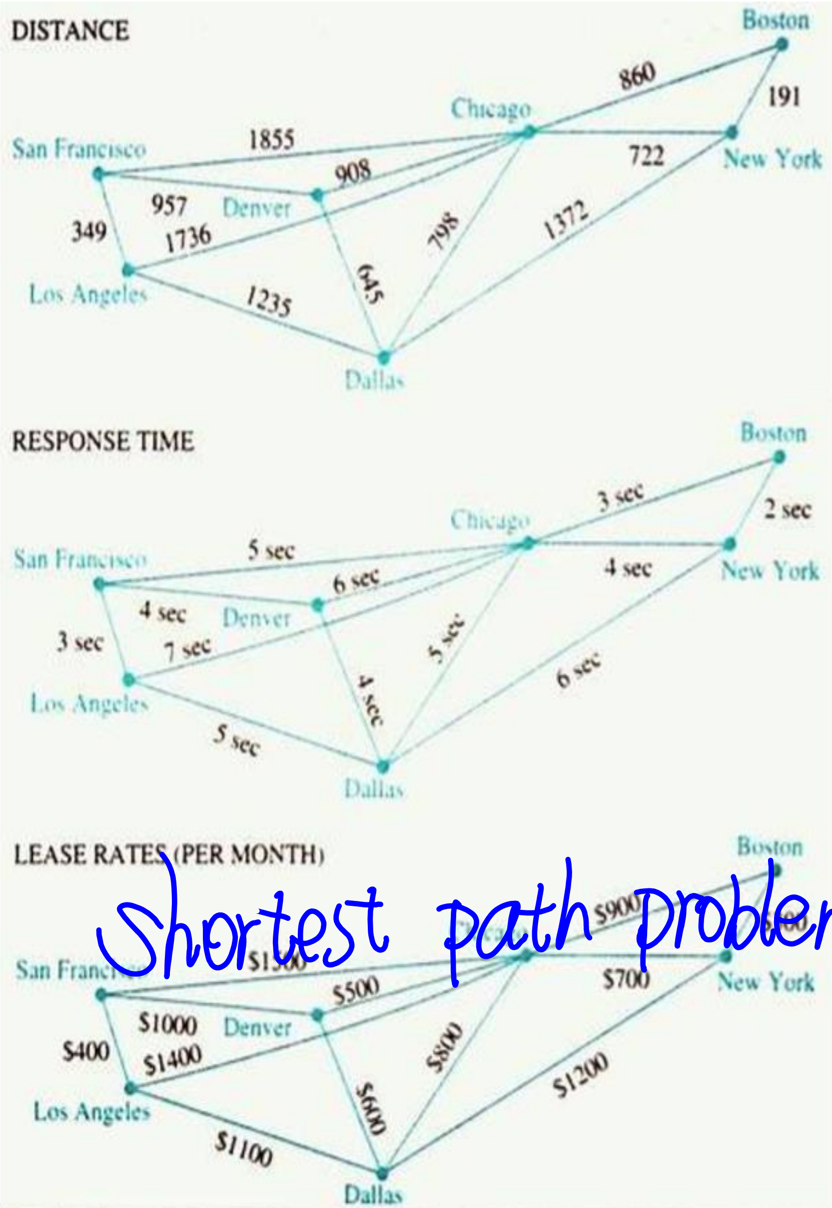
Shortest Path Problem（最短路径问题）

Zhang Yanmei

ymzhang@bupt.edu.cn

College of Computer Science & Technology

Beijing University of Posts & Telecommunications



Shortest path problem

Weighted Graphs Modeling a Computer Network.

Weighted Graphs Modeling

- Airline system
  - Mileage
  - Flight times
  - Fares
- Computer network
  - Distance
  - Response time
  - Lease rates

shortest path Problem

- There are two types of such problems
  - Determining the shortest path from a vertex to an assigned vertex.
  - Determining the shortest path of any two vertices in the graph.

Dijkstra Algorithm

- The algorithm is to find the shortest way from v_1 to v_n,at the same time,it gets the shortest way from v_1 to each other vertices in the graph.

Example1

- What is the length of a shortest path between *a* and *z* in the weighted graph shown in Figure 3?

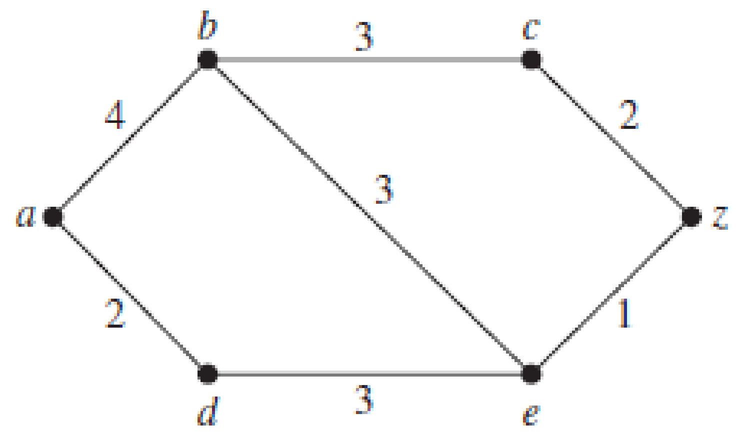


FIGURE 3 A Weighted Simple Graph.

