Chapter 8 Graph Theory 图论

Lu Han

hl@bupt.edu.cn

8.3 Hamiltonian Cycles and the Traveling Salesperson Problem 哈密顿回路和旅行商问题

Hamiltonian Cycle (哈密顿回路): a cycle in a graph G that visits each vertex once.

(Unlike the situation for Euler cycles, no easily verified necessary and sufficient conditions are known for the existence of a Hamiltonian cycle in a graph.)

Theorems 8.2.17 and 8.2.18: G is an Euler graph if and only if G is connected and all its vertices have even degree.

- A **cycle (or circuit) (回路或者环路)** is a path of nonzero length from v to v with no repeated edges.
- **Euler Cycle (欧拉回路)：** a cycle in a graph G that includes all of the edges and all of the vertices of G.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.

```
randomized_hamiltonian_cycle(G, n) {
  if (n == 1 ∨ n == 2) // trivial cases
    return false
  v_1 = random vertex in G
  i = 1
  while (i ¬= n ∨ v_1 ∉ N(v_i)) {
    N = N(v_i) − {v_1, . . . , v_{i−1}} // if i is 1, {v_1, . . . , v_{i−1}} is ∅
    // N contains the vertices adjacent to v_i (the current last vertex
    // of the path) that are not already on the path
    if (N ≠ ∅) {
      i = i + 1
      v_i = random vertex in N
    }
    else if (v_j ∈ N(v_i) for some j, 1 ≤ j < i − 1)
      (v_1, . . . , v_i) = (v_1, . . . , v_j, v_i, . . . , v_{j+1})
    else
      return false
  }
  return true
}
```

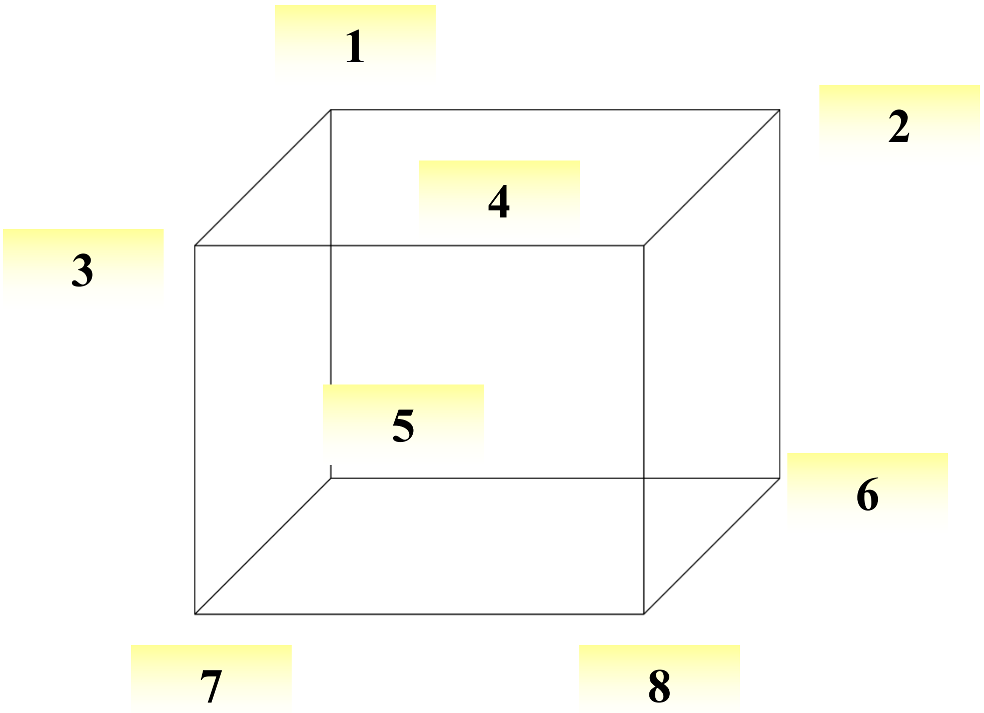
Randomized Hamiltonian Cycle 随机哈密顿回路算法

Suggest ways to improve the algorithm.

Randomized Hamiltonian Cycle 随机哈密顿回路算法

Input: A simple graph $G = (V, E)$ with n vertices.

Output: If the algorithm terminates, it returns true if it finds a Hamiltonian cycle and false otherwise.



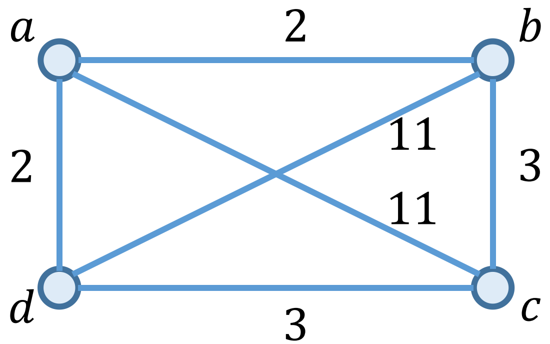
The 3-cube

Traveling Salesperson Problem 旅行商问题

Given a weighted graph G, find a minimum-length Hamiltonian cycle in G.

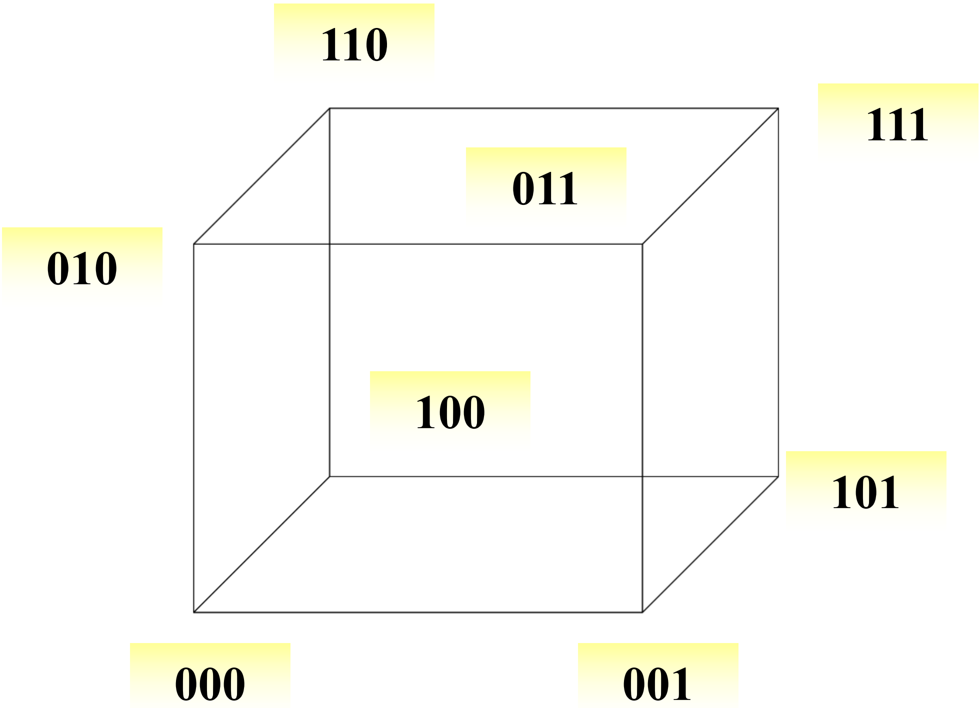
- Vertices: Cities;
- Edges Weights: Distances;
- Problem: Find a shortest route in which the salesperson can visit each city one time, stating and ending at the same city.

