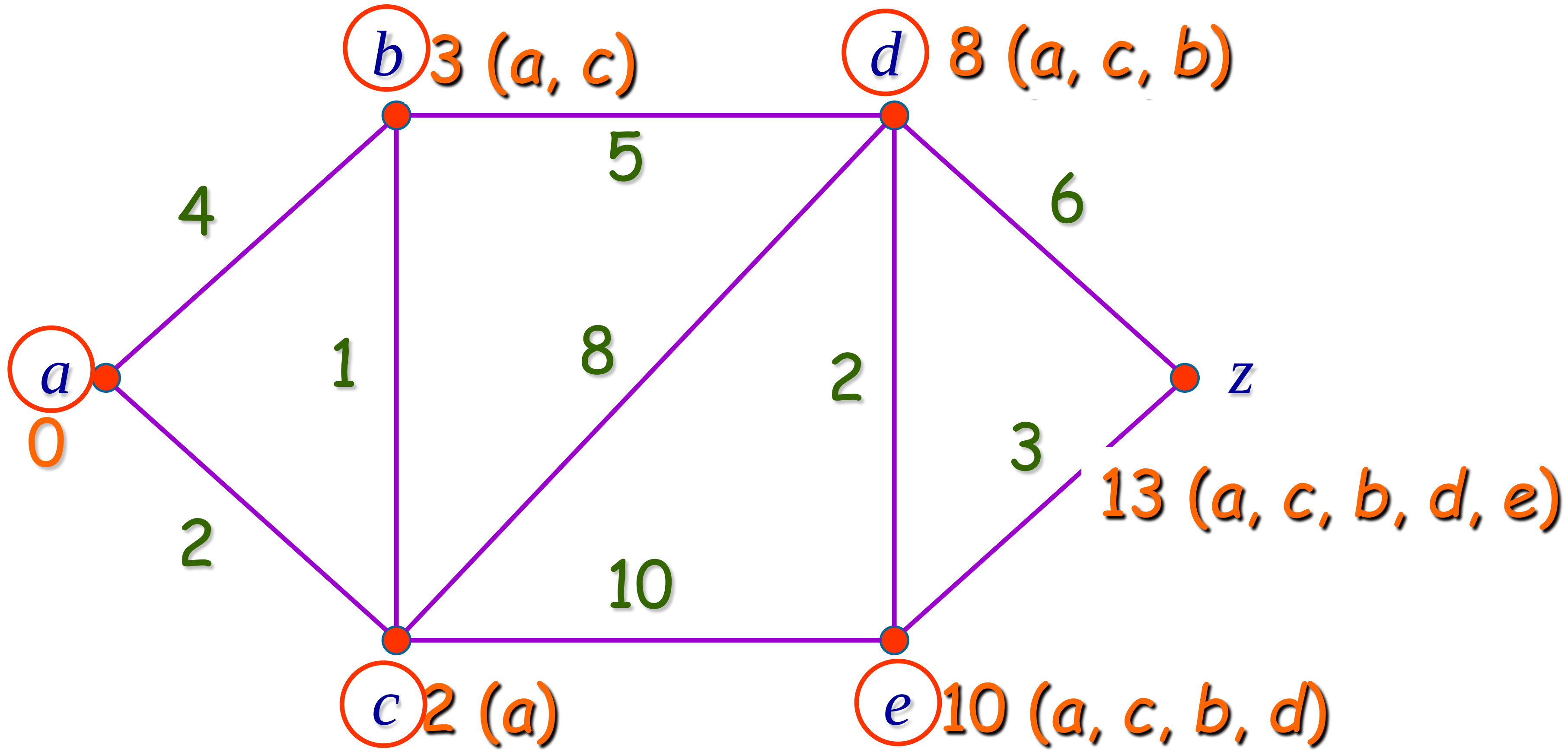
How Does Dijkstra's Algorithm Work?

