

# 离散数学基础：图论

Fundamentals of Discrete Mathematics: Graph  
Theory

周晓聪([isszxc@zsu.edu.cn](mailto:isszxc@zsu.edu.cn))

中山大学计算机科学系, 广州 510275

2006 年 12 月 5 日

版权所有，翻印必究

# 目录

目录	i
<b>第一章 图的基本概念</b>	<b>1</b>
1.1 图的基本定义	1
1.2 道路与回路	4
1.3 图的连通性	5
1.4 欧拉图与哈密顿图	8
1.5 邻接矩阵与可达矩阵	9
<b>第二章 树的基本概念</b>	<b>11</b>
2.1 树的基本定义	11
2.2 生成树	13
2.3 根树	17
2.4 哈夫曼树	21
<b>第三章 路径问题</b>	<b>27</b>
3.1 最短路径	27
3.2 最小生成树	32
3.3 关键路径	34
<b>第四章 平面图与着色</b>	<b>39</b>
4.1 平面图及其性质	39
4.2 图的着色	42
<b>第五章 支配集、覆盖集、独立集与匹配</b>	<b>45</b>
5.1 支配集、点独立集和点覆盖集	45
5.2 边覆盖与匹配	47
5.3 二部图中的匹配	52
<b>参考文献</b>	<b>55</b>



# 第一章 图的基本概念

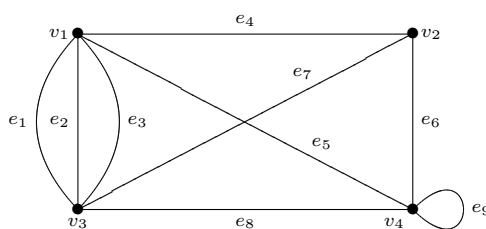
这一章介绍有关图论的一些基本概念,包括无向图(有向图)的定义、顶点与边之间的关系、顶点度数、握手定理、图的道路与回路,以及特殊的道路与回路,如欧拉图、哈密顿图等。

## 1.1 图的基本定义

**定义 1.1.1 (无向)图**(graph)是二元组 $G = (V, E)$ ,其中 $V \neq \emptyset$ 是图 $G$ 的**顶点**(vertex)集,其中的元素称为 $G$ 的**顶点**(vertex), $E$ 是图 $G$ 的**边**(edge)集,其中的元素称为 $G$ 的**边**(edge),且满足,对图 $G$ 的任意边 $e \in E$ ,都有且仅有两个顶点 $u, v \in V$ 与 $e$ 关联,称为 $e$ 的两个端点,通常将 $e$ 记为 $e = (u, v)$ 或 $e = (v, u)$ 。

对于边 $e = (u, v)$ ,这里 $u, v$ 没有顺序,因此边 $e = (u, v)$ 和 $e = (v, u)$ 是同一条边。我们只考虑有限图 $G = (V, E)$ ,也即其顶点集 $V$ 和边集 $E$ 都是有限集。上面的定义是说,在定义一个图的时候,要给出它的顶点集和边集,并说明每一条边的两个端点是那两个顶点。通常,我们可以对图作最为直观的理解,则画出这个图,并用 $V$ 和 $E$ 中的元素对这个图进行标记。

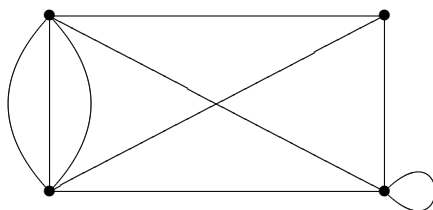
**例子 1.1.2** 下面是一个图的直观表示:



若将这个图用比较严格的数学形式给出来,则是:图 $G = (V, E)$ ,其中:

$$\begin{aligned} V &= \{v_1, v_2, v_3, v_4\} \\ E &= \{e_1 = (v_1, v_3), e_2 = (v_1, v_3), e_3 = (v_1, v_3), e_4 = (v_1, v_2), e_5 = (v_1, v_4), \\ &\quad e_6 = (v_2, v_4), e_7 = (v_2, v_3), e_8 = (v_3, v_4), e_9 = (v_4, v_4)\} \end{aligned}$$

有时为了简便,我们可能在给出图的直观形式时,没有对其顶点和边进行标记,例如下面是上面的图不带标记的形式:

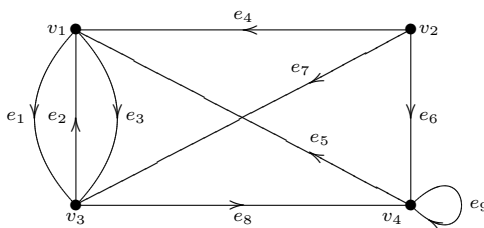


这通常是因为我们这时只关注整个图的性质，不关心图的顶点或边的性质。如果需要关心某些顶点或边的性质时，我们也可能将某些顶点和边标记，而不标记其他顶点和边。

**定义 1.1.3 (有向)图**(directed graph)是二元组  $G = (V, E)$ ，其中  $V \neq \emptyset$  是图  $G$  的**顶点**(vertex)集，其中的元素称为  $G$  的**顶点**(vertex)， $E$  是图  $G$  的**边**(edge)集，其中的元素称为  $G$  的**边**(edge)，且满足，对图  $G$  的任意边  $e \in E$ ，都有一个顶点  $u \in V$  是  $e$  的**起点**，同时有一个顶点  $v \in V$  是  $e$  的**终点**，通常将  $e$  记为  $e = \langle u, v \rangle$ 。

类似，我们也只考虑有限有向图，有向图与无向图的区别，在于有向图中与每一条边相关联的两个顶点区分为起点和终点。

**例子 1.1.4** 下面是一个有向图的直观表示：



若将这个有向图用比较严格的数学形式给出来，则是：图  $G = (V, E)$ ，其中：

$$V = \{ v_1, v_2, v_3, v_4 \}$$

$$E = \{ e_1 = \langle v_1, v_3 \rangle, e_2 = \langle v_3, v_1 \rangle, e_3 = \langle v_1, v_3 \rangle, e_4 = \langle v_2, v_1 \rangle, e_5 = \langle v_4, v_1 \rangle, \\ e_6 = \langle v_2, v_4 \rangle, e_7 = \langle v_2, v_3 \rangle, e_8 = \langle v_3, v_4 \rangle, e_9 = \langle v_4, v_4 \rangle \}$$

下面如果只说图，则是指无向图。我们下面继续给出一些简单的术语定义：

**定义 1.1.5** 对于无向图或有向图  $G = (V, E)$ ，

- (1) 对于边  $e \in E$ ，若  $e = (u, v)$  (或  $e = \langle u, v \rangle$ )，则称边  $e$  与顶点  $u, v$  **关联**；
- (2) 对于顶点  $u, v \in V$ ，如果存在边  $e \in E$ ，使得  $e = (u, v)$  (或  $e = \langle u, v \rangle$ )，则称  $u$  与  $v$  **邻接**，或说  $u$  和  $v$  是**相邻的顶点**；
- (3) 对于边  $e_1, e_2 \in E$ ，若  $e_1$  和  $e_2$  关联共同的顶点，则称  $e_1$  和  $e_2$  **邻接**，或说  $e_1$  和  $e_2$  是**相邻的边**。

这些术语给出了边与边、顶点与顶点之间的邻接 (或说相邻) 关系，以及顶点与边之间的关联关系。实际上，我们无需纠缠于这些术语的严格定义，直观理解即可，上述定义只是如果在有歧义的情况下，给出一个标准的说法而已。

**定义 1.1.6** 对于无向图或有向图  $G = (V, E)$ ，

- (1) 若边  $e = (v, v)$  (或  $e = \langle v, v \rangle$ )，即它关联同一个顶点，则称  $e$  是**环**；

(2) 若边  $e = (u, v)$  (或  $e = \langle u, v \rangle$ ) 且  $e' = (u, v)$  (或  $e' = \langle u, v \rangle$ ), 即  $e$  和  $e'$  都关联相同的顶点, 则称  $e$  和  $e'$  是**重边**。

注意, 在有向图中, 边  $e = \langle u, v \rangle$  和  $e' = \langle v, u \rangle$  并不是重边。下面我们定义图的一些特殊类别:

**定义 1.1.7** 对于无向图  $G = (V, E)$ , 如果  $G$  既没有环也没有重边, 则称  $G$  是**简单图**, 在后面我们提到图时, 如果没有特别声明, 都是指简单图。对于有向图  $G = (V, E)$ , 如果  $G$  既没有环也没有重边, 则称为**有向简单图**, 同样, 如果没有特别声明, 我们所讲的有向图, 都是有向简单图。

**定义 1.1.8** 对于无向图  $G = (V, E)$ , 如果对  $V$  的任意两个顶点  $u, v$ , 都存在边  $e = (u, v)$  或  $e = (v, u)$ , 则称  $G$  是**完全图**。具有  $n$  个顶点的完全图记为  $K_n$ 。对于有向图  $G = (V, E)$ , 如果对  $V$  的任意两个顶点  $u, v$ , 都存在边  $e = \langle u, v \rangle$  及  $e' = \langle v, u \rangle$ , 则称  $G$  是**有向完全图**。