Example 8.3.4



Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

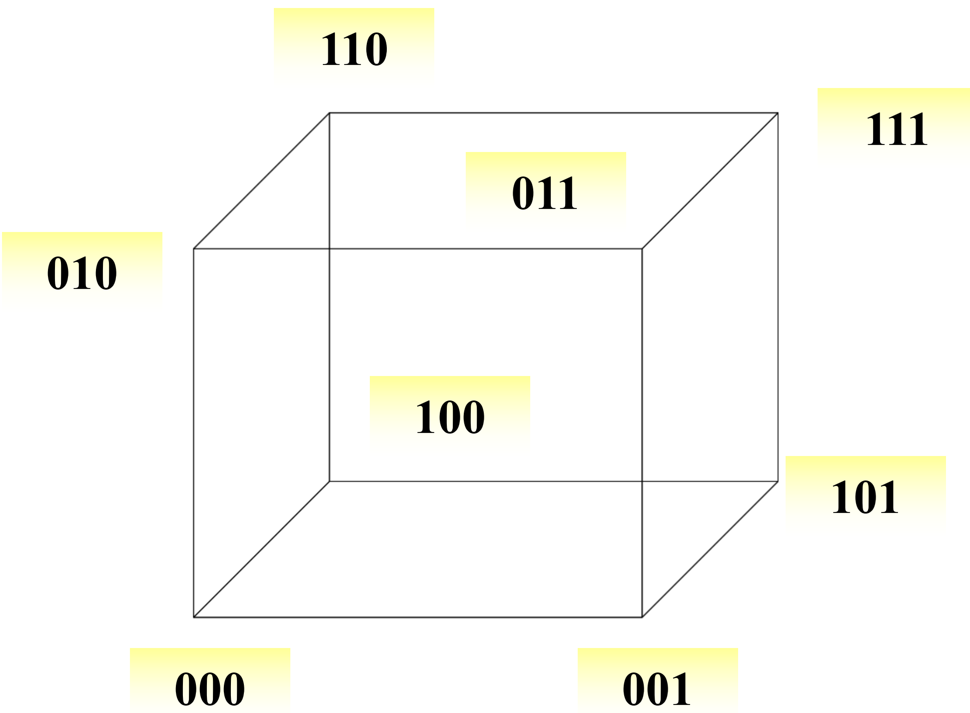
Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)



The 3-cube

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)



The 3-cube

For a 3-Cube whether the following statements are true or false?

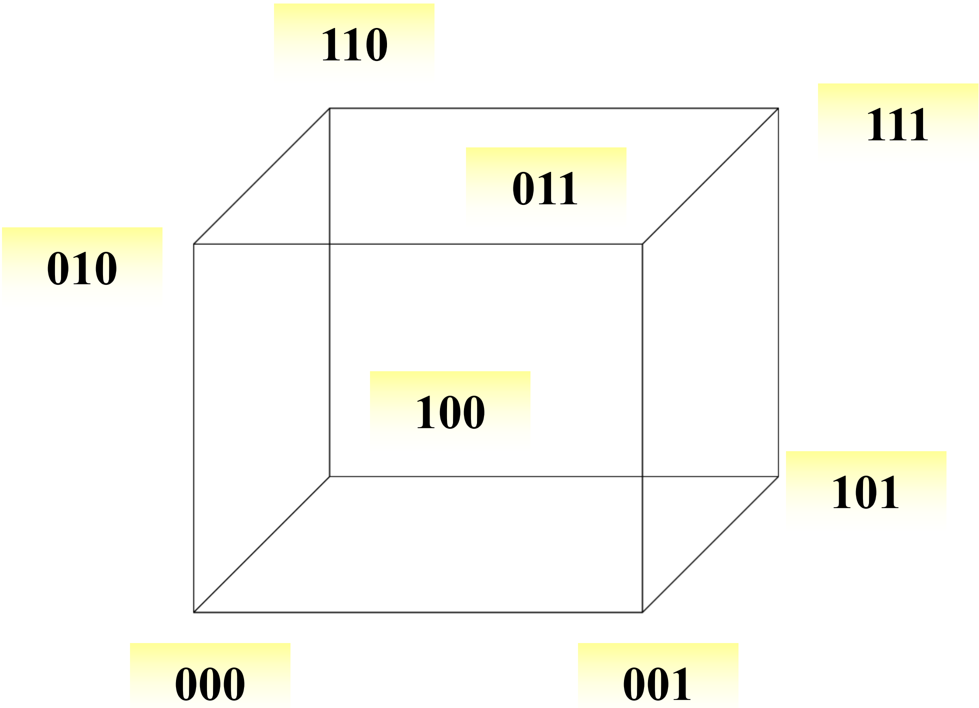
(a) There Exists Euler cycle ?

(b) There Exists Hamiltonian cycle?

(c) Is it a bipartite?

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)



The 3-cube

For an n-Cube ($n \geq 2$) whether the following statements are true or false?

(a) There Exists Euler cycle ?

(b) There Exists Hamiltonian cycle?

(c) Is it a bipartite?

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)

The n-cube has a Hamiltonian cycle if and only if $n \geq 2$ and there is a sequence,

$$s_1, s_2, \ldots, s_{2^n},$$

where each s_i is a string of n bits, satisfying:

- Every n-bit string appears somewhere in the sequence.
- s_i and s_{i+1} differ in exactly one bit, $i = 1, \ldots, 2^n - 1$.
- s_{2^n} and s_1 differ in exactly one bit.

The sequence $s_1, s_2, \ldots, s_{2^n}$ is called a **Gray code (格雷码)**.

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)

The n-cube has a Hamiltonian cycle if and only if $n \geq 2$ and there is a sequence,

$$s_1, s_2, \ldots, s_{2^n},$$

where each s_i is a string of n bits, satisfying:

- Every n-bit string appears somewhere in the sequence.
- s_i and s_{i+1} differ in exactly one bit, $i = 1, \ldots, 2^n - 1$.
- s_{2^n} and s_1 differ in exactly one bit.

The sequence $s_1, s_2, \ldots, s_{2^n}$ is called a **Gray code (格雷码)**.

When $n \geq 2$, a Gray codes corresponds to the Hamiltonian cycle $s_1, s_2, \ldots, s_{2^n}$, s_1 since every vertex appears and the edges (s_i, s_{i+1}), $i = 1, \ldots, 2^n - 1$ and (s_{2^n}, s_1) are distinct.

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)

The n-cube has a Hamiltonian cycle if and only if $n \geq 2$ and there is a sequence,

$$s_1, s_2, \ldots, s_{2^n},$$

where each s_i is a string of n bits, satisfying:

- Every n-bit string appears somewhere in the sequence.
- s_i and s_{i+1} differ in exactly one bit, $i = 1, \ldots, 2^n - 1$.
- s_{2^n} and s_1 differ in exactly one bit.

The sequence $s_1, s_2, \ldots, s_{2^n}$ is called a **Gray code (格雷码)**.

When $n = 1$, the Gray codes $0, 1$ corresponds to the path $(0, 1, 0)$ which is not a cycle becasue the edge $(0, 1)$ is repeated.

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Theorem 8.3.6 Let G_1 denote the sequence $0, 1$. We define G_n in terms of G_{n-1} by the following rules:

(a) Let G_{n-1}^R denote the sequence G_{n-1} written in reverse.