Step 4



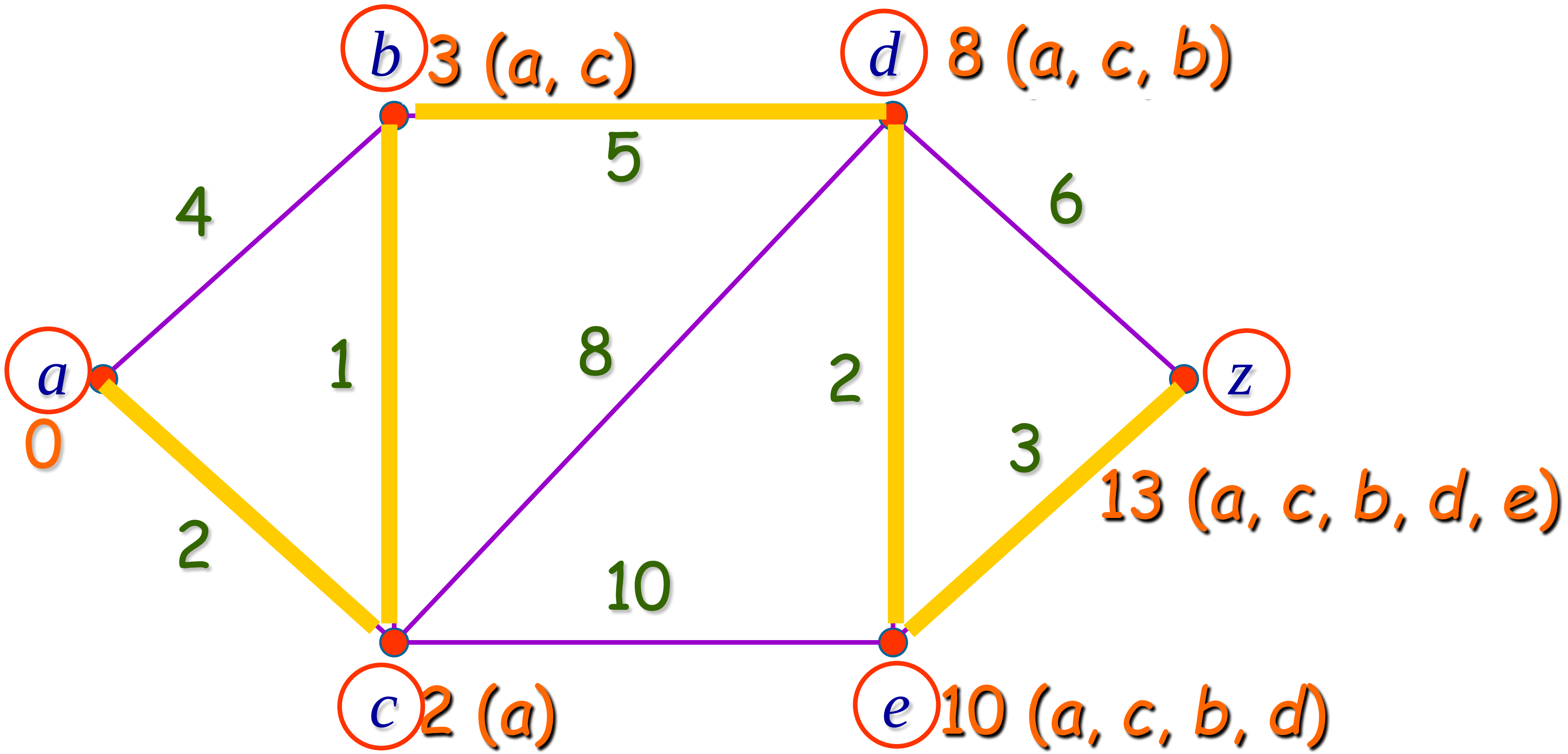
How Does Dijkstra's Algorithm Work?

Step 5

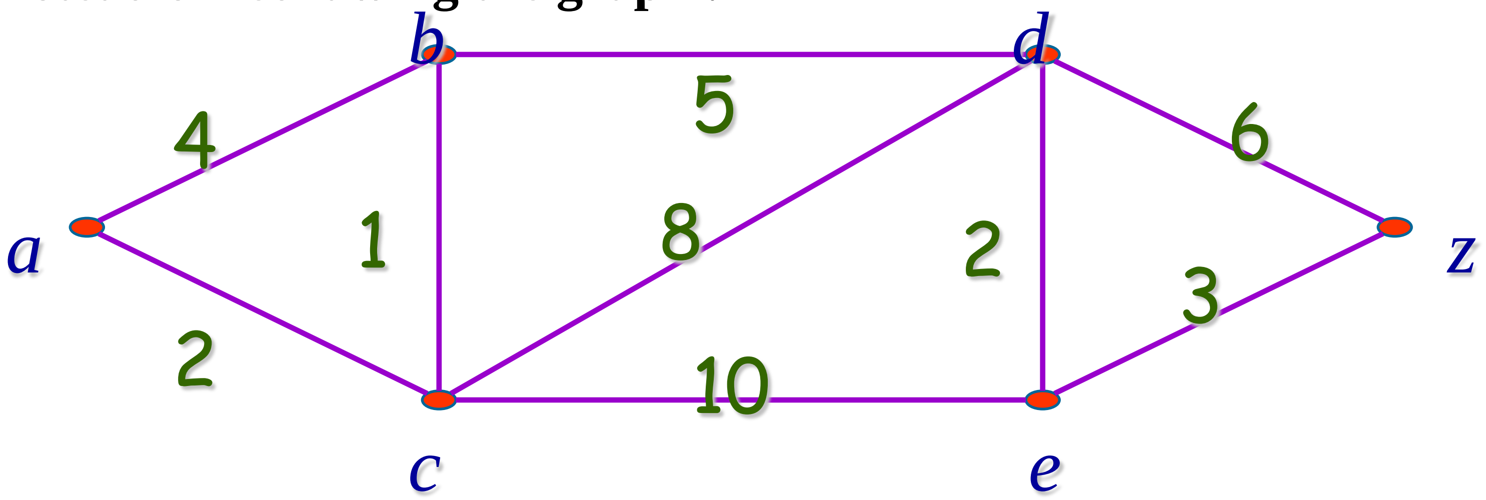


How Does Dijkstra's Algorithm Work?