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 lengths $w(v_i, v_j)$
where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
$\quad L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
other labels are ∞, and S is the empty set}
while $z \notin S$
$\quad u :=$ a vertex not in S with $L(u)$ minimal
$\quad S := S \cup \{u\}$
$\quad$**for** all vertices v not in S
$\quad\quad$**if** $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
$\quad\quad${this adds a vertex to S with minimal label and updates the
$\quad\quad$labels of vertices not in S}
return $L(z)$ {$L(z) =$ length of a shortest path from a to z}

Theroem

- Dijkstra’s algorithm finds the length of a shortest path between two vertices in a <u>connected simple undirected weighted</u> graph.
- Proof:
  - (i) the label of every vertex *v* in S is the length of a shortest path from *a* to this vertex,
  - (ii) the label of every vertex not in S is the length of a shortest path from *a* to this vertex that contains only (besides the vertex itself) vertices in S.

proof

- Basic step: $k = 0$, $S = \emptyset$, length is ∞.
- Inductive step: holds for the kth iteration, Let v be the vertex added to S at the (k+1)st iteration.
  - $L_{k+1}(a,v) = \min \{L_{k+1}(a,x), x \text{ not in } S\}$

  $= \min\{ \min \{ L_k(a,x), L_k(a,y)+w(y,x)\}, x \text{ and } y \text{ not in } S \}$

$L_k(a,x)$ and $L_k(a,y)$ are shortest length from a to x/y only contains vertices in S.

  - if $L_{k+1}(a,v)$ *isn't shortest path, let* $L'(a,u,v) < L_{k+1}(a,v)$, *u is first vertex not in S.*
  - *Then must have* $L_k(a,u)<L_k(a,v)$, *but* $L_{k+1}(a,v)= \min \{ L_k(a,v), L_k(a,u)+w(u,v)\}$ *and* $w(u,v)$, *This contradiction.*

Base idea

- Based on the fact:

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

- It functions by constructing a shortest-path tree from the initial vertex to every other vertex in the graph.