(b) Let $G_{n-1}^{'}$ denote the sequence obtained by prefixing each member of G_{n-1} with 0.

(c) Let $G_{n-1}^{''}$ denote the sequence obtained by prefixing each member of G_{n-1}^R with 1.

(d) Let G_n be the sequence consisting of $G_{n-1}^{'}$ followed by $G_{n-1}^{''}$.

Then G_n is a Gray code for every positive integer n.

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Theorem 8.3.6 Let G_1 denote the sequence $0, 1$. We define G_n in terms of G_{n-1} by the following rules:
(a) Let G_{n-1}^R denote the sequence G_{n-1} written in reverse.
(b) Let G_{n-1}' denote the sequence obtained by prefixing each member of G_{n-1} with 0.
(c) Let G_{n-1}'' denote the sequence obtained by prefixing each member of G_{n-1}^R with 1.
(d) Let G_n be the sequence consisting of G_{n-1}' followed by G_{n-1}''.
Then G_n is a Gray code for every positive integer n.

G_1	0	1
G_1^R		
G_1'		
G_1''		
G_2		
G_2^R		
G_2'		
G_2''		
G_3		

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Theorem 8.3.6 Let G_1 denote the sequence $0, 1$. We define G_n in terms of G_{n-1} by the following rules:
(a) Let G_{n-1}^R denote the sequence G_{n-1} written in reverse.
(b) Let $G_{n-1}^{'}$ denote the sequence obtained by prefixing each member of G_{n-1} with 0.
(c) Let $G_{n-1}^{''}$ denote the sequence obtained by prefixing each member of G_{n-1}^R with 1.
(d) Let G_n be the sequence consisting of $G_{n-1}^{'}$ followed by $G_{n-1}^{''}$.
Then G_n is a Gray code for every positive integer n.

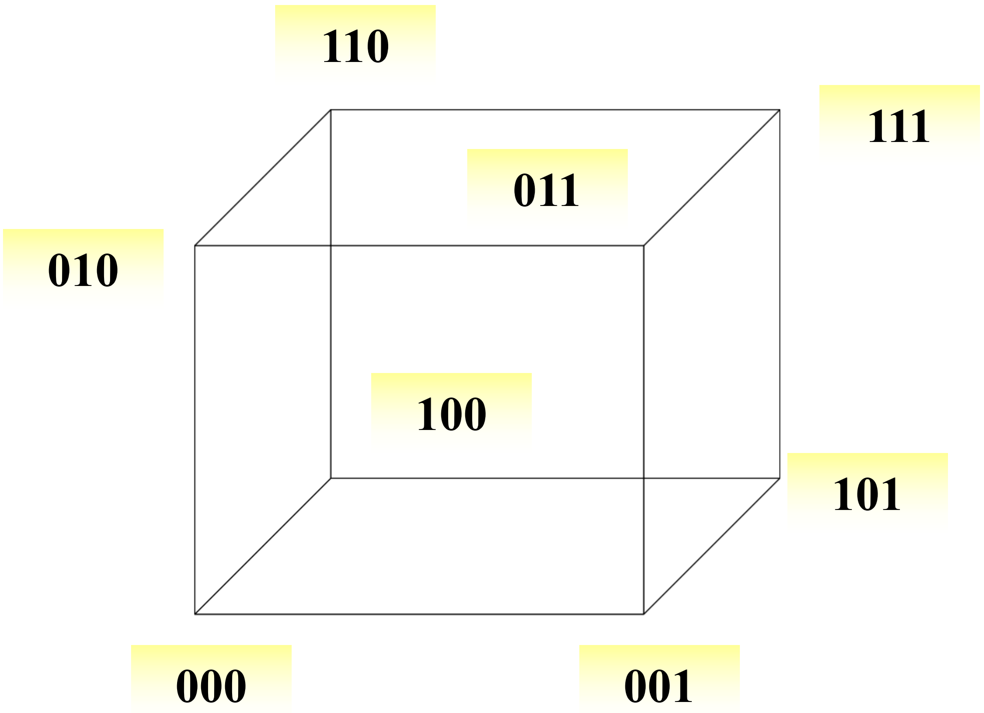
Proof We prove the theorem by induction on n.

Basic Step ($\boldsymbol{n = 1}$)
Since the sequence 0, 1 is a Gray code, the theorem is true when n is 1.

Inductive Step
Assume that G_{n-1} is a Gray code. Prove that G_n is a Gray code.

Gray Codes (格雷码) and Hamiltonian Cycles in the n-Cube

Example 8.1.8 The n-Cube (Hypercube) n-立方体 (超立方体)

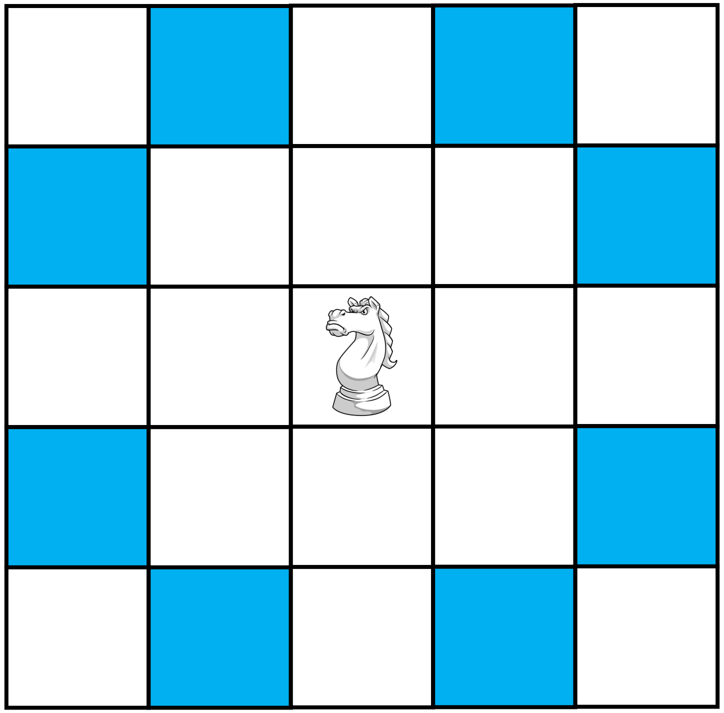


The 3-cube

Corollary 8.3.7 The n-cube has a Hamiltonian cycle for every positive integer $n \geq 2$.

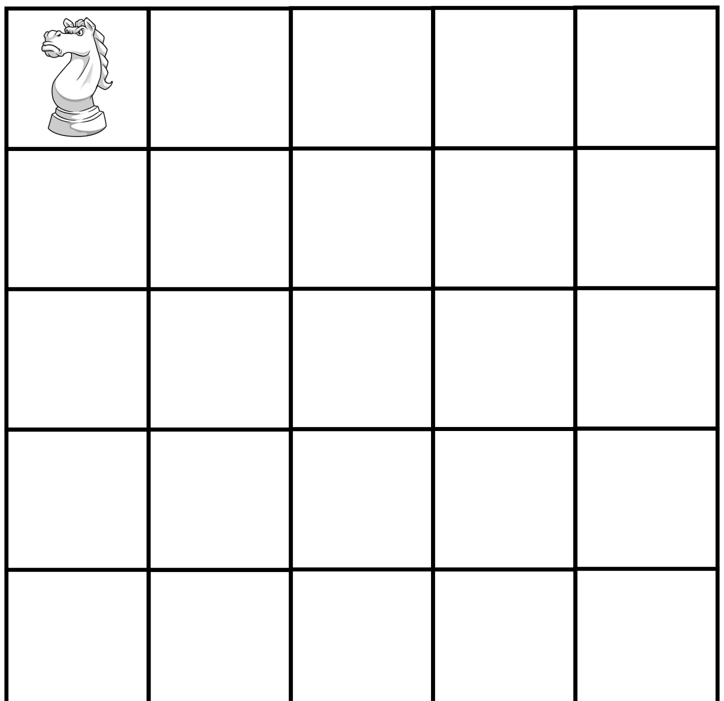
The knight’s Tour 骑士遍历问题

Example 8.3.9 A knight’s tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight’s tour exists.