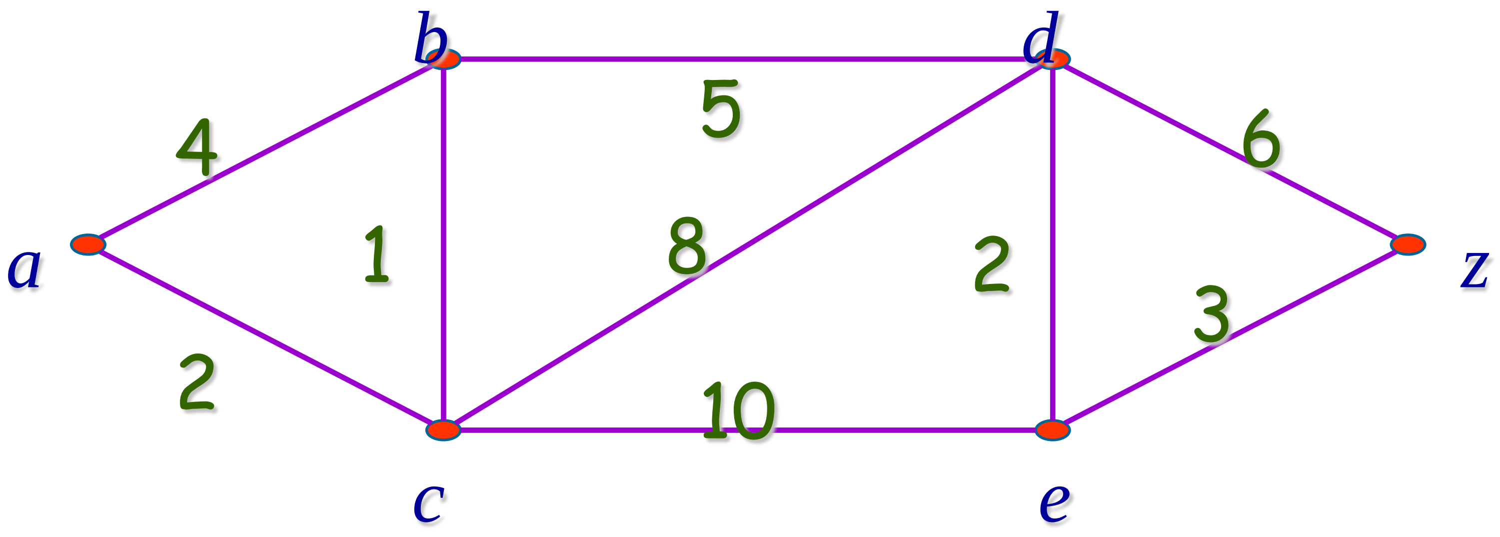
Step 6



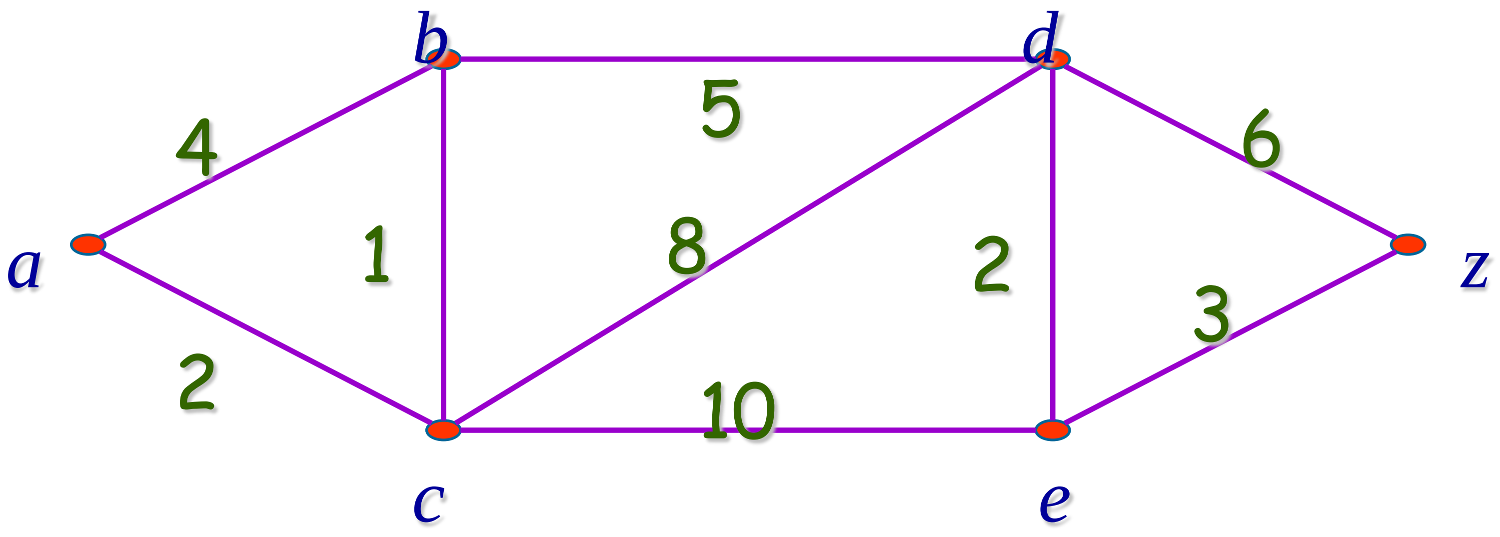
In performing Dijkstra's algorithm it is sometimes more convenient to keep track of labels of vertices using a table instead of redrawing the graph .



