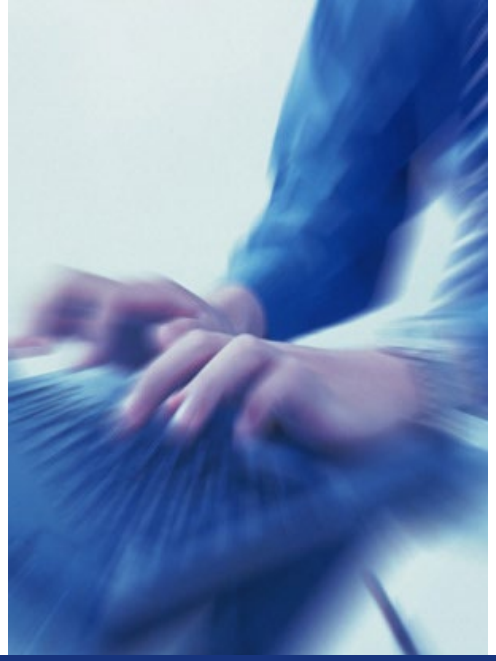
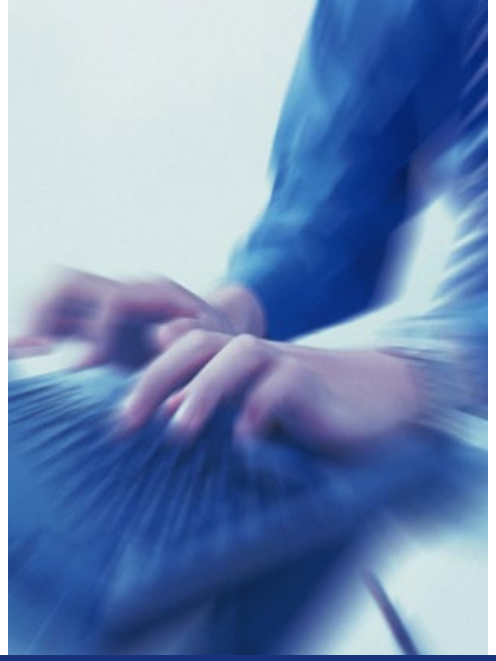