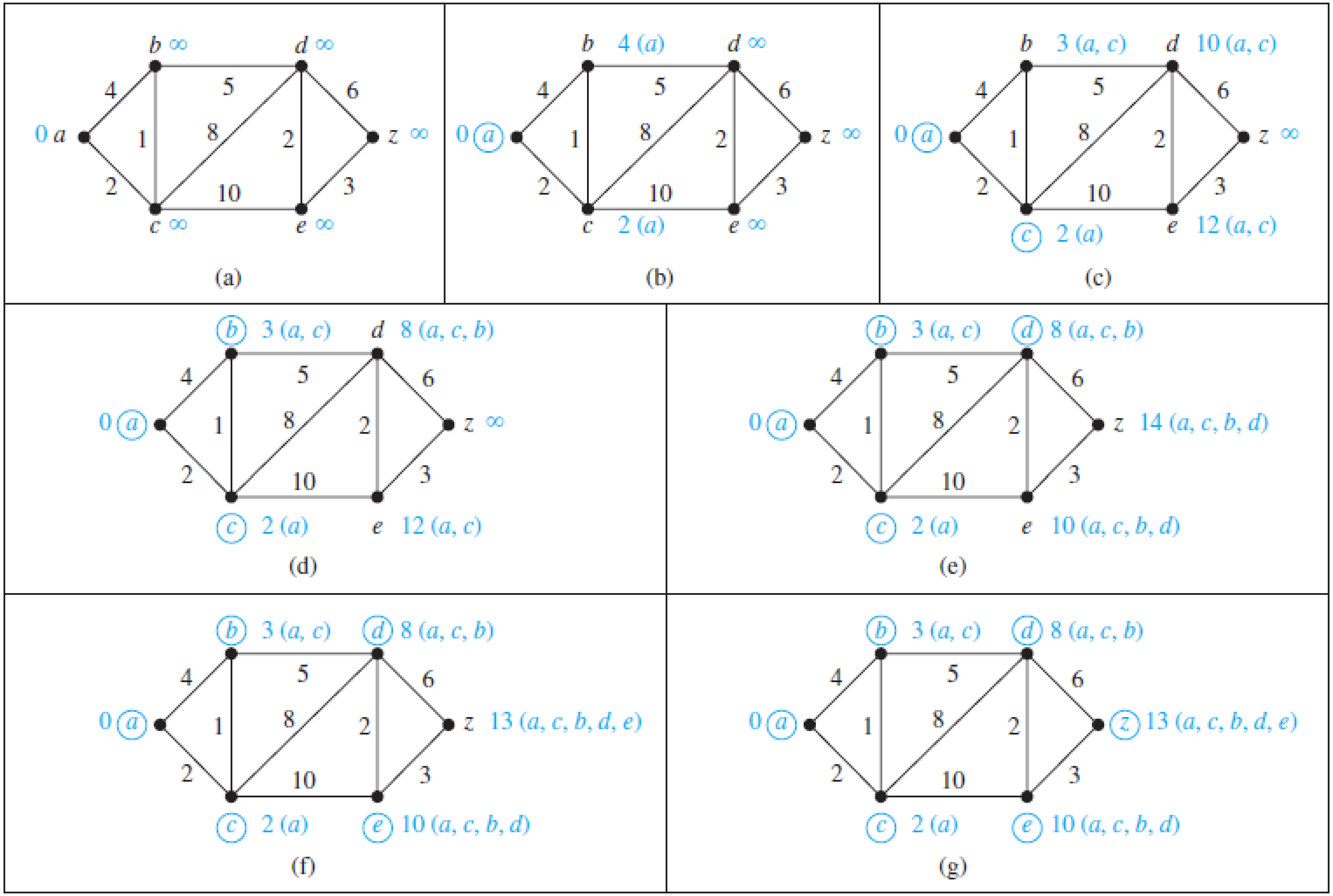
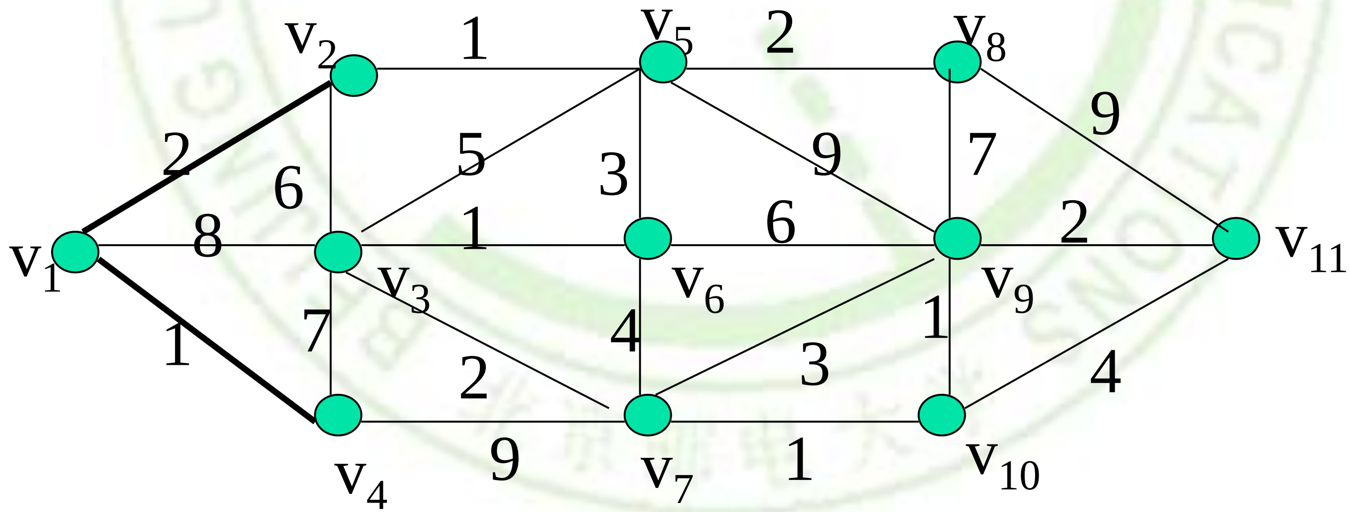