利用子集的概念可定义子图的概念:

**定义 1.1.9** 给定图  $G = (V, E)$ , 如果图  $G' = (V', E')$  满足  $V' \subseteq V$  且  $E' \subseteq E$ , 则称  $G'$  是  $G$  的**子图**。特别地, 如果  $V = V'$ , 则称  $G'$  是  $G$  的**生成子图**。对于  $G$  的顶点集  $V$  的任意子集  $V' \subseteq V$ , 取  $E' = \{e \in E \mid e = (u, v) \text{ 且 } u \in V' \text{ 且 } v \in V'\}$ , 也即  $E'$  是  $E$  中那些两个端点都在  $V'$  中边构成的集合, 则称  $G' = (V', E')$  是  $G$  的由  $V$  **导出的子图**。

上述定义是针对无向图定义的, 读者不难对有向图也定义子图、生成子图及导出子图的概念。可在图  $G = (V, E)$  上定义一些操作, 而得到它的一些子图:

**定义 1.1.10** 对于图  $G = (V, E)$ ,

(1) **删除顶点集**: 设  $V' \subseteq V$ , 则  $G - V' = (V - V', E')$  是  $G$  的子图,  $G - V'$  的顶点集是  $V - V'$ , 而  $E' = \{e \in E \mid e = (u, v) \text{ 且 } u \notin V' \text{ 且 } v \notin V'\}$ , 也即  $E'$  是  $E$  中那些两个端点都不在  $V'$  中的边构成的集合;

(2) **删除边集**: 设  $E' \subseteq E$ , 则  $G - E' = (V, E - E')$  是  $G$  的子图,  $G - E'$  的顶点集仍是  $V$ , 而边集是  $E - E'$ 。

(3) **删除子图**: 设  $G' = (V', E')$  是  $G$  的子图, 则  $G - G' = (V, E - E')$ 。

注意,  $G - V'$  不仅要删除  $G$  中那些在  $V'$  中的顶点, 而且要将这些顶点相关联的边也删除, 因为边是不难单独存在的, 它一定要与两个顶点关联, 而  $G - E'$  只删除  $G$  中那些在  $E'$  中的边, 不改变  $G$  的顶点。特别地, 当  $V' = \{v\}$ , 只有一个顶点时, 将  $G - V'$  记为  $G - v$ , 类似地有  $G - e$ 。对于有向图, 也可类似地定义有向图删除顶点集、边集及子图的操作。

**定义 1.1.11** 对于无向图  $G = (V, E)$ , 设  $|V| = n$ , 即  $G$  有  $n$  个顶点, 则称  $\overline{G} = K_n - G$  为  $G$  的**补图**。

下面我们定义顶点的度数:

**定义 1.1.12** 对于无向图  $G = (V, E)$ , 定义顶点  $v \in V$  的**度数**  $d(v) = |\{e \in E \mid e = (u, v) \text{ 或 } e = (v, u)\}|$ , 也即  $d(v)$  是  $v$  所关联的边数, 但若  $e = (v, v)$  是环, 则边  $e$  要计算两次。

对于有向图  $G = (V, E)$ , 定义顶点  $v \in V$  的**入度**  $d^-(v) = |\{e \in E \mid e = \langle u, v \rangle\}|$ , 则  $d^+v$  是以  $v$  为终点的边数; 定义顶点  $v \in V$  的**出度**  $d^+(v) = |\{e \in E \mid e = \langle v, u \rangle\}|$ , 则  $d^+v$  是以  $v$  为起点的边数; 定义顶点  $v \in V$  的**度数**  $d(v) = d^-(v) + d^+(v)$ 。

很简单地,可证明下面的定理:

**定理 1.1.13 握手定理:** 对于任意的无向图  $G = (V, E)$ , 有:  $\sum_{v \in V} d(v) = 2|E|$ ; 对于任意的有向图  $G = (V, E)$ , 有:  $\sum_{v \in V} d^-(v) = \sum_{v \in V} d^+(v) = |E|$ 。

**例子 1.1.14** 很显然, 对于  $n$  个顶点的完全图  $K_n$ , 因为每个顶点都与其它  $n-1$  个顶点有边, 因此每个顶点的度数都是  $n-1$ , 从而  $\sum d(v) = n(n-1)$ , 因此  $K_n$  的边有  $\frac{n(n-1)}{2}$  条。