Logo

Discrete Mathematics

Dr. Han Huang

South China University of Technology




Logo

Chapter 4. Graphs

Shortest-Path Problems

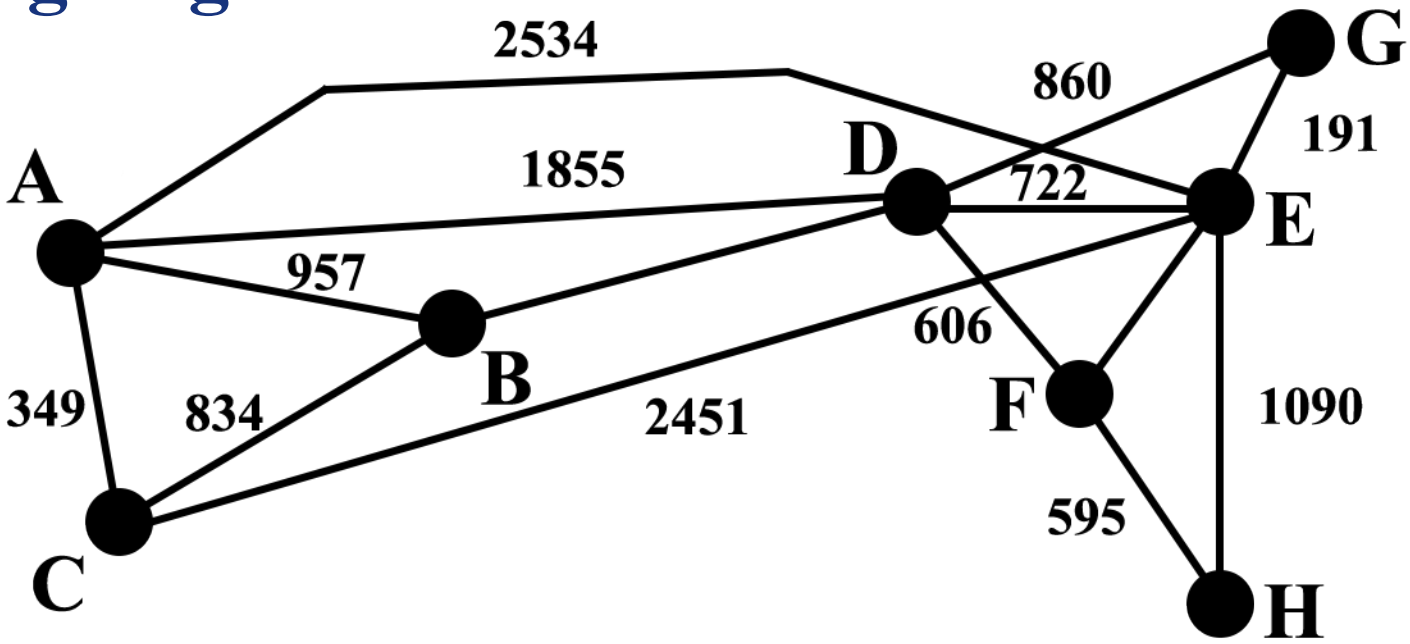
Section 4.6

Contents

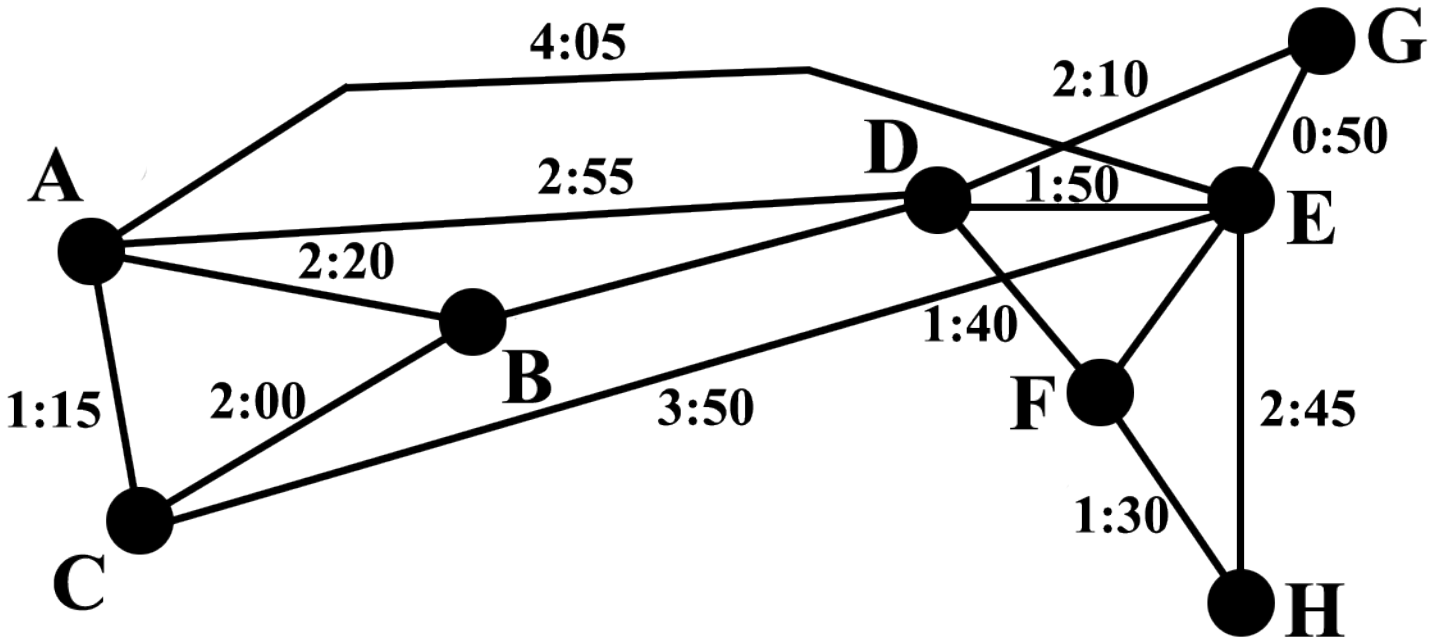

Shortest Path
Problems

Introduction

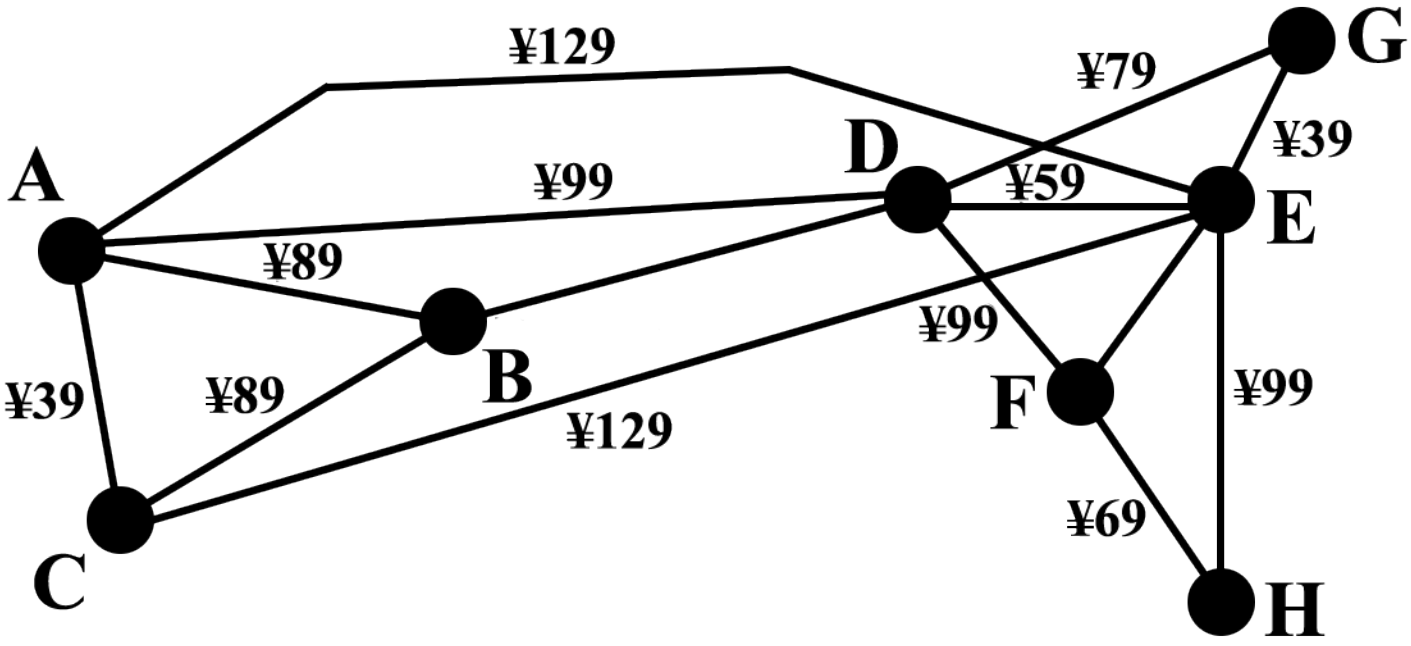
- Many problems can be modeled using graphs with weights assigned to their edges.
- Problems involving distances can be modeled by assigning distances between cities to the edges.

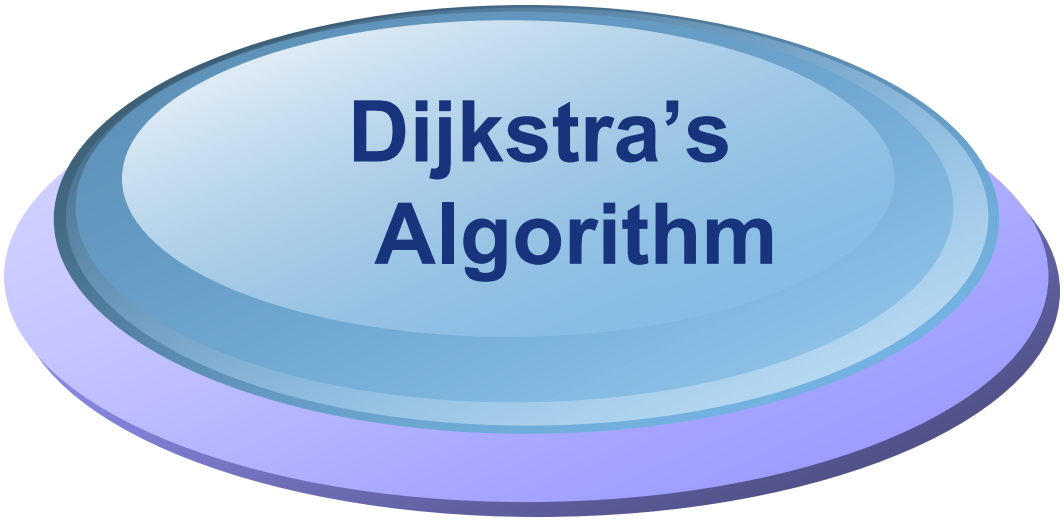


- Problems involving flight time can be modeled by assigning flight time to the edges.