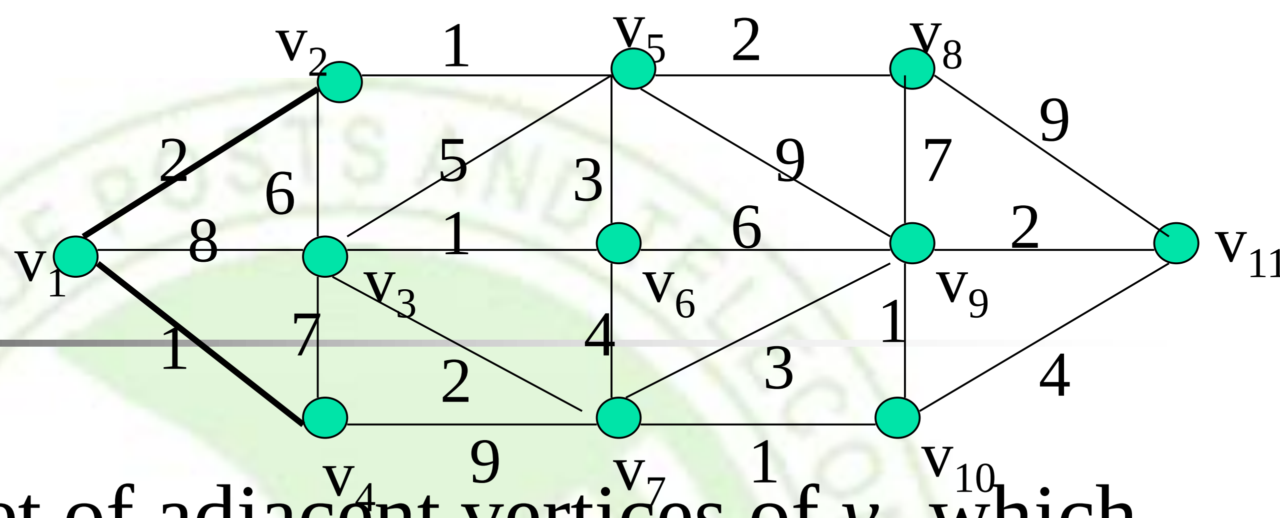
FIGURE 4 Using Dijkstra's Algorithm to Find a Shortest Path from a to z.