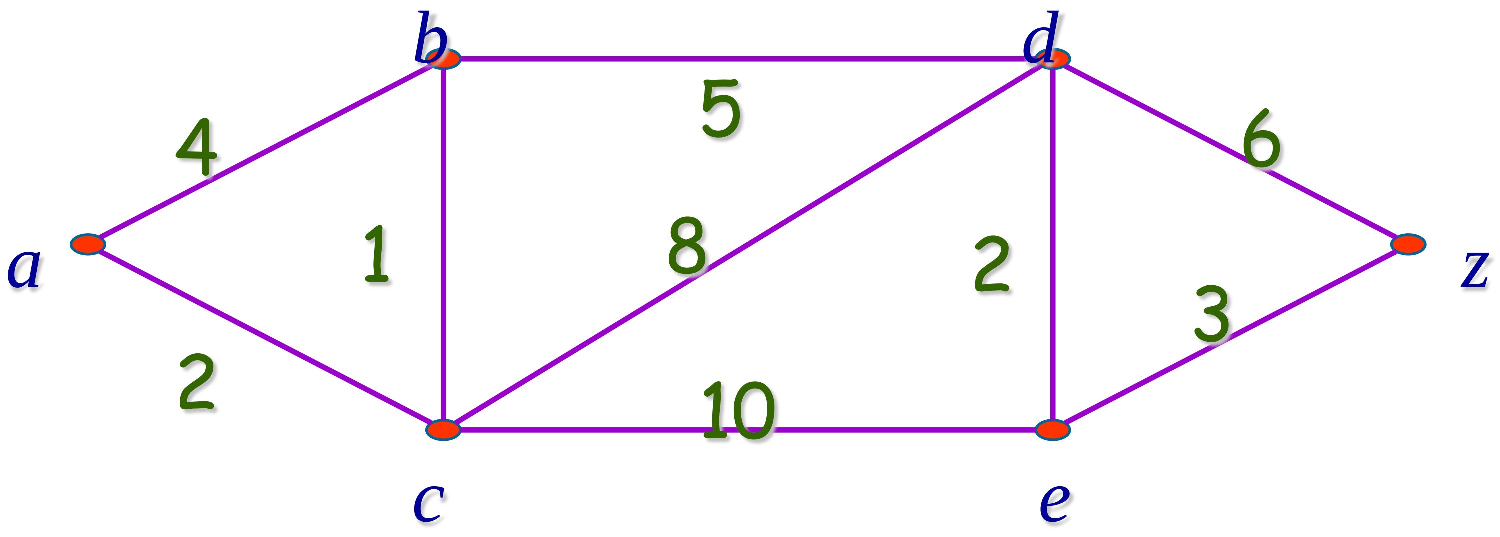
Vertex	S	Link						
a								
b								
c								
d								
e								
z								



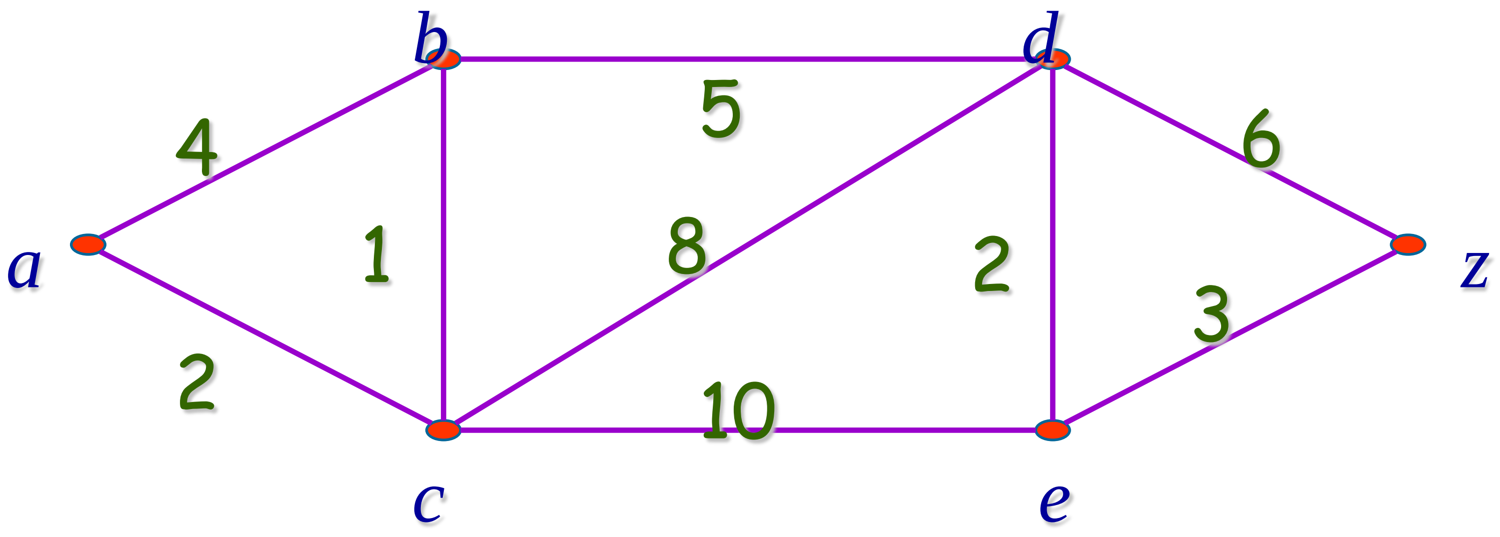
Vertex	S	Link	L_0					
a			0					
b			∞					
c			∞					
d			∞					
e			∞					
z			∞					