- Problems involving fares can be modeled by assigning fares to the edges.

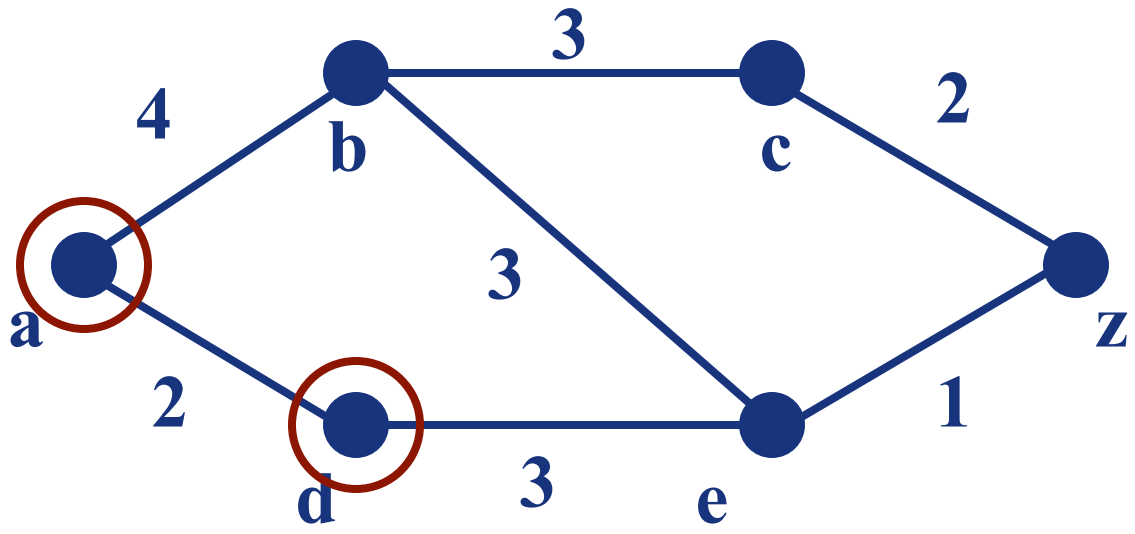



Dijkstra's
Algorithm

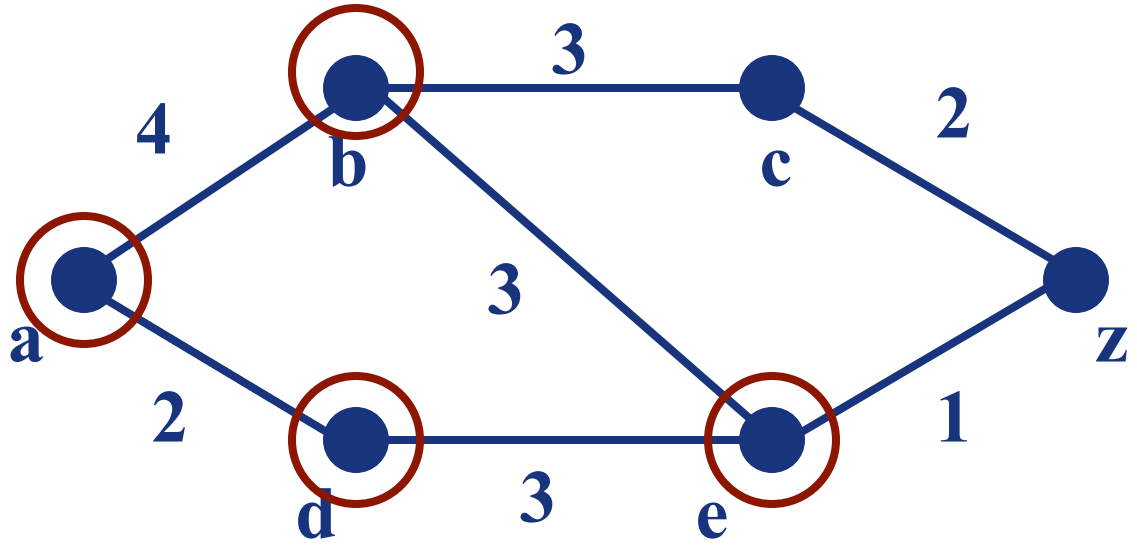
A Shortest-Path Algorithm

- There are several different algorithms that find a shortest path between two vertices in a weighted graph.

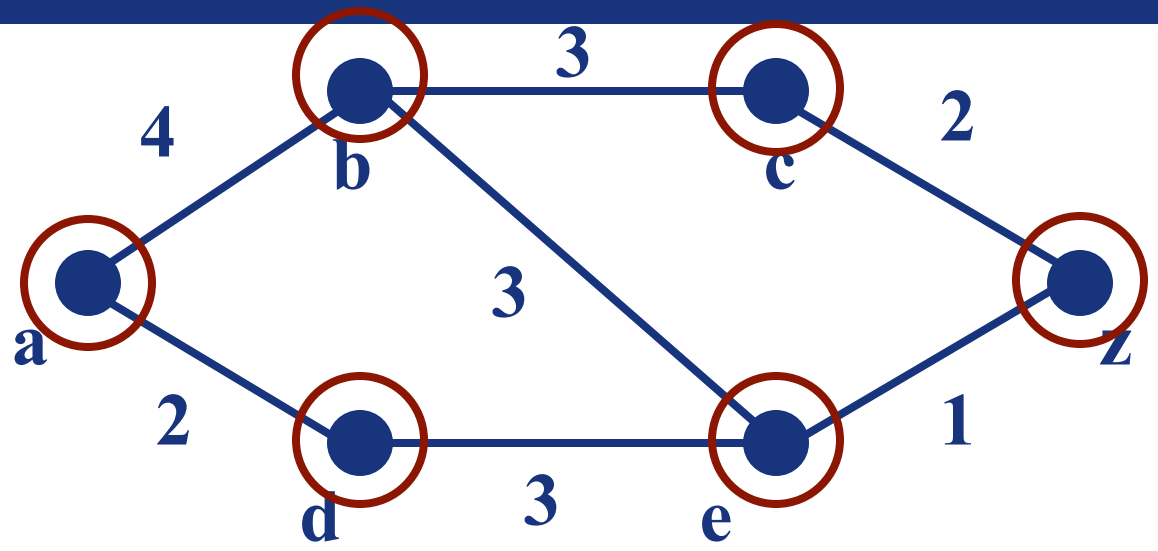
- This section will present the classical algorithm: Dijkstra's Algorithm proposed by Dutch mathematician Edsger Dijkstra.