Example

- Let l_i represent the length from v_1 to v_i
- Let d_{ij} represent the length of the edge (v_i,v_j)
- Find the shortest path from v_1 to v_{11}

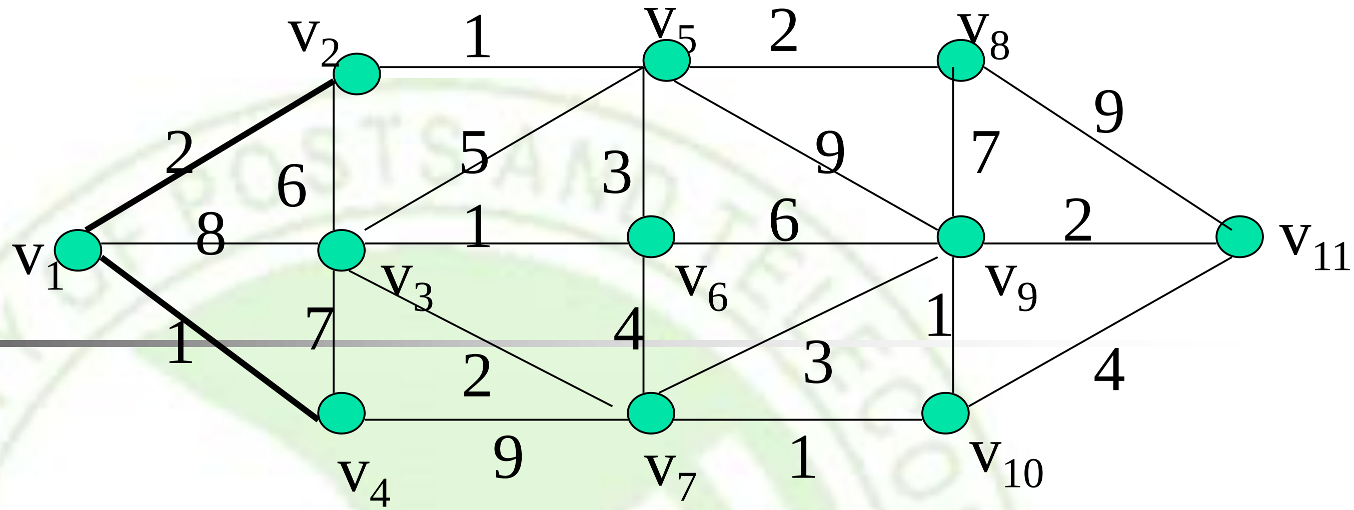


Step 1



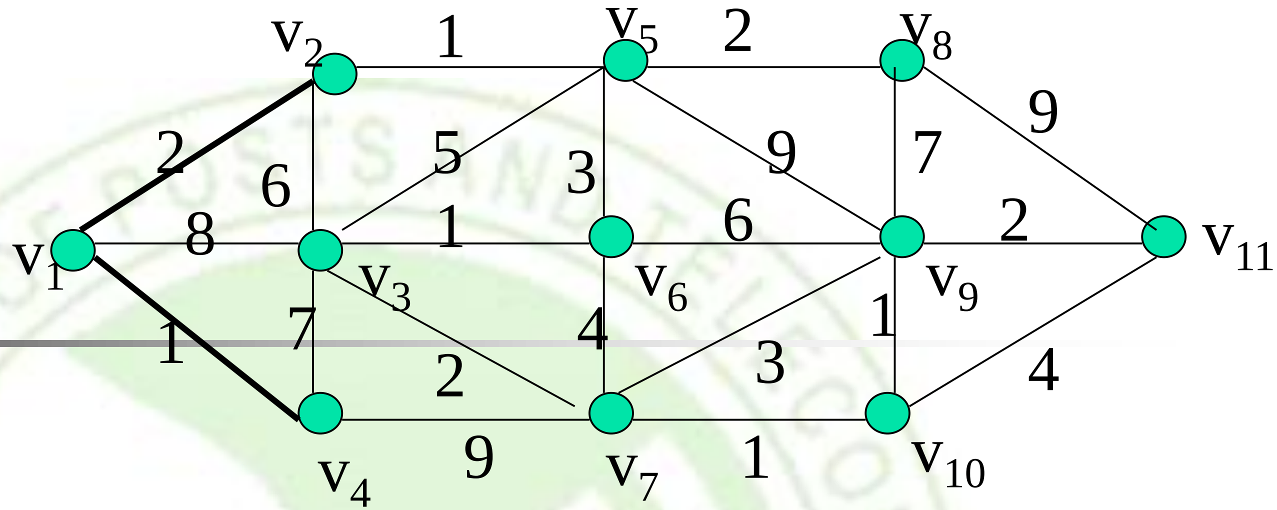
- Find the set of adjacent vertices of v_1,which are $\{v_2, v_3, v_4\}$
- Find the length from v_1 to the vertices in the set.
  - $l_2=d_{12}=2$ $l_3=d_{13}=8$ $l_4=d_{14}=1$
- Find the shortest length of $l_2,l_3\,,l_4$
  - $\mathrm{Min}\{l_2,l_3\,,l_4\}= l_4=1$
- Connect v_1 to v_4

Step 2



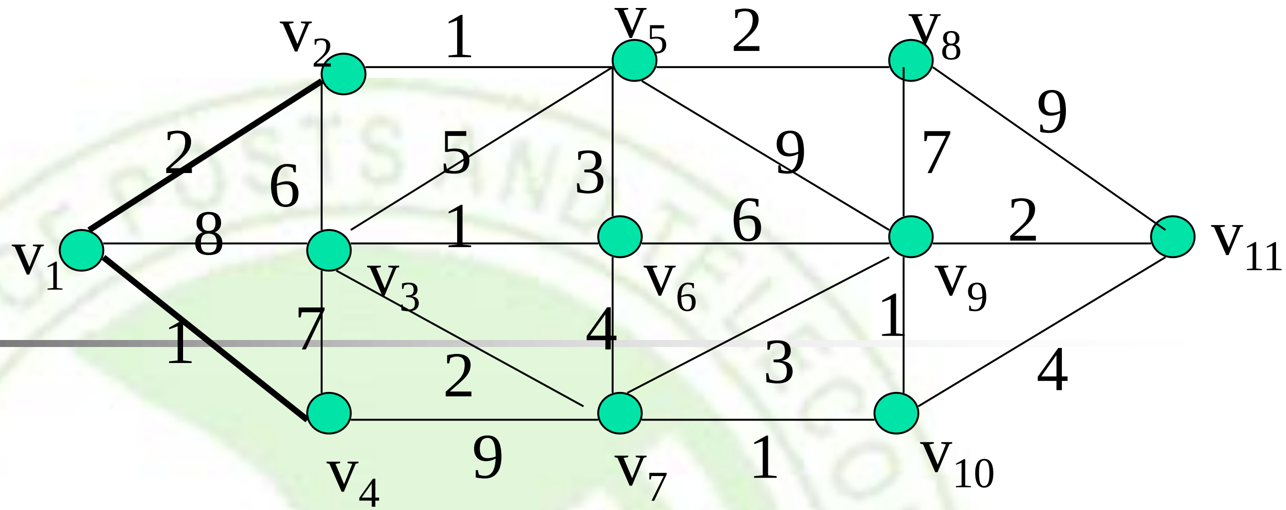
- Find the set of adjacent vertices of $\{v_1, v_4\}$ which are $\{v_2, v_3, v_7\}$
- Find the length from v_1 to the vertices in the set.
  - $l_2=2$ $l_3=8$ $l_7= l_4 + d_{47}=1+9=10$
- Find the shortest length of l_2,l_3,l_7
  - $\text{Min}\{l_2,l_3,l_7\}= l_2=2$
- Connect v_1 to v_2