The knight's Tour 骑士遍历问题

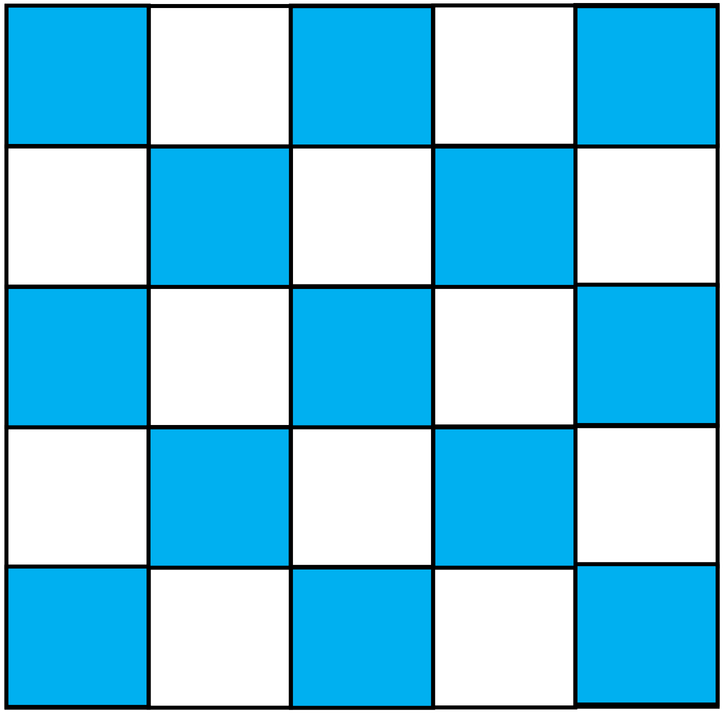
Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.



Do you think $\mathbf{n} = 5$ is OK for me?

The knight's Tour 骑士遍历问题

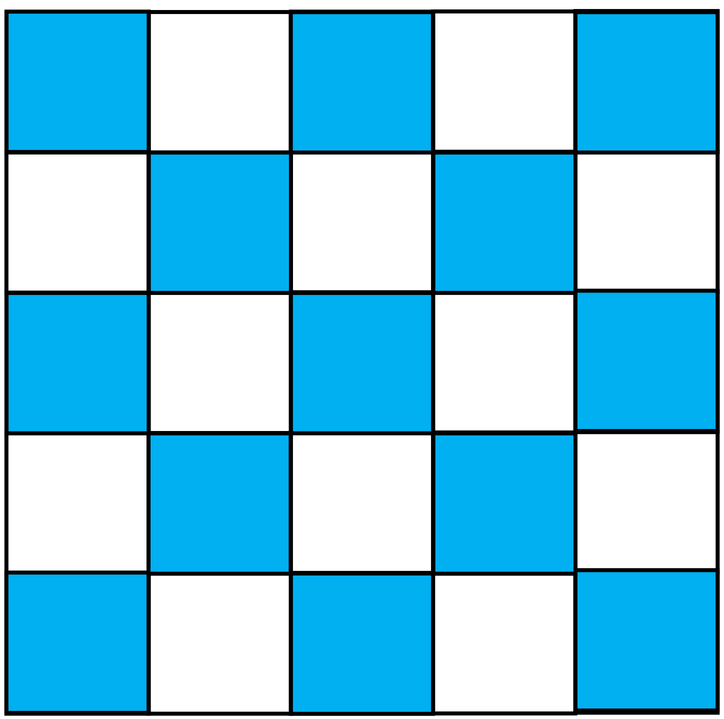
Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.



Do you think $n = 5$ is OK for me?

The knight's Tour 骑士遍历问题

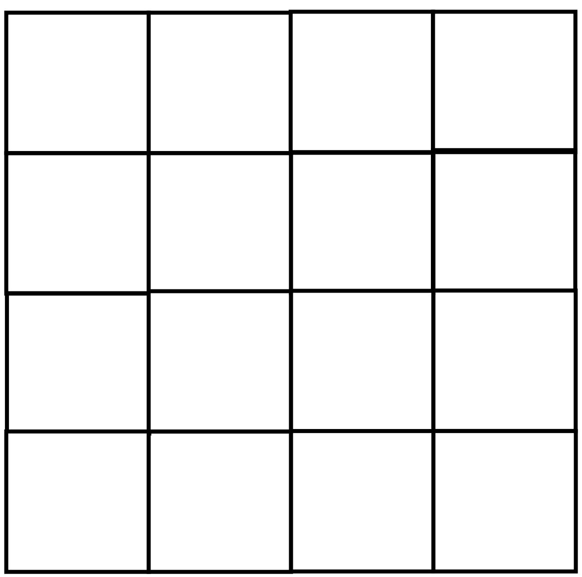
Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.



If GK_n has a Hamiltonian cycle, n is even.

The knight's Tour 骑士遍历问题

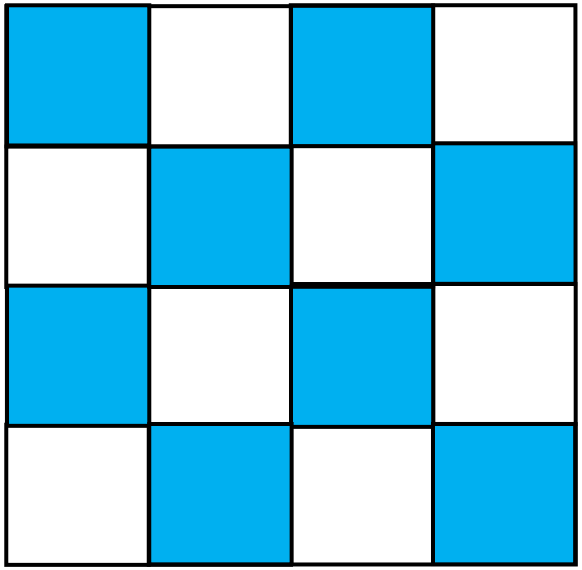
Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.



Do you think $\mathbf{n = 4}$ is OK for me?