- The only paths starting at a that contain no vertex other than a (until the terminal vertex is reached) are a, b and a, d.
- Since the lengths of a, b and a, d are 4 and 2, d is the closet vertex to a.



- We can find the next closet vertex by looking at all paths that go through only a and d.
- The shortest such path to b is still a, b, with length 4, and the shortest such path to e is a, d, e, with length 5. So the next closet vertex to a is b.



- For the next closet vertex to a, we need to examine only paths that go through only a, d, and b (until the terminal vertex is reached).
- There is a path of length 7 to c, namely a, b, c and a path of length 6 to z, namely, a, d, e, z. So z is the next closet vertex to a, and its length is 6.

The idea of Dijkstra's Algorithm

- The algorithm begins by labeling a with 0 and the other vertices with ∞ .
- Note that $L_0(a) = 0$ and $L_0(v) = \infty$ for these labels before any iterations have taken place.
- These labels are the lengths of shortest paths from a to the vertices, where the paths contain only the vertex a.

The idea of Dijkstra's Algorithm

- Dijkstra's algorithm proceeds by forming a distinguished set of vertices.
- Let S_k denote this set after k iterations of labeling procedure.
- $S_0 = \emptyset$, S_k is formed from S_{k-1} by adding a vertex u not in S_{k-1} with the smallest label.

The idea of Dijkstra's Algorithm

- Once u is added S_k, we 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 a shortest path from a to v that contains vertices only in S_k.
- Let v be a vertex not in S_k.
- To update the label of v, note that $L_k(v)$ is the length of a shortest path from a to v containing only vertices in S_k.

The idea of Dijkstra's Algorithm

- The observation is used:
- A shortest path from a to v containing only elements of S_k is either a shortest path from a to v that contains only elements of S_{k-1}.
- Or it is a shortest path from a to u at the $(k-1)st$ stage with the edge (u, v)added.

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

The idea of Dijkstra's Algorithm

- This procedure is iterated by successively adding vertices to the distinguished set until z is added.
- When z is added to the distinguished set, its label is the length of a shortest path from a to z.

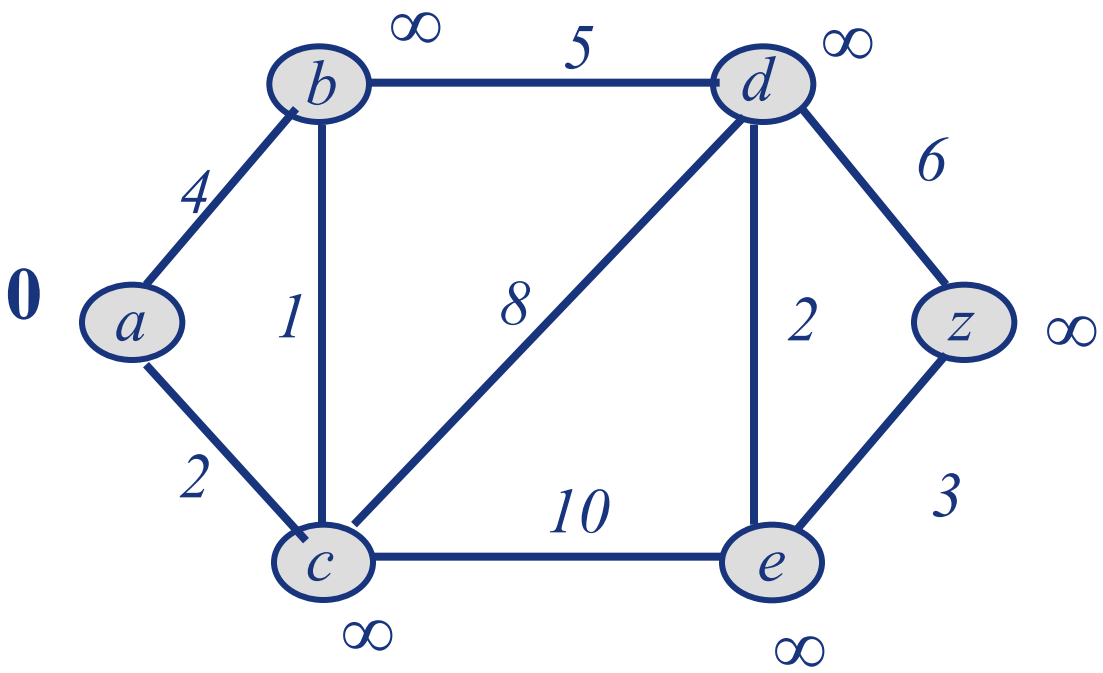
Algorithm 1: Dijkstra's Algorithm

- Procedure Dijkstra (G: weighted connected simple graph, with all weights positive)
- {G has vertices $a = v_0, v_1, \ldots, v_n = z$ and weight $w(v_i, v_j)$ 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 initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}