Step 3



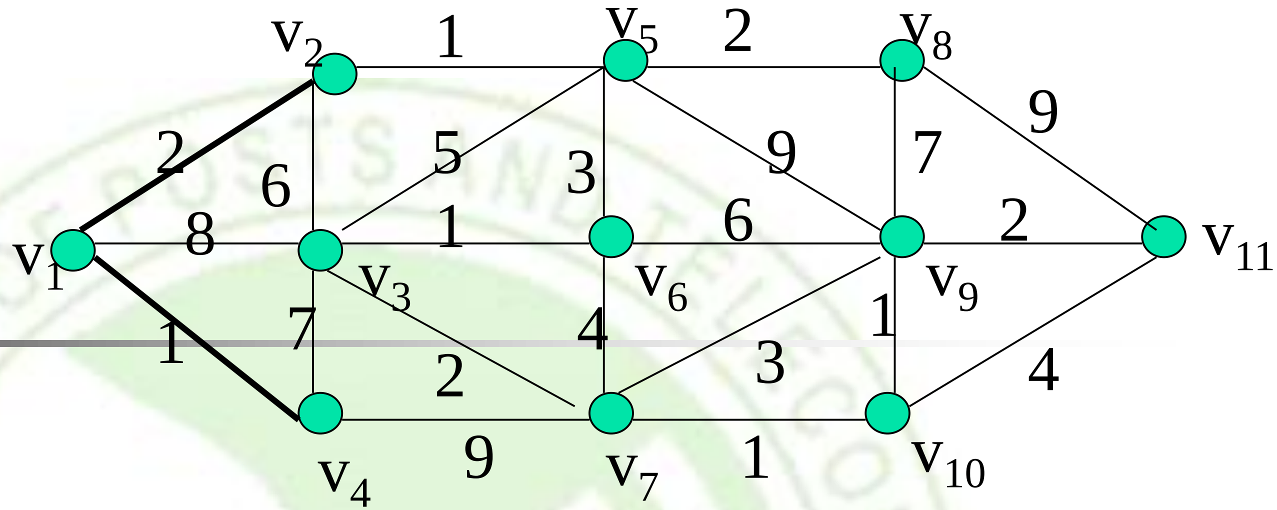
- Find the set of adjacent vertices of $\{v_1, v_2, v_4\}$ which are $\{v_3, v_5, v_7\}$
- Find the length from v_1 to the vertices in the set.
  - $l_3 = \min\{8, l_2 + d_{23}, l_4 + d_{43}\} = \{8,8,8\} = 8$
  - $l_5 = l_2 + d_{25} = 3$
  - $l_7 = l_4 + d_{47} = 10$
- $\min\{l_3, l_5, l_7\} = l_5 = 3$
- Connect v_2 to v_5

Step 4



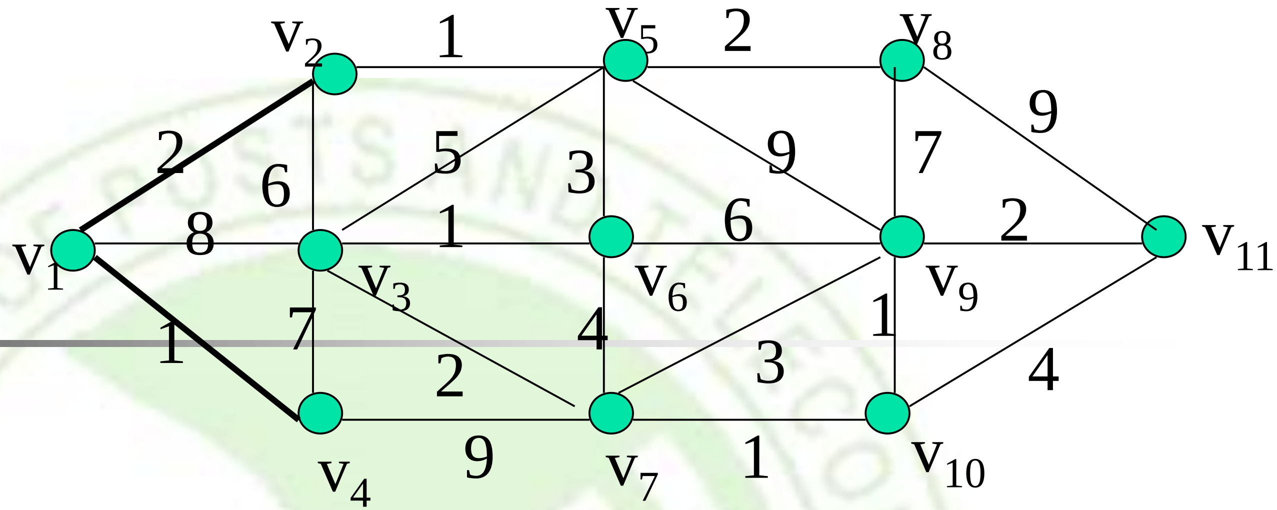
- Find the set of adjacent vertices of $\{v_1, v_2, v_4, v_5\}$ which are $\{v_3, v_6, v_7, v_8, v_9\}$
- Find the length from v_1 to the vertices in the set.
  - $l_3 = 8$ $\quad l_6 = l_5 + d_{56} = 6$ $\quad l_7 = 10$
  - $l_8 = l_5 + d_{58} = 5$ $\quad l_9 = l_5 + d_{59} = 12$
- $\text{Min}\{l_3, l_6, l_7, l_8, l_9\} = l_8 = 5$
- Connect v_5 to v_8