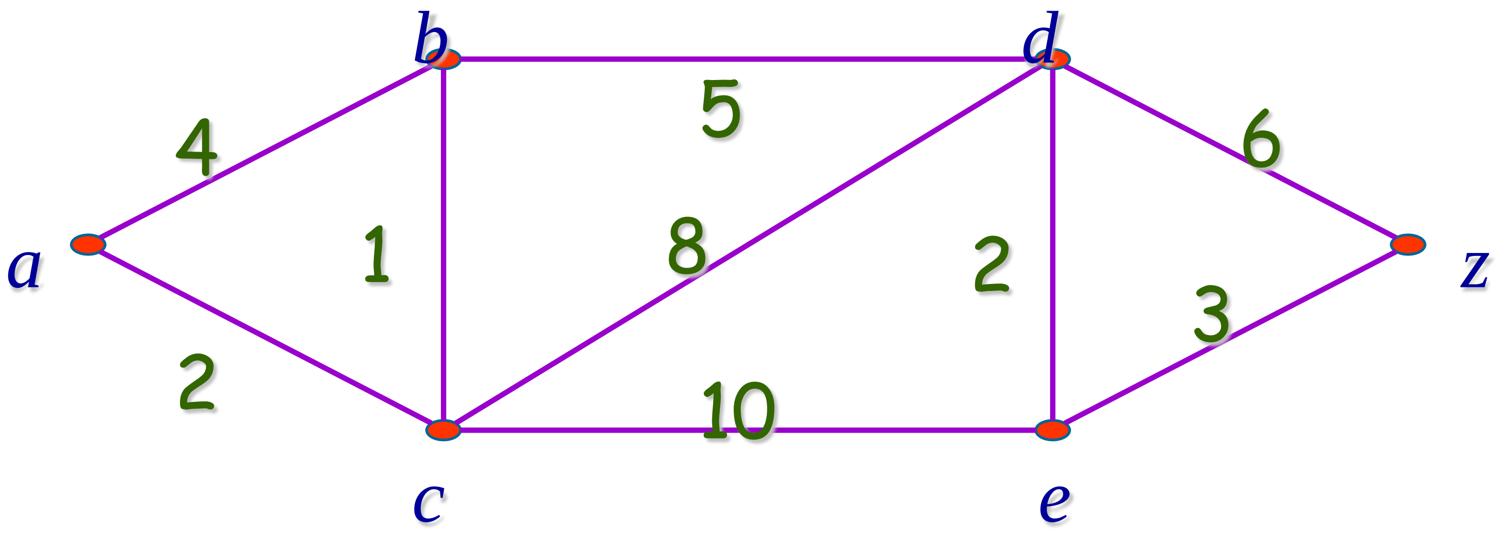
Vertex	S	Link	L_0	L_1				
a	1		0					
b		***a***	∞	4				
c		***a***	∞	2				
d			∞	∞				
e			∞	∞				
z			∞	∞				