Algorithm 1: Dijkstra's Algorithm

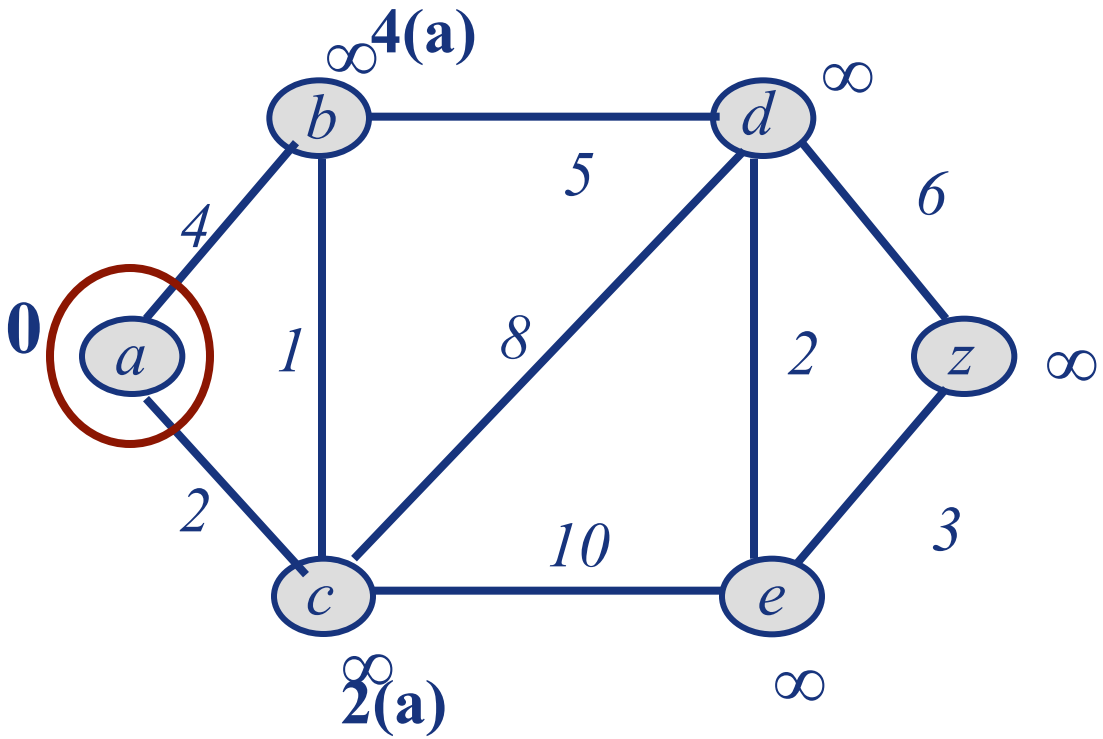
- while z not at S
- begin
- u:=a vertex not in S with L(u) minimal
- S:= S U {u}
- for all vertices v not in S
- if L(u)+w(u,v) < L(v) then
- L(v):=L(u)+w(u,v)
- {this adds a vertex to S with minimal label and update the labels of vertices not in S}
- end {L(z)=length of a shortest path from a to z}

Example 2



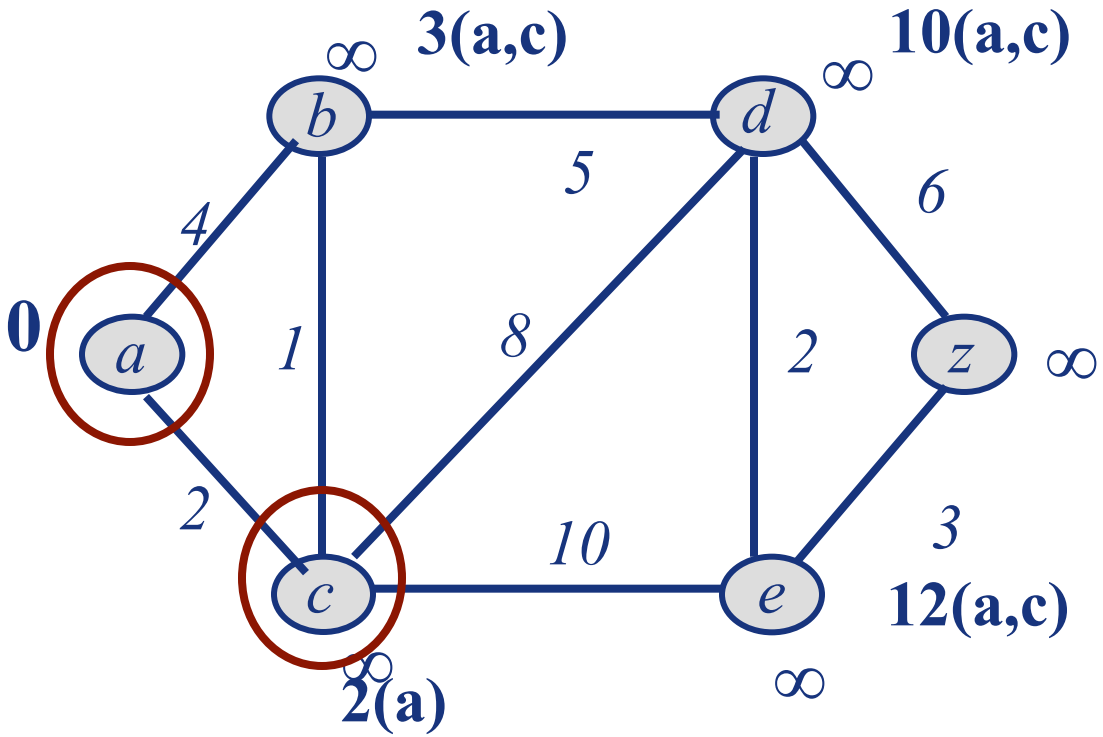
- for i:=1 to n
- L(vi):=∞
- L(a):=0
- S:=Ø

Example 2



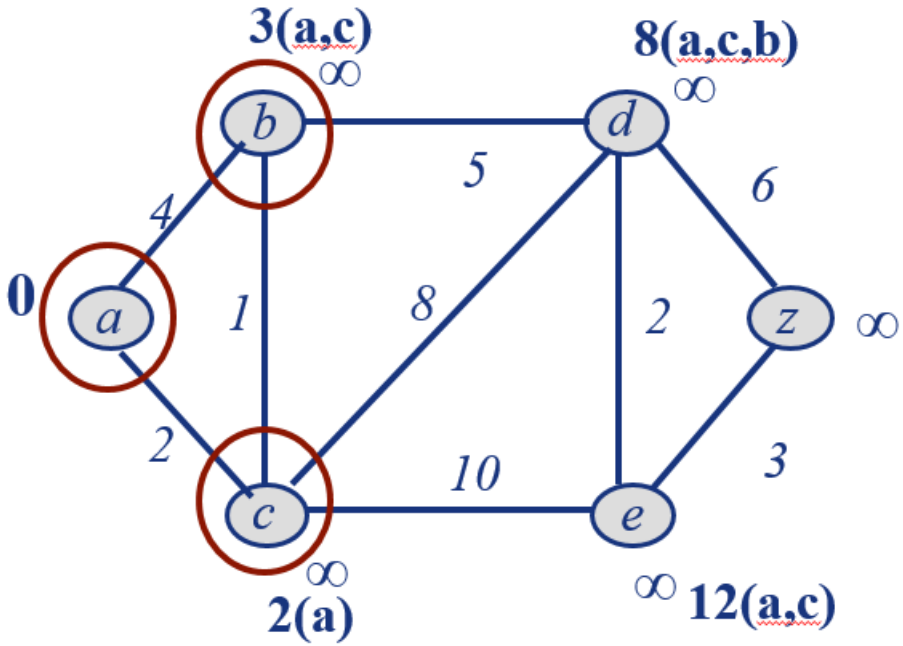
- u:=a vertex not in S with L(u) minimal
- S:= S U {u}