Step 5



- Find the set of adjacent vertices of $\{v_1, v_2, v_4, v_5, v_8\}$ which are $\{v_3, v_6, v_7, v_9, v_{11}\}$
- Find the length from v_1 to the vertices in the set.
  - $l_3=8$ $\quad l_6= l_5+d_{56}=6$ $\quad l_7=10$
  - $l_9= l_8+d_{89}=12$ $\quad l_{11}= l_8+d_{8,11}= 14$
- $\text{Min}\{l_3, l_6, l_7, l_9, l_{11}\}= l_6=6$
- Connect v_5 to v_6

Step 6



- Find the set of adjacent vertices of $\{v_1, v_2, v_4, v_5, v_6, v_8\}$ which are $\{v_3, v_7, v_9, v_{11}\}$
- Find the length from v_1 to the vertices in the set.
  - $l_3 = \min\{8, l_6 + d_{36}\} = \{8,7\} = 7$
  - $l_7 = \min\{l_4 + d_{47}, l_6 + d_{67}\} = \min\{10,10\} = 10$
  - $l_9 = \min\{l_8 + d_{89}, l_6 + d_{69}, l_5 + d_{59}\} = \min\{10,10\} = 10$
  - $l_{11} = 14$
- $\text{Min}\{l_3, l_7, l_9, l_{11}\} = l_3 = 7$
- Connect v_6 to v_3

Step 7

- Continue to
  - connect v_3 to v_7
  - connect v_7 to v_{10}
  - connect v_{10} to v_9
  - connect v_9 to v_{11}
- Then the path from v_1 to v_{11} is the answer.
- At the same time ,we also get the shortest path from v_1 to other vertices in the graph.

Distance Matrix

- Distance Matrix: Adjacent Matrix with Weights
- Let G is a graph with n vertices.
- The distance matrix of G is $D=(d_{ij})_{n\times n}$
  - d_{ij} represent the the weights of the edge (v_i,v_j)
  - If there's no edge between v_i and v_j then $d_{ij}=\infty$

Opreator *

- The distance matrix of G is $D=(d_{ij})_{n\times n}$
  - $D^2 = D*D = (d_{ij}^2)_{n\times n}$
  - $d_{ij}^2 = \min\{d_{i1}+d_{1j}, d_{i2}+d_{2j}, \ldots, d_{in}+d_{nj},\}$
  - *As the same* $D^k=D^{k-1}*D=(d_{ij}^k)_{n\times n}$
  - d_{ij}^k is the shortest length of the path from v_i to v_j with k edges. k条路径的最短长度.

$D^2 = D*D = (d_{ij}^2)_{n\times n}$

$d_{ij}^2 = \min\{d_{i1}+d_{1j}, d_{i2}+d_{2j}, \cdots, d_{in}+d_{nj}\}$

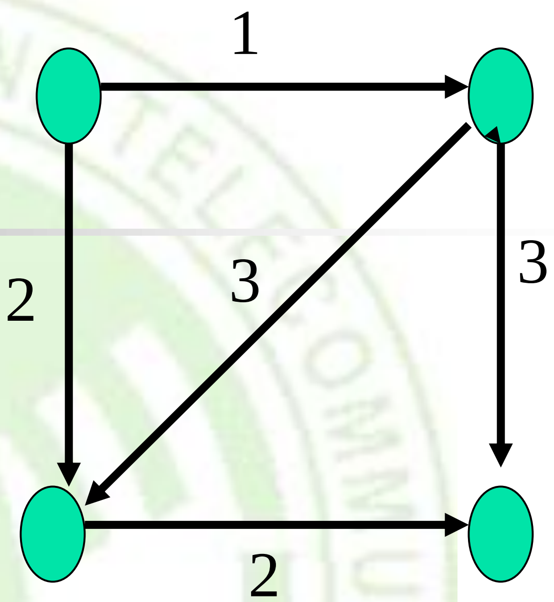
Operator $\oplus$

- Let $A = (a_{ij})_{n\times n}$, $B = (b_{ij})_{n\times n}$
- $C = A \oplus B = (c_{ij})_{n\times n}$
  - $c_{ij} = \min(a_{ij}, b_{ij})$

Definition of Shortest Length Matrix

- Definition of Shortest Length Matrix:
- $P = D \oplus D^2 \oplus D^3 \oplus \ldots \oplus D^n$
  - $p_{ij} = (p_{ij})_{n \times n}$
  - p_{ij} represent the shortest length from v_i to v_j

Example



$$D = \begin{bmatrix} \infty & 1 & 2 & \infty \\ \infty & \infty & 3 & 3 \\ \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

$$D^2 = \begin{bmatrix} \infty & 1 & 2 & \infty \\ \infty & \infty & 3 & 3 \\ \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix} * \begin{bmatrix} \infty & 1 & 2 & \infty \\ \infty & \infty & 3 & 3 \\ \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & 4 & 4 \\ \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

用弗洛伊德算法求最短路径

$$D^3 = \begin{bmatrix} \infty & \infty & 4 & 4 \\ \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} * \begin{bmatrix} \infty & 1 & 2 & \infty \\ \infty & \infty & 3 & 3 \\ \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & \infty & 6 \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