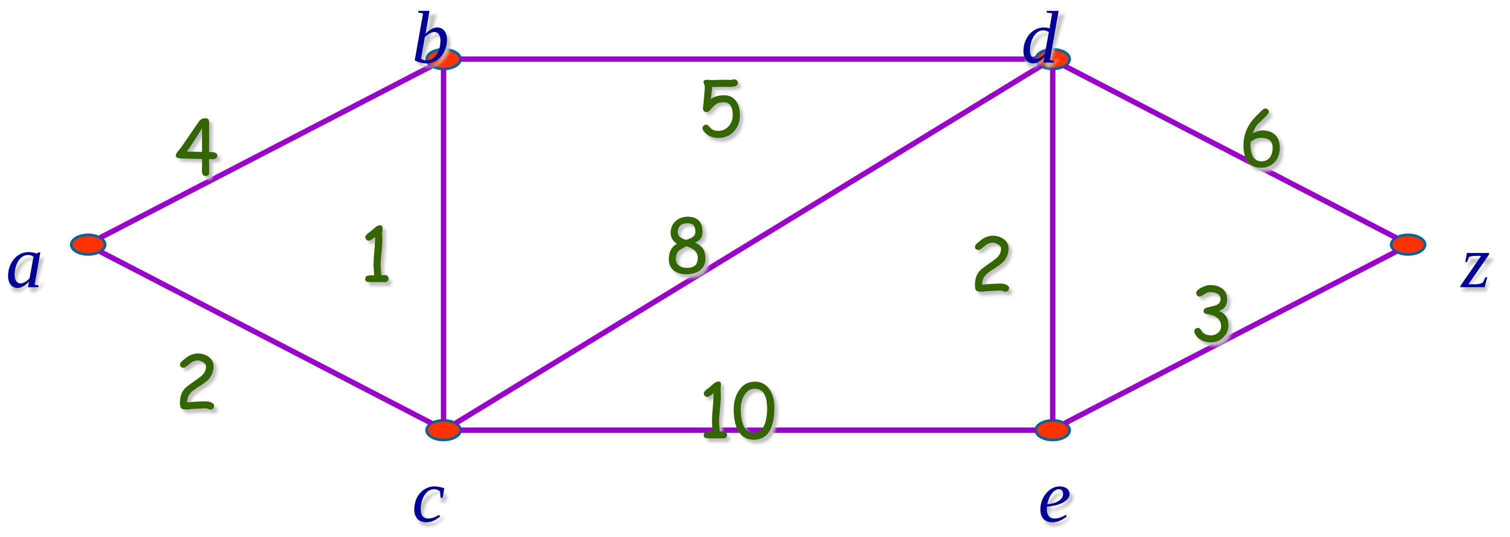
Vertex	S	Link	L_0	L_1	L_2			
a	1		0					
b		*a*→*c*	∞	4	3			
c	1	*a*	∞	2				
d		*a*→*c*	∞	∞	10			
e		*a*→*c*	∞	∞	12			
z			∞	∞	∞			



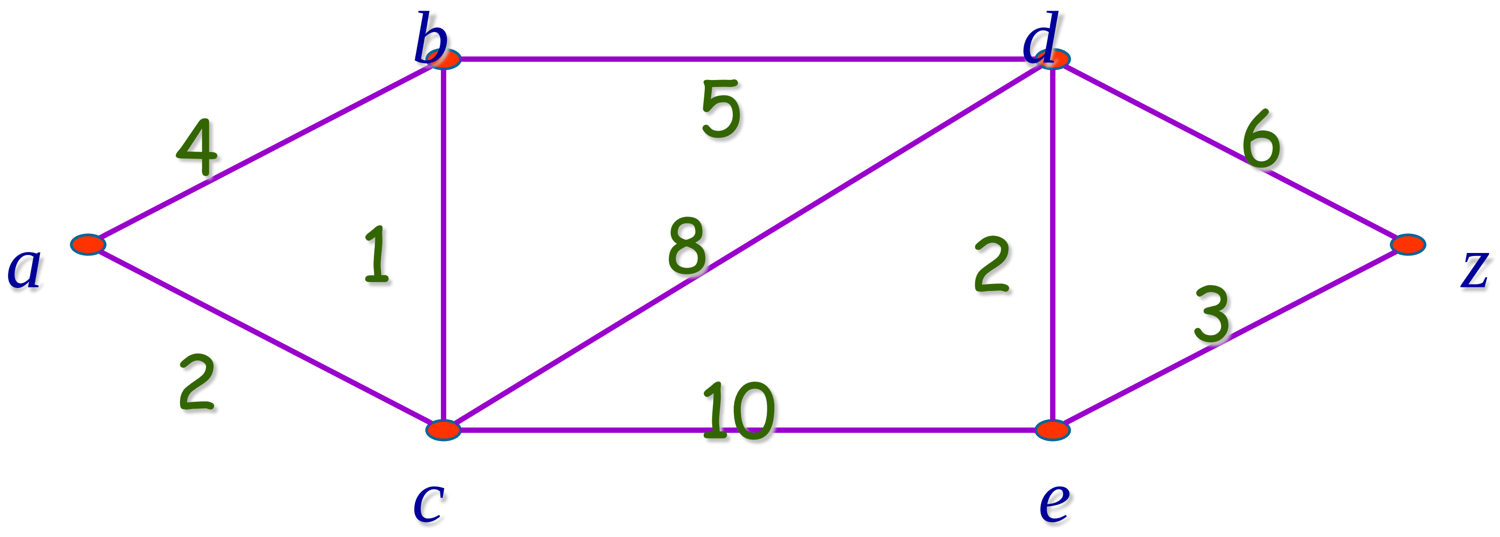
Vertex	S	Link	L_0	L_1	L_2	L_3		
a	1		0					
b	1	*a*→*c*	∞	4	3			
c	1	*a*	∞	2				
d		*a*→*c*→*b*	∞	∞	10	8		
e		*a*→*c*	∞	∞	12	12		
z			∞	∞	∞	∞		



Vertex	S	Link	L_0	L_1	L_2	L_3	L_4	
a	1		0					
b	1	*a→c*	∞	4	3			
c	1	*a*	∞	2				
d	1	*a→c→b*	∞	∞	10	8		
e		*a→c→b→d*	∞	∞	12	12	10	
z		*a→c→b→d*	∞	∞	∞	∞	14	