Example 2



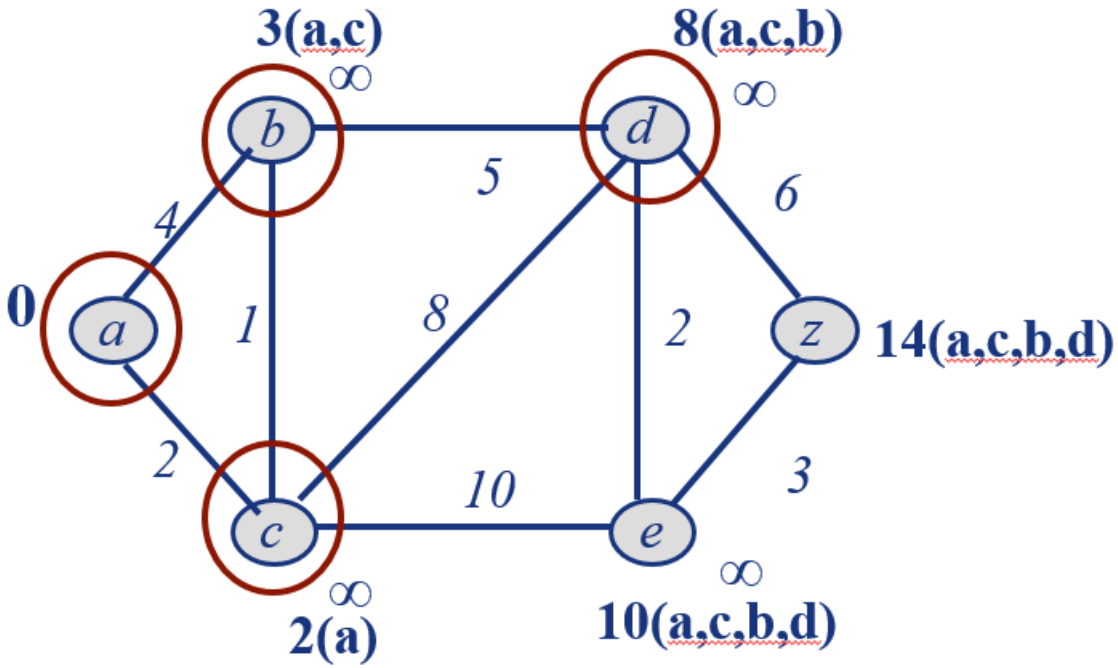
- if L(u)+w(u,v) < L(v) then
- L(v):=L(u)+w(u,v)

Example 2



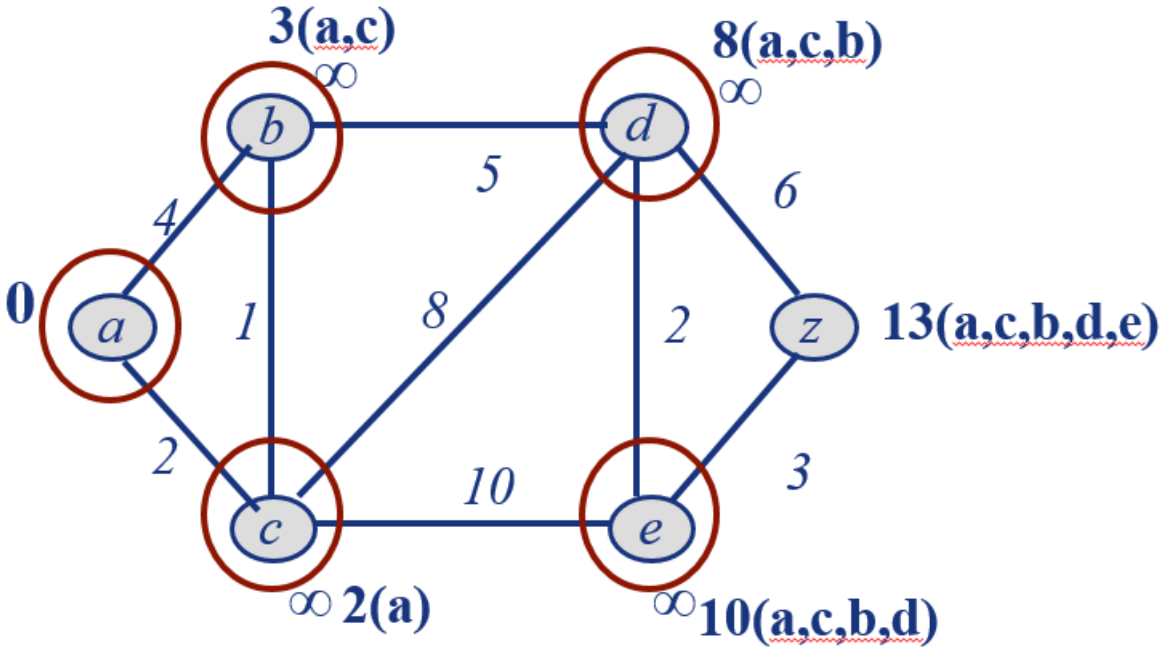
- if $L(u) + w(u, v) < L(v)$ then
- $L(v) := L(u) + w(u, v)$

