例：称与其补图同构的简单无向图为自补图。证明每个自补图的阶能被4整除或被4除余数为1.

证明：设G为n阶完全无向图K_n的生成子图，K_n-G为G的补图，且G与K_n-G同构。

由于K_n的边数为n(n-1)/2，且G与K_n-G同构，因此G与K_n-G的边数均为n(n-1)/4。

由于n(n-1)/4为整数，设为k，即 n(n-1)/4 = k。得 4| n(n-1)。

下面证明 当4|n(n-1)时，4| n或 n= 4i+1，i, n∈N, n≥1。

由于n与n-1不能同为奇数，也不能同为偶数，因此必有4|n或4|n-1。因此结论成立。

主要内容

1. 图论的基本概念
2. 子图和图的运算
3. 路径、回路和连通图
4. 欧拉图和哈密顿图
5. 图的矩阵表示
6. 树、有向树和有序树

§7.3 路径、回路和连通性

路径、回路和连通性

目的：了解与路径、回路、连通性、分支、非循环图相关的基本概念；掌握求加权路径的算法、判一个图是否有回路、有有向回路、有半回路的过程；

重点：路径、回路、连通、分支等重要概念；求加权路径的算法；判回路、有向回路、半回路、循环图；

难点：几种判定方法及其原理。

路径的应用

- 无向图的结点和边分别表示城市和连接城市的双轨铁路。
- 从城市 v_0 到城市 v_n 的 路径： 由一个结点和边组成的序列来表示：

$$v_0\ e_1\ v_1\ e_2\ v_2\ \cdots\ v_{n-1}\ e_n\ v_n$$

其中，

- ✓ e_i（$1 \le i \le n$）表示连接城市的铁路；
- ✓ v_1，v_2，…，v_{n-1} 表示途经的城市。

路径

定义 7.3.1 设 $n \in N$，v_0，$v_1 \ldots$，v_n 是图 G 的结点，e_1，e_2，…，e_n是图G的边，并且v_{i-1} 和v_i 分别是 e_i 的起点和终点 ($i=1$，2，…，n)，则称序列 $v_0 e_1 v_1 e_2 \ldots v_{n-1} e_n v_n$ 为图 G 中从 v_0 至 v_n 的路径，n 称为该路径的长度。

(i) 如果 $v_0 = v_n$，则称该路径为闭的，否则称为开的。

(ii)如果 e_1，e_2，…，e_n 互不相同，则称该路径为简单的。

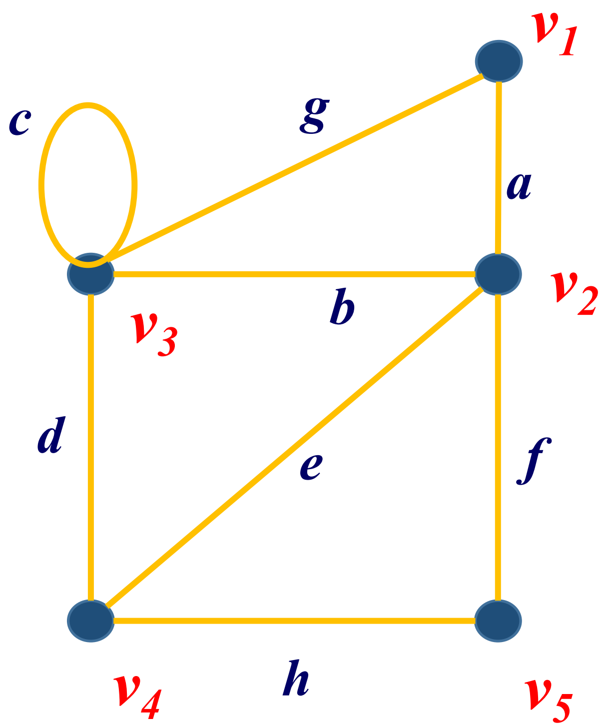
(iii) 如果v_0，v_1，…，v_n互不相同，则称该路径为基本的。

◆ 基本路径必为简单路径

路径（另一组术语）

- **链（chain or walk）**：顶点和边交错出现的序列称为链，在序列中边的前后两 个顶点正好是边的端点，序列的第一个顶点和最后一个顶点为链的端点，其余的点为内点。
- **迹（trail）**：边互不相同的链称为迹。即迹中无重边。
- **路（path）**：内部点互不相同的链称为路。即路中无重点。