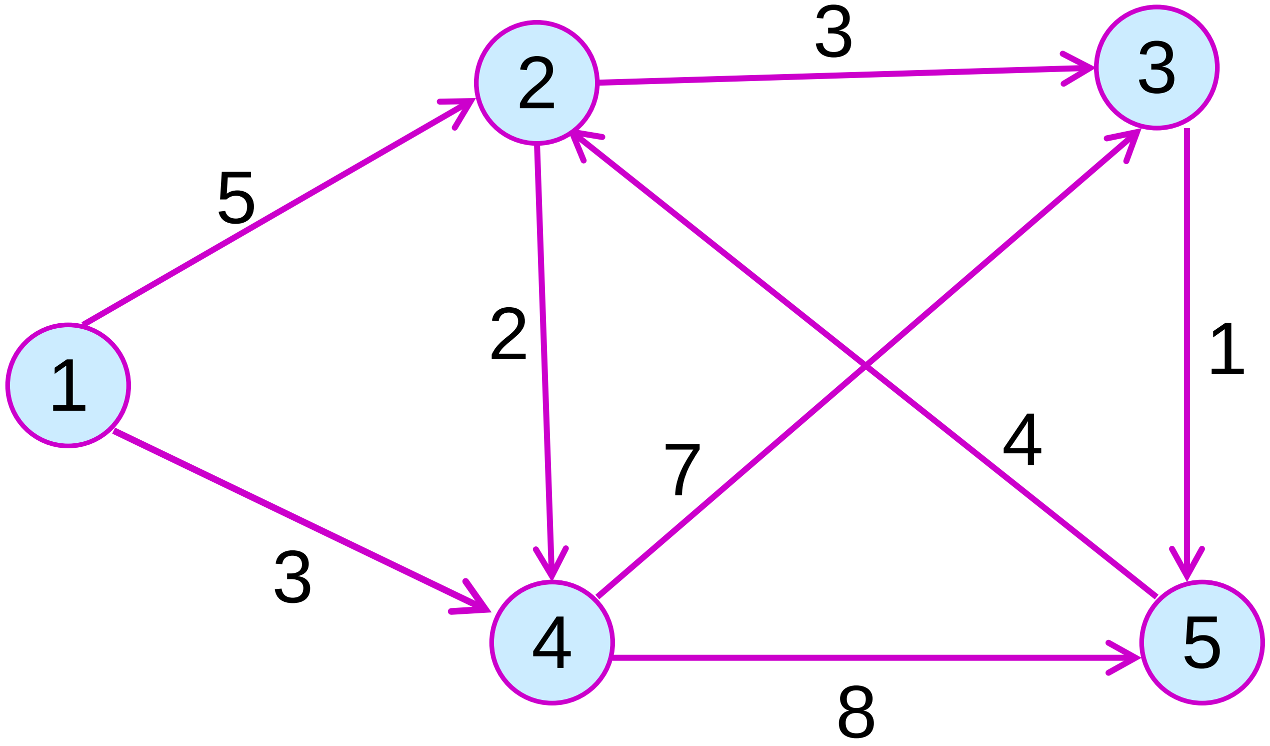


Vertex	S	Link	L_0	L_1	L_2	L_3	L_4	L_5
a	1		0					
b	1	*a→c*	∞	4	3			
c	1	*a*	∞	2				
d	1	*a→c→b*	∞	∞	10	8		
e	1	*a→c→b→d*	∞	∞	12	12	10	
z		*a→c→b→d → e*	∞	∞	∞	∞	14	13



Vertex	S	Link	L_0	L_1	L_2	L_3	L_4	L_5
a	1		0					
b	1	*a*→*c*	∞	4	3			
c	1	*a*	∞	2				
d	1	*a*→*c*→*b*	∞	∞	10	8		
e	1	*a*→*c*→*b*→*d*	∞	∞	12	12	10	
z	1	*a*→*c*→*b*→*d* → *e*	∞	∞	∞	∞	14	13

Dijkstra's Algorithm applies to a directed graph.



Some Questions

1. How to extend Dijkstra's algorithm to find the length of a shortest path between the vertex a and every other vertex of the gragh?
2. How to extend Dijkstra's algorithm to constructed a shortest path between these two vertices?
3. How to find the length of a shortest path between all pairs of vertices in a weighted connected simple graph?

The Correctness of Dijkstra's Algorithm

【Theorem 1】 Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.

Proof:

We use an inductive argument. Take as the induction hypothesis the following assertion: At the *k*th iteration

I. the label of every vertex v in S is the length of the shortest path from a to this vertex, and

II. the label of every vertex not in S is the length of the shortest path from a to this vertex that contains only vertices in S.

(1) k=0

$L_0(a)=0, L_0(v)=\infty, S=\phi$

The Correctness of Dijkstra's Algorithm

(2) Assume that the inductive hypothesis holds for the kth iteration.

Let v be the vertex added to S at the (k+1)st iteration so that v is a vertex not in S at the end of the kth iteration with the smallest label.

- **(I) holds at the end of the (k+1)st iteration**
  - ✓ **The vertices in S before the (k+1)st iteration are labeled with the length of the shortest path from a.**
  - ✓ **v must be labeled with the length of the shortest path to it from a.**

    If this were not the case, at the end of the kth iteration there would be a path of length less than $L_k(v)$ containing a vertex not in S.