Example 2



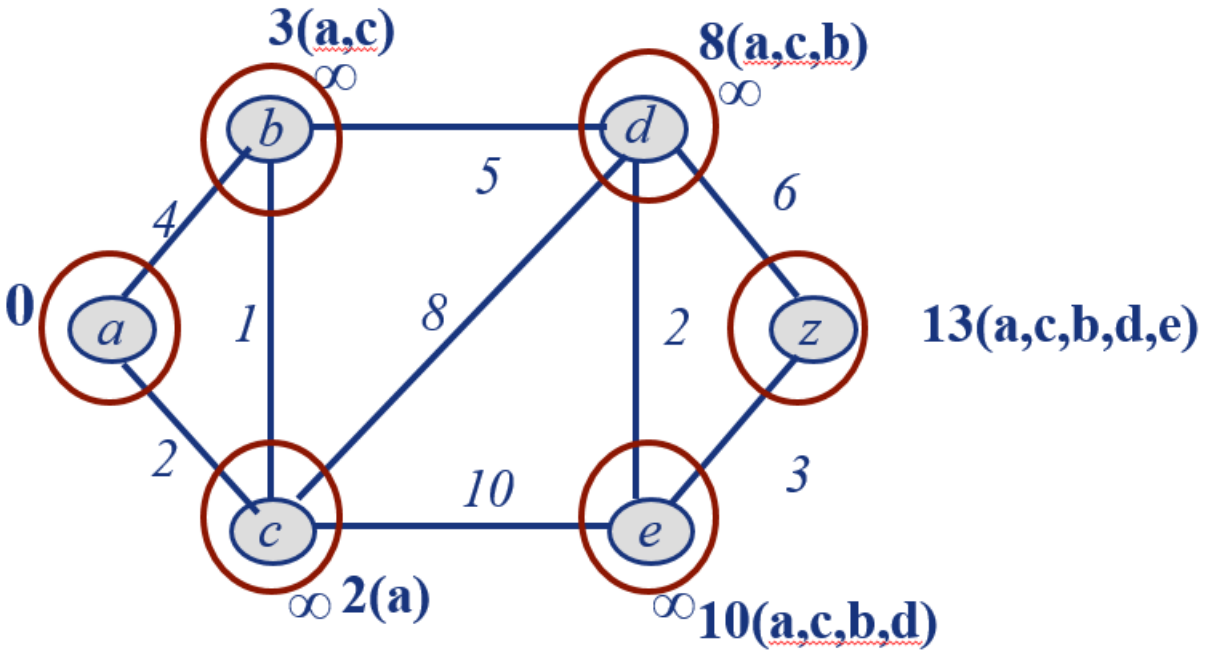
- if $L(u) + w(u, v) < L(v)$ then
- $L(v) := L(u) + w(u, v)$

Example 2



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

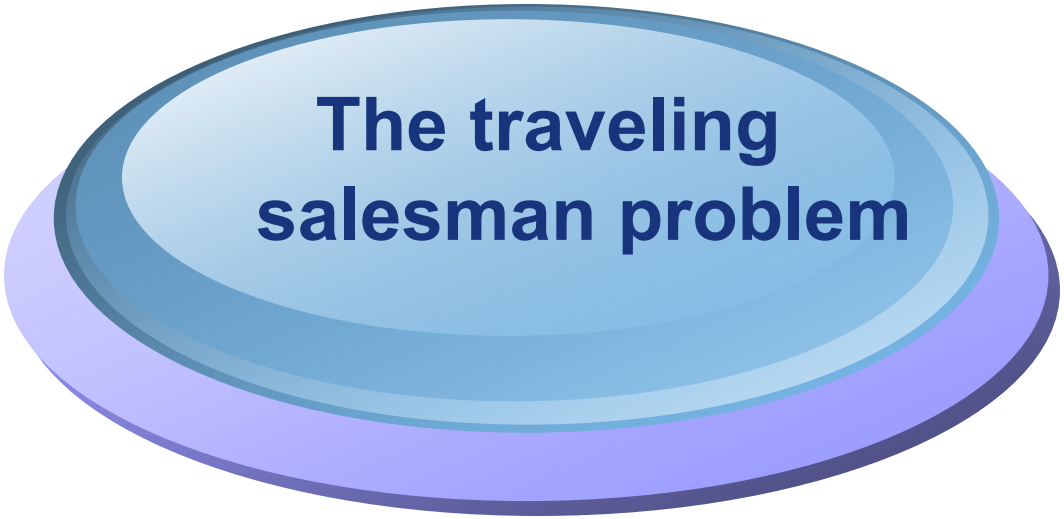
Example 2



- The shortest path is a, c, b, d, e, z with length 13.

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

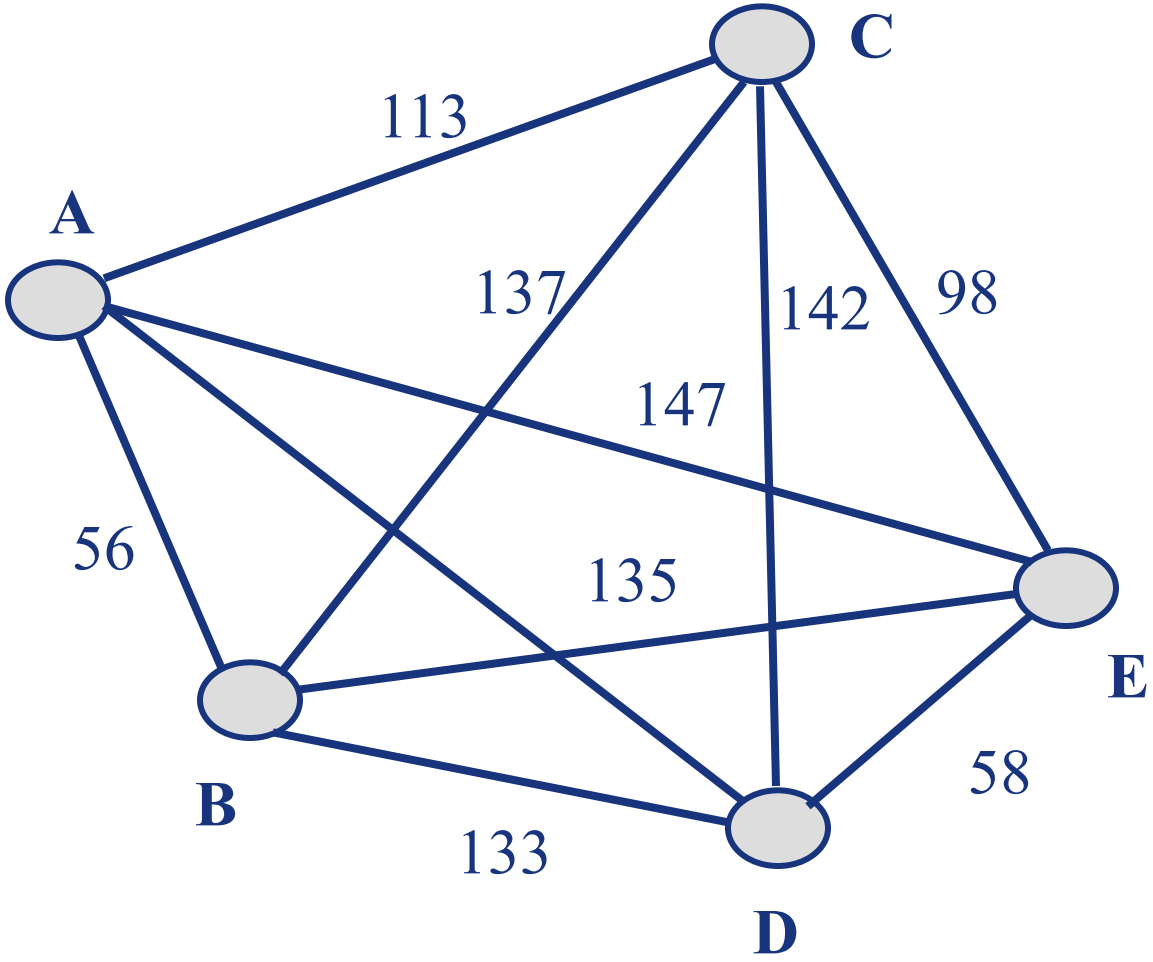
- **Theorem 2: Dijstr's algorithm uses O(n^2) operation (additions and comparisons) to find the length of shortest path between two vertices in a connected simple undirected weighted graph with n vertices.**