例：无向图G



(1) $v_2\, b\, v_3\, d\, v_4\, e\, v_2\, b\, v_3$ 路径

(2) $v_2\, b\, v_3\, c\, v_3\, d\, v_4$ 简单路径

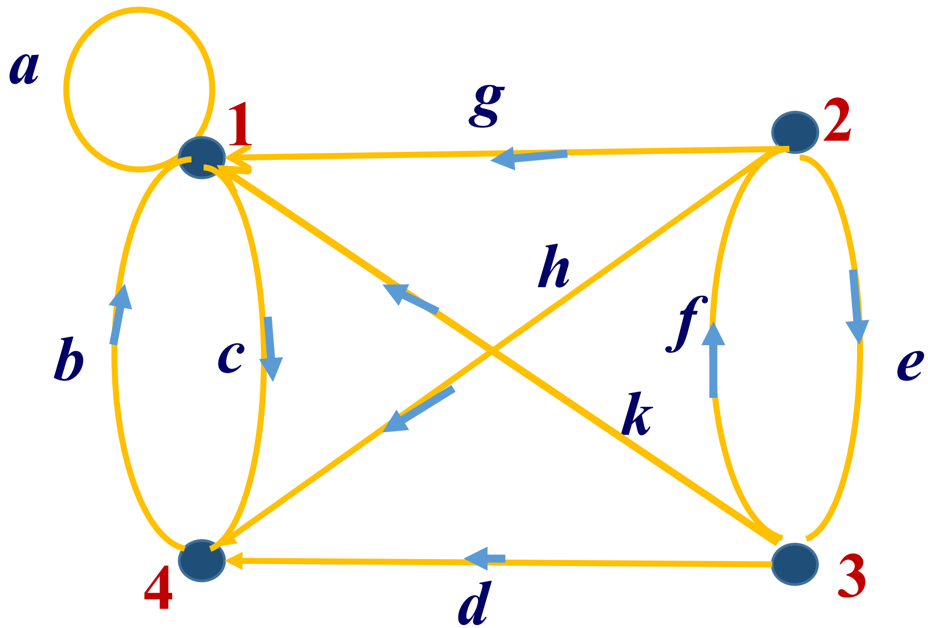
(3) $v_3\, c\, v_3\, c\, v_3$ 闭路径

(4) $v_1\, g\, v_3\, c\, v_3$变为一个基本路径？

基本路径： $v_1\, g\, v_3$

直观结论：从路径中去掉闭路径，能够得到基础路径。

例：有向图G



(1) 1c4b1c4 路径

(2) 1a1c4 简单路径

(3) 1c4 基本路径

路径：一些基本性质

- 当 n = 0，路径 v_0 的长度为 0，基本路径。
  - ✓ 任何结点到自身总存在路径。
- v到 v′ 存在路径　⇒　　v′ 到 v 存在路径？

  （无向图 √　　　　有向图 ╳ ）

定理7.3.1 设图 $G = \langle V, E, \Psi \rangle$，$v, v' \in V$。如果存在从 v 至 v'的路径，则存在从 v 至 v'的基本路径。

证明：（第二归纳法）假设从 v 至 v'的路径长度为n，对n进行归纳证明。

(1) 当n=0时，此时v=v'，路径长度为0，是基本路径；

(2) 假设对每个自然数k，当 $0 \le k < n$时，若存在长度为k的从v到v'的路径，则一定存在从存在从v 至 v'的基本路径。下面证明结论对k=n时成立。

假设路径 $\rho = v_0 e_1 v_1 \ldots v_{n-1} e_n v_n$是从 v 至 v'的路径，且不是基本路径， 其中 $v_0 = v$, $v_n = v'$，则必有 i 和 j 使 $0 \le i < j \le n$ 且 $v_i = v_j$。

故$\rho' = v_0 e_1 v_1 \ldots v_i e_{j+1} v_{j+1} \ldots v_{n-1} e_n v_n$ 是从 v 至 v'的长度为$n-(j-i)$的路径。