- $D_4=(\propto)_{4\times 4}$
- $P=D \oplus D^2 \oplus D^3 \oplus D^4$
- note: ex.21 floyd

$$P = \begin{bmatrix} \infty & 1 & 2 & 4 \\ \infty & \infty & 3 & 3 \\ \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

弗洛伊德算法

Floyd algorithm

procedure *Floyd*(G: weighted simple graph)

{G has vertices $v_1, v_2, \ldots, v_n$ and weights $w(v_i, v_j)$
  with $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge}

for $i := 1$ **to** n
  for $j := 1$ **to** n
    $d(v_i, v_j) := w(v_i, v_j)$

for $i := 1$ **to** n
  for $j := 1$ **to** n
    for $k := 1$ **to** n
      if $d(v_j, v_i) + d(v_i, v_k) < d(v_j, v_k)$
        then $d(v_j, v_k) := d(v_j, v_i) + d(v_i, v_k)$

return $[d(v_i, v_j)]$ {$d(v_i, v_j)$ is the length of a shortest path between v_i and v_j for $1 \le i \le n$, $1 \le j \le n$}

The traveling salesman problem

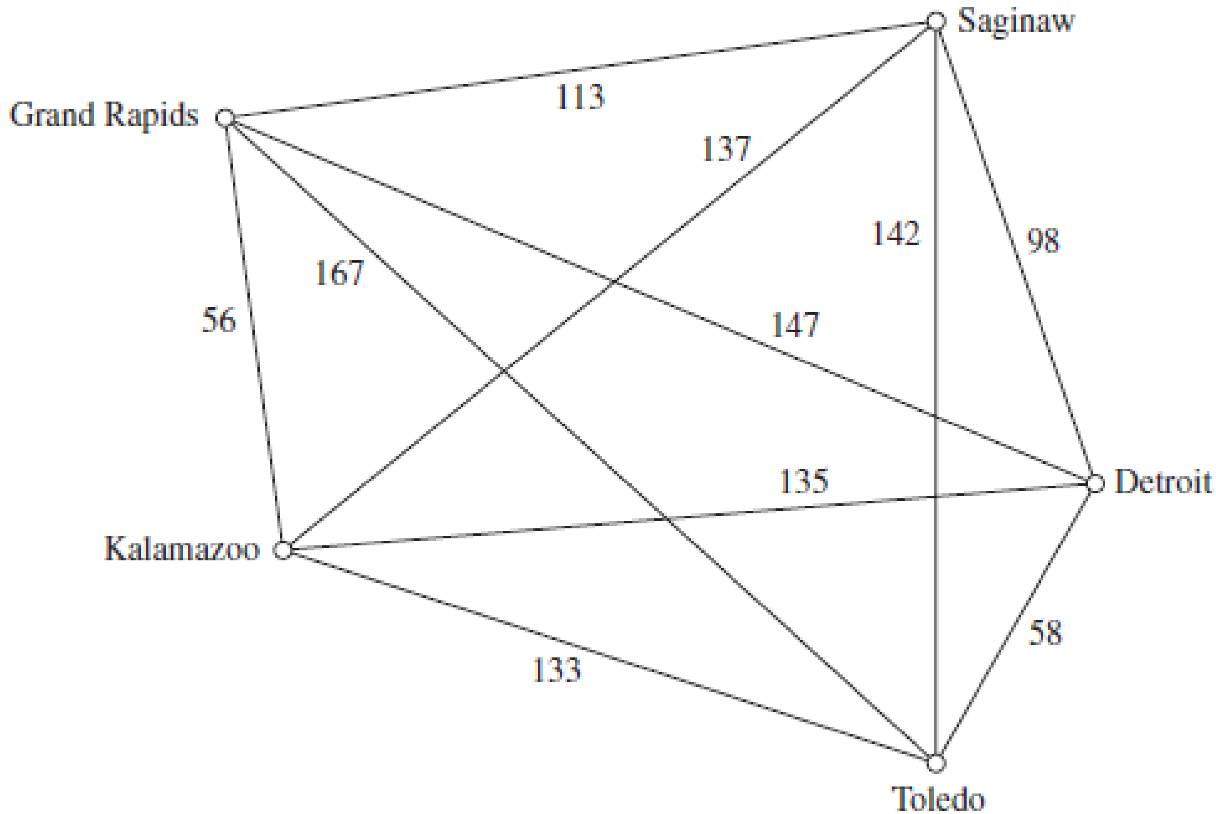


FIGURE 5 The Graph Showing the Distances between Five Cities.

- The traveling salesperson problem asks for the Hamilton circuit of minimum total weight in a weighted,complete,undirected graph that visits each vertex exactly once and returns to its starting point.
- we need only examine $(n-1)!/2$ circuits to

find our answer.

homework

- § 10.6
  - 8(shortest), 16, 18(not be unique), 26(salesman)