The knight's Tour 骑士遍历问题

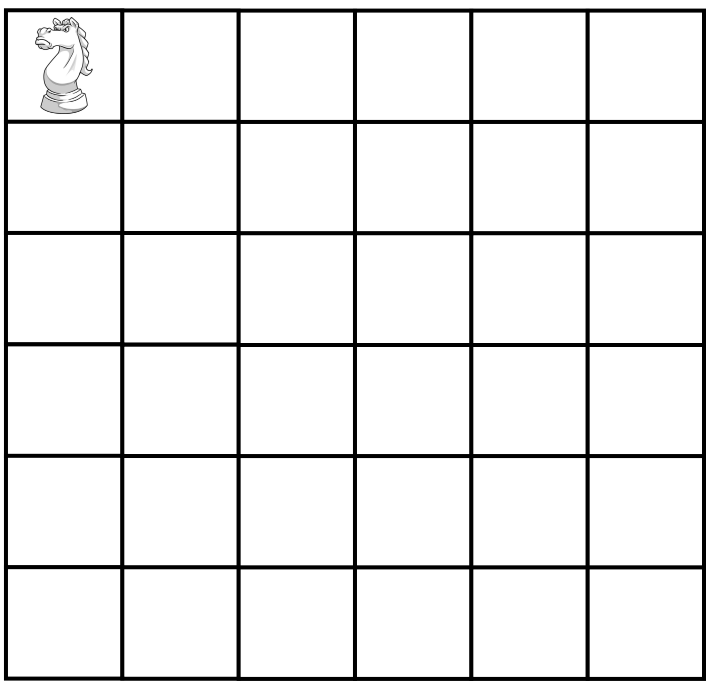
Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.



Outside & Inside Squares

The knight's Tour 骑士遍历问题

Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.



Do you think $\mathbf{n} = 6$ is OK for me?

The knight's Tour 骑士遍历问题

Example 8.3.9 A knight's tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight's tour exists.

	16	27	10	7	18
28	11	36	17	26	9
15	2	29	8	19	6
12	35	14	23	32	25
3	22	33	30	5	20
34	13	4	21	24	31

Do you think $n = 6$ is OK for me?

The knight’s Tour 骑士遍历问题

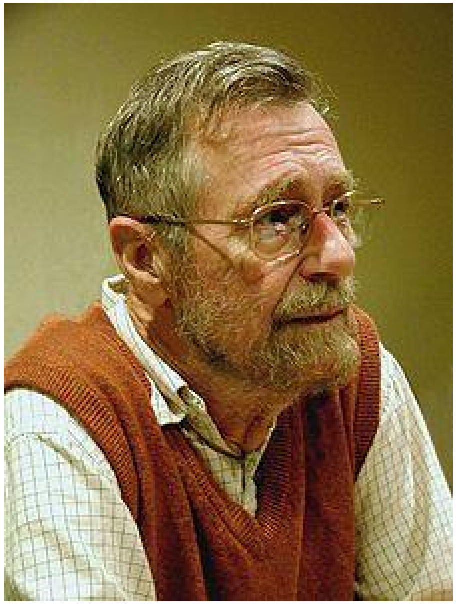
Example 8.3.9 A knight’s tour of an $n \times n$ board begins at some square, visits each square exactly once making legal mobes, and returns to the initial square. The problem is to determine for which n a knight’s tour exists.

	16	27	10	7	18
28	11	36	17	26	9
15	2	29	8	19	6
12	35	14	23	32	25
3	22	33	30	5	20
34	13	4	21	24	31

GK_n has a Hamiltonian cycle for all even $n \geq 6$.

8.4 A Shortest-Path Algorithm 最短路径算法

Due to **Edsger W. Dijkstra (艾兹格·迪科斯彻)**, Dutch computer scientist born in 1930.



Edsger W. Dijkstra (1930–2002) was born in The Netherlands. He was an early proponent of programming as a science. So dedicated to programming was he that when he was married in 1957, he listed his profession as a programmer. However, the Dutch authorities said that there was no such profession, and he had to change the entry to "theoretical physicist." He won the prestigious Turing Award in 1972.

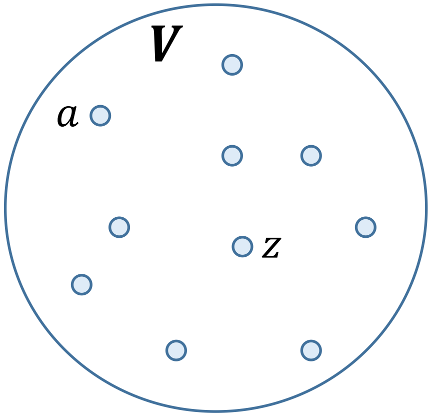
Dijkstra's algorithm (狄克斯特拉算法) finds the length of the shortest path from a single vertex to any other vertex in a connected weighted graph.

Dijkstra's algorithm (狄克斯特拉算法)

The given graph G is **connected, weighted graph**. Assume that **the weights are positive numbers**. We want to find a shortest path from vertex a to vertex z.

Dijkstra's algorithm (狄克斯特拉算法)

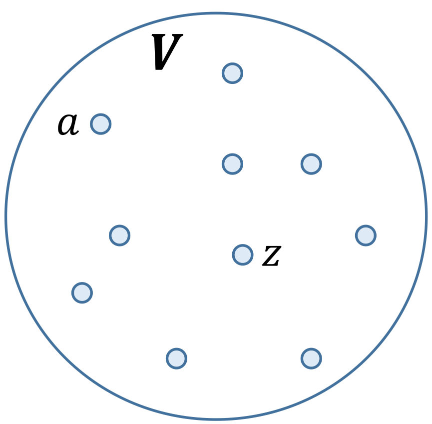
The given graph G is **connected, weighted graph**. Assume that **the weights are positive numbers**. We want to find a shortest path from vertex a to vertex z.



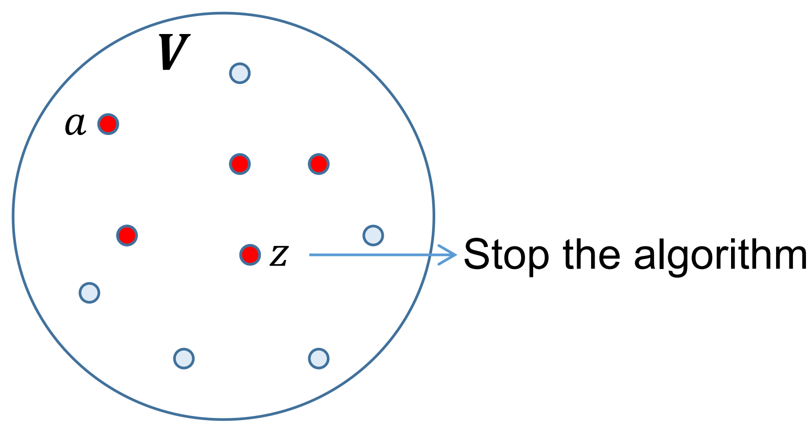
Initialization. Temporarily labelled each vertex $v \in V$ with a value $L(v)$.

Dijkstra's algorithm (狄克斯特拉算法)

The given graph G is **connected, weighted graph**. Assume that **the weights are positive numbers**. We want to find a shortest path from vertex a to vertex z.



Initialization. Temporarily labelled each vertex $v \in V$ with a value $L(v)$.



Each iteration changes the status of one temporairly labelled vertex from temporary to permanent. Update the label of some related vertices.

Dijkstra's algorithm (狄克斯特拉算法)

Input: A connected, weighted graph in which all weights are positive; vertices a and z.
Output: $L(z)$, the length of a shortest path from a to z.

Dijkstra's algorithm (狄克斯特拉算法)

Input: A connected, weighted graph in which all weights are positive; vertices a and z.