根据归纳假设，必存在从 v 至 v'的基本路径。

定理7.3.2 n 阶图中的基本路径的长度小于 n。

（因为基本路径中的结点互不相同，即最多仅含 n 个结点，所以所经过的边数必定小于 n。）

可达

定义 7.3.2 设图 $G=\langle V, E, \Psi\rangle$，$v_1$，$v_2\in V$。若存在从 v_1 至 v_2 的路径，则称在G中从v_1可达v_2，或从v_1到v_2可达；否则称在G 中从 v_1 不可达 v_2 或从v_1到v_2不可达。

对于图 G 的结点 v，用 R (v) 表示从 v 可达的全体结点的集合。

- 在无向图中，若从v_1到v_2可达，则从v_2到v_1必可达
- 在有向图中，从v_1到v_2可达不能保证从v_2到v_1必可达

定理7.3.3 设图 $G=\langle V, E, \Psi\rangle$，$v_1$，$v_2\in V$。从 v_1 可达 v_2 当且仅当存在从 v_1 至 v_2 的基本路径。

距离

定义7.3.3 设图 G ＝〈V， E，Ψ〉， v_1， $v_2 \in V$。
i) 若从 v_1 可达 v_2，则称从 v_1 至 v_2 的路径中长度最短者为从 v_1 至 v_2 的 测地线，并称该测地线的长度为从 v_1 至 v_2 的距离，记作 $d(v_1, v_2)$。
ii) 若从 v_1 不可达 v_2，则称 v_1 至 v_2 的距离 $d(v_1, v_2)$ 为 ∞。
并且规定:

$$\infty + \infty = \infty;\quad \forall n \in N,\quad \infty > n,\quad n + \infty = \infty + n = \infty。$$

定义7.3.4 图 G ＝〈V， E，Ψ〉的直径定义为

$$\max_{v,v' \in V} d(v, v')$$

例

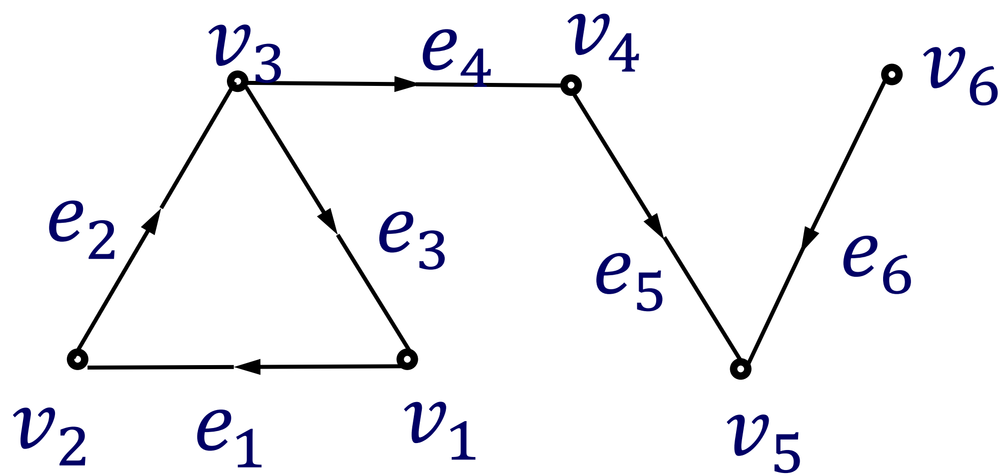


图 7.3-3 图中节点的可达性

R (v_1) = R (v_2) = R (v_3) ={v1, v2, v3, v4, v5}

R (v_4) = {v4, v5}

R (v_5) = {v5}

R (v_6) = {v_5, v_6}

d (v_1, v_2) = 1

d (v_2, v_1) = 2

d (v_5, v_6) = ∞

看作无向图时，直径为4

加权图

定义7.3.5 设图 G ＝〈V, E, Ψ〉。若 W : E → R^+（R^+是正实数集）， 则称〈G，W〉为加权图。

i) 若 e∈E，称 W(e) 为边 e 的加权长度。

ii) 路径中所有边的加权长度之和称为该路径的加权长度。

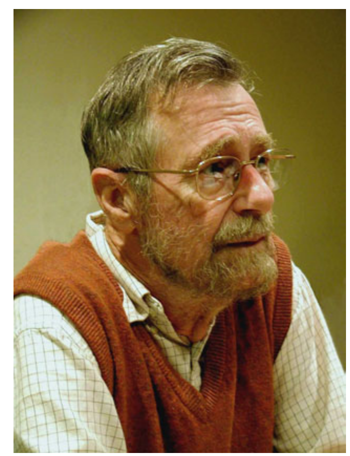
iii)从结点v至结点 v′的路径中, 加权长度最小的称为从 v 至 v′的 最短路径。