    Let u be the first vertex not in S in such a path. There is a path with length less than $L_k(v)$ from a to u containing only vertices of S. This contradicts the choice of v.

The Correctness of Dijkstra's Algorithm

- **(II) is true.**

Let u be a vertex not in S after k+1 iteration.

A shortest path from a to u containing only elements of S either contains v or it does not.

- **If it does not contain v, then by the inductive hypothesis its length is $L_k(u)$.**
- **If it does contain v, then it must be made up of a path from a to v of the shortest possible length containing elements of S other than v, followed by the edge from v to u. In this case its length would be $L_k(v)+w(v,u)$.**

This shows that (II) is true, because $L_{k+1}(v)=\min\{L_k(v), L_k(u)+w(u,v)\}$

The Computational Complexity of Dijkstra's Algorithm

【 Theorem 2】 Dijkstra's algorithm uses $O(n^2)$ operations (additions and comparisons) to find the length of the shortest path between two vertices in a connected simple undirected weighted graph.

Analysis:

- Use no more than n-1 iteration
- Each iteration,

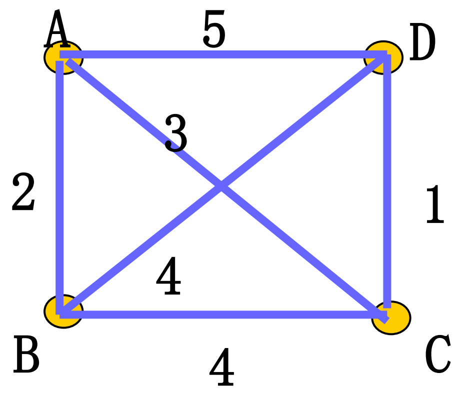
  using no more than n-1 comparisons to determine the vertex not in S_k with the smallest label

  no more than 2(n-1) operations are used to update no more than n-1 labels

The Traveling Salesperson Problem (TSP)

- **Problem:** A traveling salesperson wants to visit each of *n* cities exactly once and return to his starting point with minimum total …
- **The graph model:** weighted,complete,undirected graph
- **The equivalent problem for TSP:** Find a Hamilton circuit with minimum total weight in the weighted complete undirected graph.

An example



The shortest H circuit :
(A,B,D,C,A)， length is10

The Traveling Salesperson Problem (TSP)

- **Solving TSP**
- The most straightforward one:
  - Examine all possible Hamilton circuits and select one of minimum total length.

How many are there different length of Hamilton circuits in a complete graph with *n* vertices?

$$(n-1)!/2$$

Note:

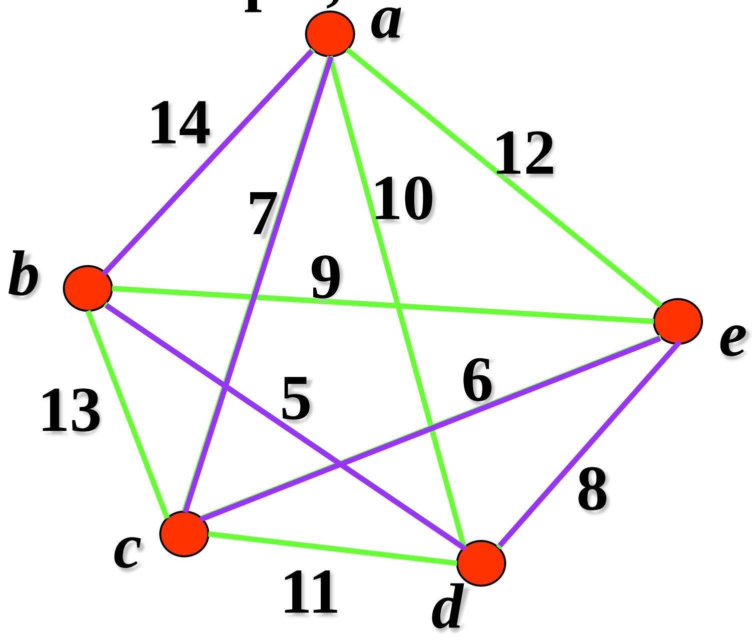
$(n-1)!/2$ **grows extremely rapidly.**

For example, with 25 vertices, $24!/2 \approx 3.1\times 10^{23}$

The Traveling Salesperson Problem (TSP)

- **Approximation algorithm**
  - do not necessary produce the exact solution
  - to produce a solution that is close to an exact solution

For example,



The length of this path: 40

The exact solution: 37

(*a*,*c*,*e*,*b*,*d*,*a*)

The time complexity: $1+2+3+\ldots+(n-2)=\frac{1}{2}(n-1)(n-2)$

Compare with *d* and *d*0: $\frac{d}{d_0} \le \frac{1}{2}\lceil \log_2 n \rceil + \frac{1}{2}$

The Traveling Salesperson Problem (TSP)

✵ More about TSP

TSP has both pratical and theoretical importance.

Website for TSP: http://www.tsp.gatech.edu/

Homework:

SE: P. 716 3, 5(3), 16, 17, 26

EE: P. 751 3, 5(3), 16, 17, 26