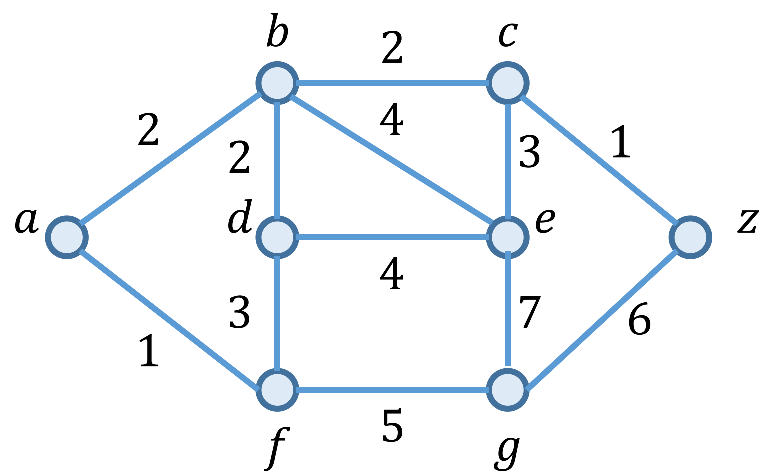
Output: $L(z)$, the length of a shortest path from a to z.

procedure $dijkstra(w, a, z, L)$
  $L(a) := 0$
  for each node $x \neq a$ **do**
  $L(x) := \infty$
  T:= set of all nodes
  //T is the set of vertices whose shortest
  // distance from a has not been found
  while $z \in T$ **do**
    chose $v \in T$ with minimum $L(v)$
    $T := T - \{v\}$
    for each $x \in T$ adjacent to v **do**
      $L(x) := min\,\{L(x),\ L(v) + w(v, x)\}$
    end
  end dijkstra

Dijkstra's algorithm (狄克斯特拉算法)

Example 8.4.2

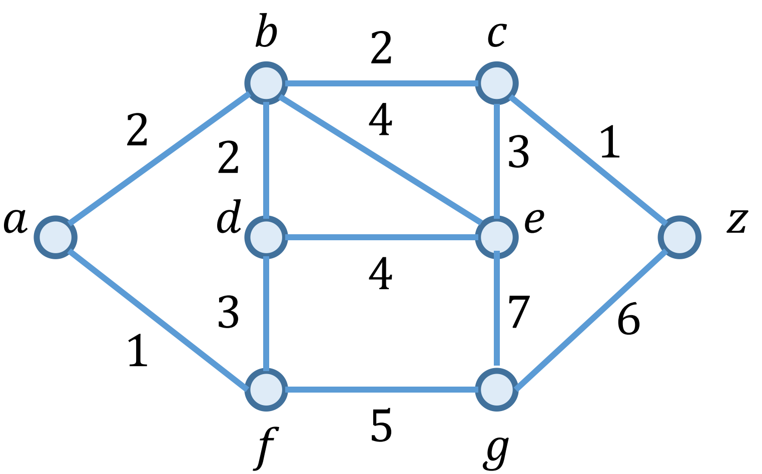
procedure $dijkstra(w, a, z, L)$
  $L(a) := 0$
  for each node $x \neq a$ **do**
  $L(x) := \infty$
 T:= set of all nodes
 //T is the set of vertices whose shortest
 // distance from a has not been found
 while $z \in T$ **do**
   chose $v \in T$ with minimum $L(v)$
   $T := T - \{v\}$
   for each $x \in T$ adjacent to v **do**
    $L(x) := min\,\{L(x),\ L(v) + w(v, x)\}$
   end
 end dijkstra



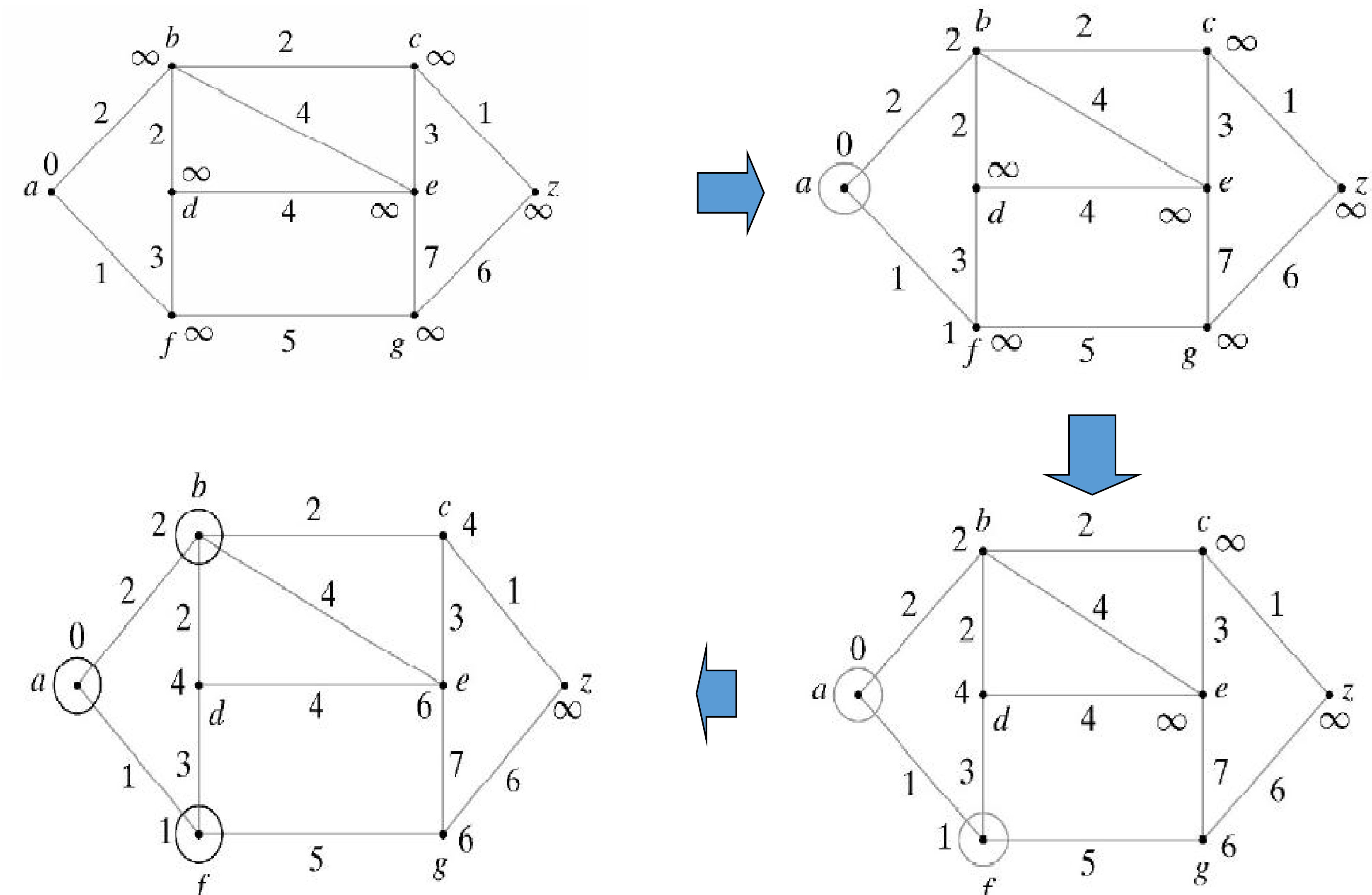
Dijkstra's algorithm (狄克斯特拉算法)