iv) 若从v可达v′，则称从v 至 v′的最短路径的加权长度为从v至v′的加权距离。

v) 若从 v 不可达 v′，则称从 v 至 v′的加权距离为 ∞。

迪克斯特拉（Dijkstra）

- 艾兹格·**W**·迪克斯特拉（**Edsger Wybe Dijkstra**，**1930**年**5**月**11**日~**2002**年**8**月**6**日）
- 荷兰人。 计算机科学家，毕业就职于荷兰**Leiden**大学，早年钻研物理及数学，而后转为计算学。
- **1972**年获得图灵奖

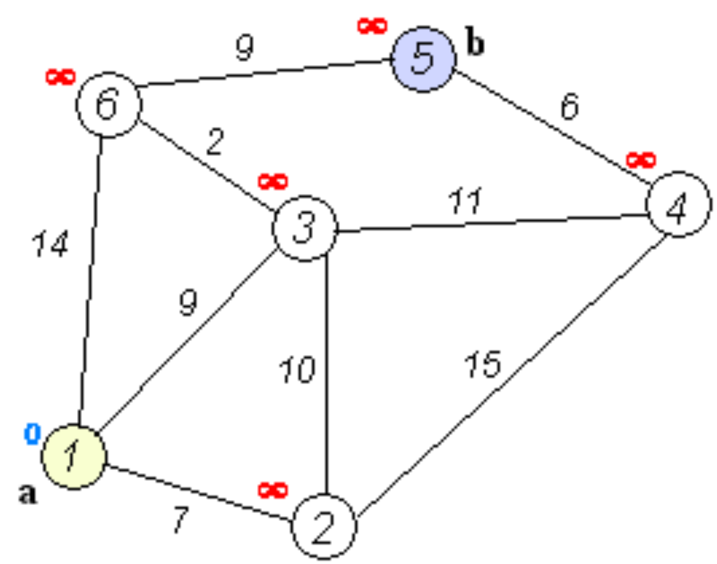


戴克斯特拉（Dijkstra）

- 1 提出“goto有害论”；
- 2 提出信号量和pv原语;
- 3 解决了“哲学家聚餐”问题;
- 4 Dijkstra最短路径算法和银行家算法的创造者;
- 5 第一个Algol 60编译器的设计者和实现者;
- 6 THE操作系统的设计者和开发者;
- 与D. E. Knuth并称为我们这个时代最伟大的计算机科学家的人。
- 与癌症抗争多年，于2002年8月6日在荷兰Nuenen自己的家中去世，享年72岁

迪克斯特拉（Dijkstra）算法

- **1959年, 最短路径算法**
- 应用产物
  - ✓ 单源路径计算（**Single-source shortest paths problem**）
  - ✓（连通）有权（有向）图
  - ✓边的权值非负数
- 贪心算法**(Greedy Algorithm)**



迪克斯特拉（Dijkstra）算法

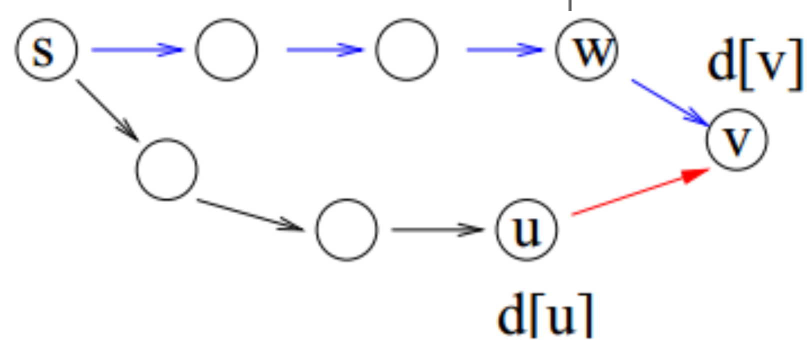
Input: A graph $\boldsymbol{G}$, a matrix $\boldsymbol{w}$ representing the weights between vertices in G, source vertex $\boldsymbol{s}$