The traveling
salesman problem

Traveling salesman problem

- TSP Problem: A traveling salesman wants to visit each of n cities exactly once and return to his starting point.
- In which order should he visit these cities to travel the minimum total distance.

- For a n cities TSP problem, there are (n-1)!/2 different Hamilton circuits to examine.



- 5 vertices
- 5 cities
- 4!/2=12

Hamilton circuits

- For a TSP of 25 cities, the problem will have 25 vertices, and a total of 24!/2 different Hamilton circuits.

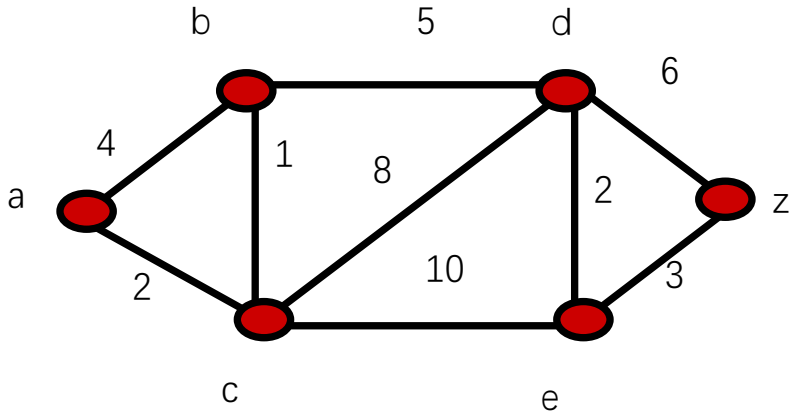
$$24!/2 \approx 3.1 \times 10^{23}$$

- If it took just one nanosecond (10^{-9} second) to examine each Hamilton circuit, the required time would be approximately ten million years.

- Since the traveling salesman problem has both practical and theoretical importance, a great deal of effort has been devoted to devising efficient algorithms that solve it.
- However, no algorithm with polynomial worst-case time complexity is known for solving this problem.
- Furthermore, if a polynomial worst-case time complexity algorithm were discovered for TSP, many other difficult problems would be also solved.

Exercises

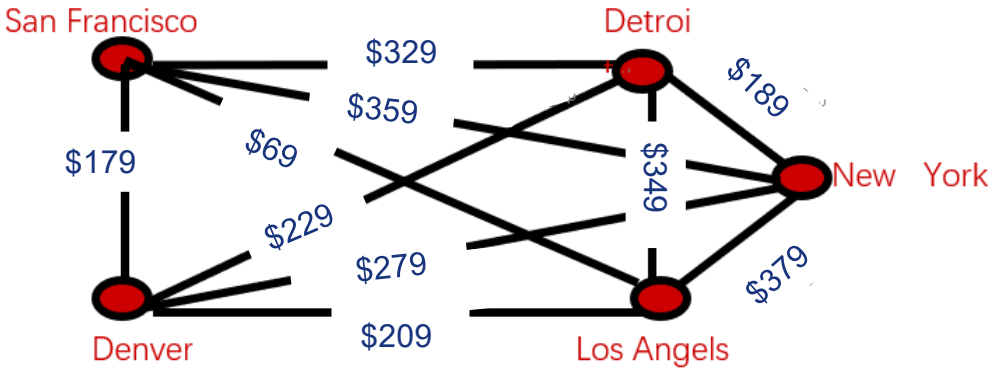
- 1. Use Dijkstra's algorithm to find the length of the shortest path between the vertices z and a in the weighted graph.



z-e: 3
z-e-d:5
z-e-d-b:10
z-e-d-b-c:11
z-e-d-b-c-a:13
so the shortest path: *zedbca*

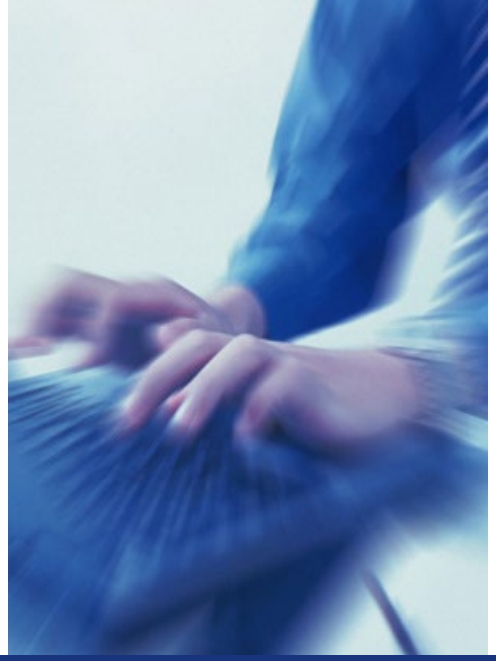
Exercises

- 2. **Find a route with the least total airfare that visits each of the cities in the graph where the weight on an edge is the least price available for a flight between the two cities.（ San Francisco- $69- Los Angels, San Francisco- $359-New York , Denver -$229-Detroi, Detroi-$349- Los Angels ）**



San Francisco- Denver -Detroi-New York –Los Angels –San Francisco

Logo



End of Section 4.6