Example 8.4.2

procedure $dijkstra(w, a, z, L)$
$L(a) := 0$
for each node $x \neq a$ **do**
$L(x) := \infty$
T:= set of all nodes
//T is the set of vertices whose shortest
// distance from a has not been found
while $z \in T$ **do**
chose $v \in T$ with minimum $L(v)$
$T := T - \{v\}$
for each $x \in T$ adjacent to v **do**
$L(x) := min\,\{L(x),\ L(v) + w(v, x)\}$
end
end dijkstra

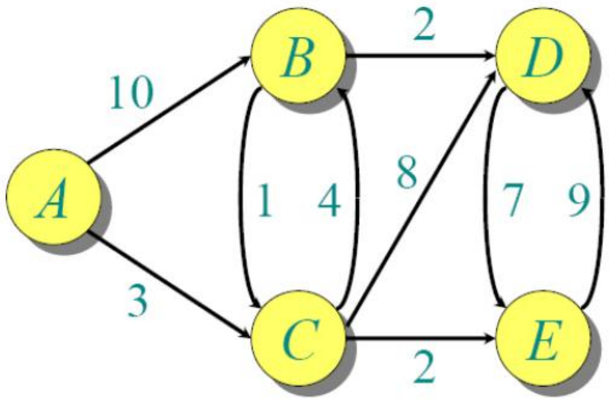


a	b	c	d	e	f	g	z



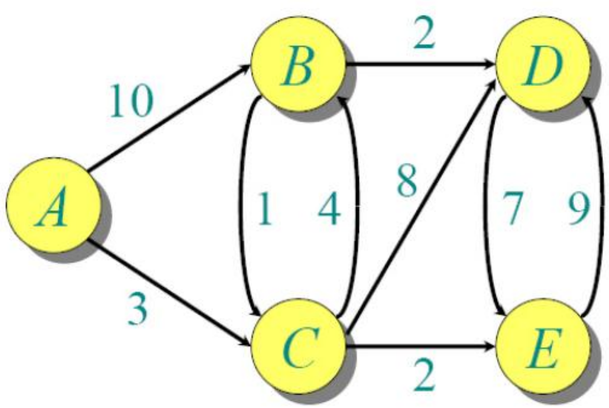
8.4 A Shortest-Path Algorithm 最短路径算法

Exercise Use Dijkstra's Shorest-Path Algorithm to find the length of a shortest path from A to D.



8.4 A Shortest-Path Algorithm 最短路径算法

Exercise Use Dijkstra's Shorest-Path Algorithm to find the length of a shortest path from A to D.



A	B	C	D	E
$\boxed{0}$	∞	∞	∞	∞
	10	$\boxed{3}$	∞	∞
	7		11	$\boxed{5}$
	$\boxed{7}$		11	
			$\boxed{9}$	

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra’s Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra’s shortest-path algorithm correctly finds the length of a shortest path form a to z.

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

In addition to circle a vertex, we will also label it with the name of the vertex from which it was labeled.

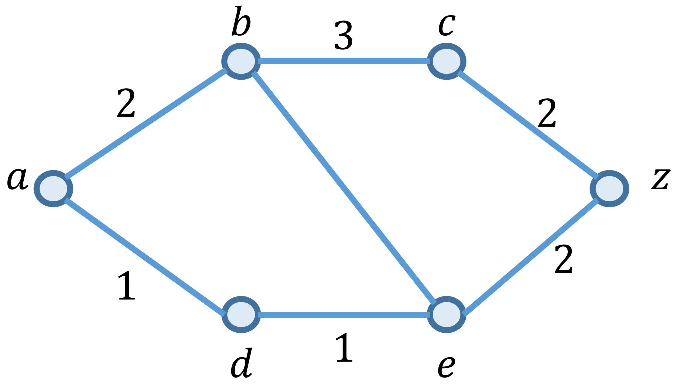
8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shortest-Path Algorithm

Theorem 8.4.3 Dijkstra's shortest-path algorithm correctly finds the length of a shortest path form a to z.

In addition to circling a vertex, we will also label it with the name of the vertex from which it was labeled.

Example 8.4.4



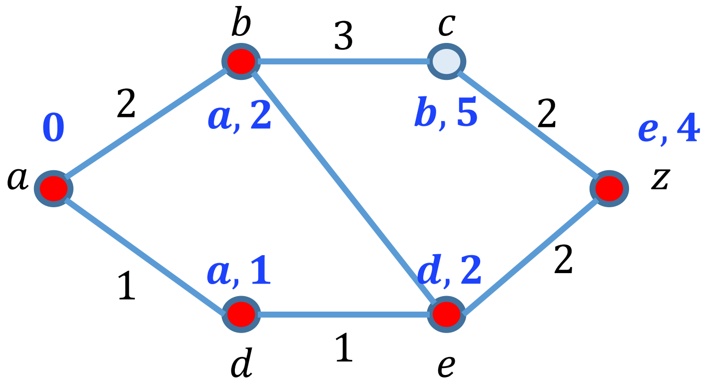
8.4 A Shortest-Path Algorithm 最短路径算法

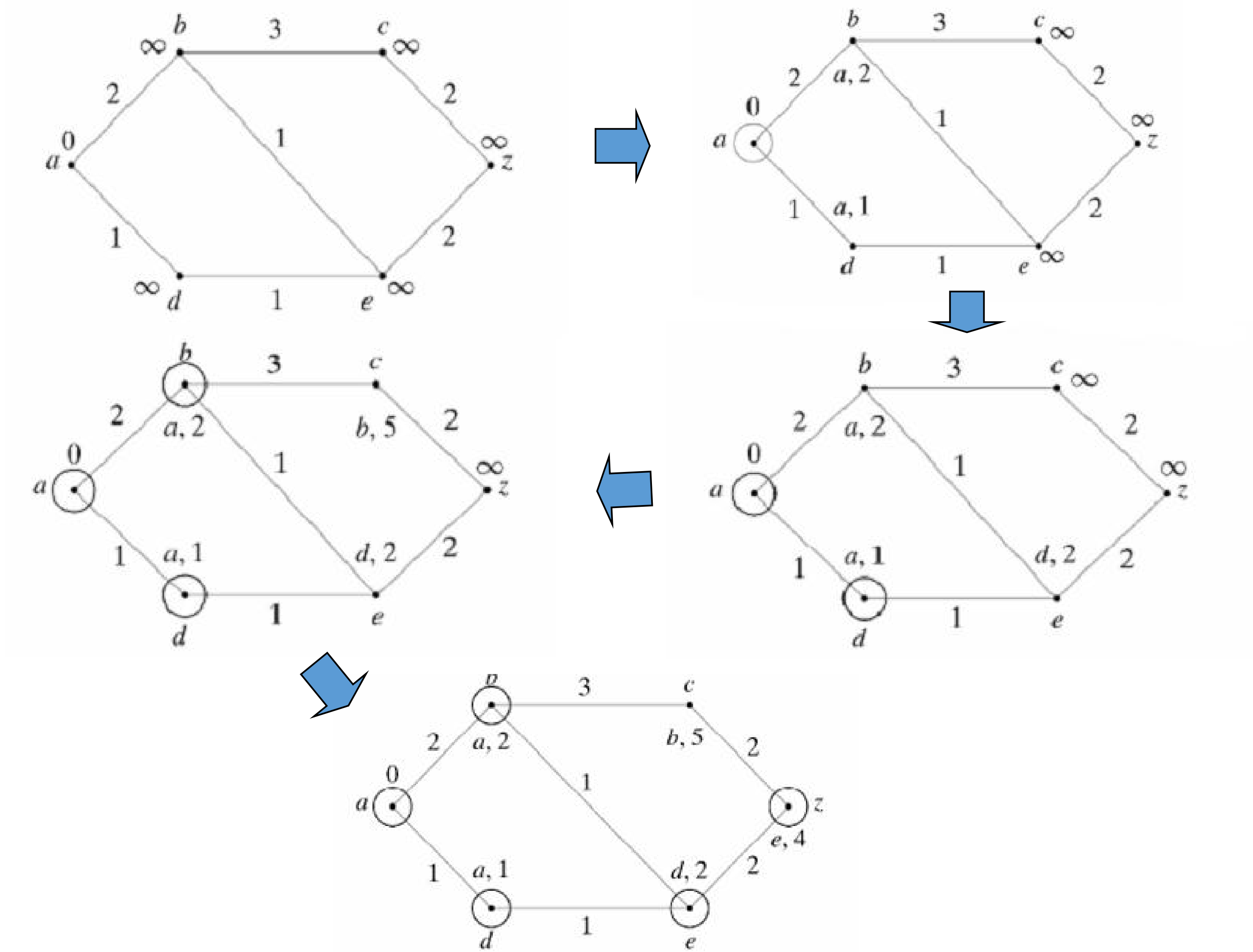
Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