Output: None

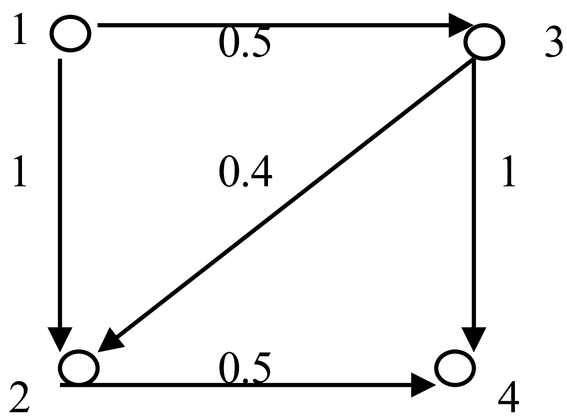
for $u \in V$ **do**
    $d[u] \leftarrow \infty, color[u] \leftarrow$ WHITE;// Initialize
end
$d[s] \leftarrow 0$;
$pred[s] \leftarrow$ NULL;
$Q \leftarrow$queue with all vertices;
while *Non-Empty(Q)* **do**
    // Process all vertices
    $u \leftarrow$Extract-Min(Q);// Find new vertex
    for $v \in Adj[u]$ **do**
        if $d[u] + w(u, v) < d[v]$ **then**
            // If estimate improves
            $d[v] \leftarrow d[u] + w(u, v)$;// relax
            Decrease-Key($Q, v, d[v]$);
            $pred[v] \leftarrow u$;
        end
    end
    $color[u] \leftarrow$BLACK;
end

- **d[u]：结点u到源点s的最短距离**
- **Q：优先队列**

v.key=d[v]



例子（加权距离）



当前点 \ d[v] \ 结点	1	2	3	4
	0	∞	∞	∞
1	/	1	0.5	∞
3	/	0.9	/	1.5
2	/	/	/	1.4
4				

从 1 到 4 的加权距离为 1.4。

迪克斯特拉（Dijkstra）算法

Input: A graph $\boldsymbol{G}$, a matrix $\boldsymbol{w}$ representing the weights between vertices in G, source vertex $\boldsymbol{s}$

Output: None

for $u \in V$ **do**

  $d[u] \leftarrow \infty, color[u] \leftarrow$ WHITE;// Initialize

end

$d[s] \leftarrow 0$;

$pred[s] \leftarrow$ NULL;

$Q \leftarrow$queue with all vertices;

while *Non-Empty(Q)* **do**

  // Process all vertices

  $u \leftarrow$Extract-Min(Q);// Find new vertex

  for $v \in Adj[u]$ **do**

    if $d[u] + w(u,v) < d[v]$ **then**

      // If estimate improves

      $d[v] \leftarrow d[u] + w(u,v)$;// relax

      Decrease-Key($Q, v, d[v]$);

      $pred[v] \leftarrow u$;

    end

  end

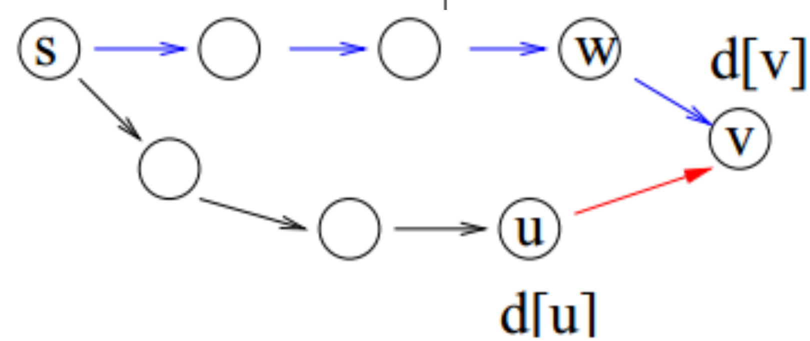
  $color[u] \leftarrow$BLACK;

end

怎么构造最短路径？

- **d[u]：结点u到源点s的最短距离**
- **Q：优先队列**

v.key=d[v]



最短路径算法扩展

- 放松最短路条件
  - 任意值，即可能存在负数，可能有圈
  - 任意两点之间的最短路？
- 其他算法
  - 任意权值、单源：Bellman-Ford
  - 任意权值、任意两点：Folyd-Warshall