In addition to circling a vertex, we will also label it with the name of the vertex from which it was labeled.

Example 8.4.4




b 3 c
∞ ∞
2 2
0 1 ∞
a z
1 2
∞ d 1 e ∞
b 3 c ∞
2 a, 2 2
0 1 ∞
a z
1 a, 1 2
d 1 e ∞
b 3 c ∞
2 a, 2 2
0 1 ∞
a z
1 a, 1 d, 2 2
d 1 e
b 3 c
2 a, 2 b, 5 2
0 1 ∞
a z
1 a, 1 d, 2 2
d 1 e
3 c
2 a, 2 b, 5 2
0 1
a z
e, 4
1 a, 1 d, 2 2
d 1 e

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z.

We want to prove that

(i) $L(z) \geq$ length of P

(ii) $L(z) \leq$ length of P

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shortest-Path Algorithm

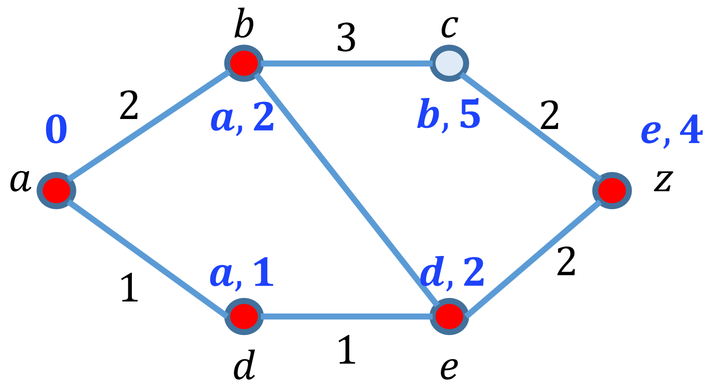
Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z.

We want to prove that

(i) $L(z) \geq$ length of P

(ii) $L(z) \leq$ length of P



8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z.

We want to prove that

(i) $L(z) \geq$ length of P

(ii) $L(z) \leq$ length of P

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z. (ii) $L(z) \leq$ length of P

Proof We use mathematical induction on i to prove that the ith time we choose a vertex v with minimum $L(v)$, $L(v)$ is the length of a shortest path from a to v.

8.4 A Shortest-Path Algorithm 最短路径算法

Dijkstra's Shorest-Path Algorithm

Theorem 8.4.3 Dijkstra's shorest-path algorithm correctly finds the length of a shortest path form a to z.

Let P be a shortest path from a to z. (ii) $L(z) \leq$ length of P

Proof We use mathematical induction on i to prove that the ith time we choose a vertex v with minimum $L(v)$, $L(v)$ is the length of a shortest path from a to v.

Basis Step ($i = 1$)

Inductive Step ($k < i$)
If there is a path from a to w whose length is less than $L(v)$, then w is not in T.

8.4 A Shortest-Path Algorithm 最短路径算法

Modify Dijkstra's shortest-path algorithm so that it accepts a weighted graph that is not necessarily connected. At termination, what is $L(z)$ if there is no path from a to z?

8.4 A Shortest-Path Algorithm 最短路径算法

True or false? When a connected, weighted graph and vertices a and z are input to the following algorithm, it returns the length of a shortest path from a to z.

Algorithm 8.4.6

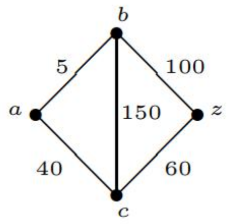
```
algor(w, a, z) {
    length = 0
    v = a
    T = set of all vertices
    while (v ¬= z) {
        T = T − {v}
        choose x ∈ T with minimum w(v, x)
        length = length + w(v, x)
        v = x
    }
    return length
}
```

8.4 A Shortest-Path Algorithm 最短路径算法

True or false? When a connected, weighted graph and vertices a and z are input to the following algorithm, it returns the length of a shortest path from a to z.

Algorithm 8.4.6

algor(w, a, z) {
  length $= 0$
  $v = a$
  $T =$ set of all vertices
  while $(v\neg = z)$ {
    $T = T - \{v\}$
    choose $x \in T$ with minimum $w(v, x)$
    length $=$ *length* $+ w(v, x)$
    $v = x$
  }
  return *length*
}

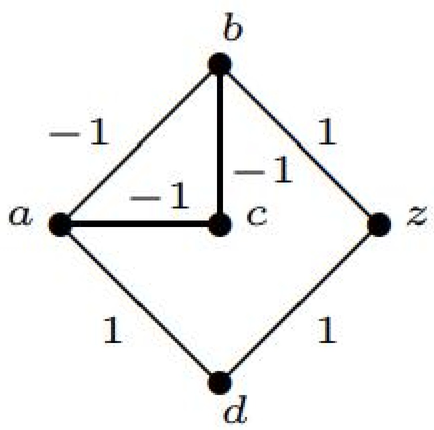


8.4 A Shortest-Path Algorithm 最短路径算法

True or false? Dijkstra’s shorest-path algorithm finds the length of a shortest path in a connected, weighted graph even if some weights are negative. If true, prove it; otherwise, provide a counterexample.

8.4 A Shortest-Path Algorithm 最短路径算法

True or false? Dijkstra's shortest-path algorithm finds the length of a shortest path in a connected, weighted graph even if some weights are negative. If true, prove it; otherwise, provide a counterexample.



8.5 Representations of Graphs 图的表示

adjacent & incident?

8.5 Representations of Graphs 图的表示

adjacent & incident？

- Two vertices are called **adjacent** if they are connected by an edge.
- A vertex and an edge are called **incident**, if the vertex is one of the two vertices the edge connects.

8.5 Representations of Graphs 图的表示

Adjacency Matrix (邻接矩阵)

adjacent & incident?

- Two vertices are called **adjacent** if they are connected by an edge.
- A vertex and an edge are called **incident**, if the vertex is one of the two vertices the edge connects.