**定义 1.1.15** 给定图  $G = (V, E)$ , 定义图  $G$  的**最小度**  $\delta(G) = \min\{d(v) \mid v \in V\}$ , 即  $\delta(G)$  是度数最小的顶点的度数; 定义图  $G$  的**最大度**  $\Delta(G) = \max\{d(v) \mid v \in V\}$ , 即  $\Delta(G)$  是度数最大的顶点的度数。

**引理 1.1.16** 若  $G$  是简单图, 则  $\Delta(G) \leq n-1$ 。

**定义 1.1.17** 若图  $G = (V, E)$  任意顶点的度数都等于  $k$ , 则称  $G$  为 **$k$ 正则图**。

最后, 我们定义图的同构这个概念:

**定义 1.1.18** 对于图  $G = (V, E)$  和  $G' = (V', E')$ , 如果存在双函数  $f: V \rightarrow V'$ , 满足: 对  $E$  的任意边  $e = (u, v)$ ,  $E'$  存在惟一的边  $e' = (f(u), f(v))$ , 而且对  $E'$  的任意边  $e' = (u', v')$ ,  $E$  存在惟一的边  $e = (f^{-1}(u'), f^{-1}(v'))$ , 则称  $G$  和  $G'$  同构, 记为  $G \sim G'$ 。

**定义 1.1.19** 如果图  $G = (V, E)$  与它的补图  $\bar{G}$  同构, 则称  $G$  是**自补图**。

由于目前尚没有发现两个图同构的充分必要条件, 也没有发现判断两个图是否同构的有效算法, 因此我们只能直观地理解两个图同构的含义, 即在不考虑图的标记的情况, 这两个图在适当移动顶点和边之后有相同的图形表示, 或者说在不考虑图的标记的情况下, 这两个图的顶点和边之间的关联情况完全一致。

## 1.2 道路与回路

**定义 1.2.1** 给定**无向图**  $G = (V, E)$ ,  $G$  中的一条**道路** (或说路、通路、链等)  $\Gamma = v_0 e_1 v_1 e_2 \cdots e_n v_n$  是  $G$  中顶点和边的序列, 且满足, 对任意的  $1 \leq i \leq n$  有  $e_i = (v_{i-1}, v_i)$  (或  $e_i = (v_i, v_{i-1})$ ), 这时称  $v_0$  和  $v_n$  是  $\Gamma$  的两个端点,  $n$  是  $\Gamma$  的长度。如果还满足  $v_0 = v_n$ , 则称  $\Gamma$  是**回路** (或说圈)。

**定义 1.2.2** 给定**有向图**  $G = (V, E)$ ,  $G$  中的一条**有向道路**  $\Gamma = v_0 e_1 v_1 e_2 \cdots e_n v_n$  是  $G$  中顶点和边的序列, 且满足, 对任意的  $1 \leq i \leq n$  有  $e_i = \langle v_{i-1}, v_i \rangle$ , 这时称  $v_0$  是  $\Gamma$  的起点, 而  $v_n$  是  $\Gamma$  的终点,  $n$  是  $\Gamma$  的长度。如果还满足  $v_0 = v_n$ , 则称  $\Gamma$  是**有向回路**。

**定义 1.2.3** 给定**无向图**  $G = (V, E)$ ,  $\Gamma$  是  $G$  的一条道路 (回路), 如果  $\Gamma$  中**不存在重复的边**, 则称  $\Gamma$  是**简单道路 (简单回路)**; 如果除  $\Gamma$  的两个端点可能相同之外没有重复的顶点, 则称  $\Gamma$  是**初级道路 (初级回路)**, 初级道路也称为**路径**。类似地可定义有向简单道路、有向简单回路、有向初级道路、有向初级回路。

后面为方便起见, 我们用记号  $v \in \Gamma$  表示顶点  $v$  出现在  $\Gamma$  的顶点与边的序列中, 类似地用  $e \in \Gamma$  表示边  $e$  在  $\Gamma$  的顶点与边序列中。



对于道路 $\Gamma = v_0e_1v_1e_2\cdots e_nv_n$ ，由于边的两个端点在图中是固定的，因此我们可以只用边的序列 $e_1e_2\cdots e_n$ 表示道路 $\Gamma$ 。进一步，在简单图中，由于顶点之间最多有一条边，因此可只用顶点的序列 $v_0v_1\cdots v_n$ 表示道路 $\Gamma$ 。很容易证明图中的道路（回路）的如下性质：

**引理 1.2.4** 给定图 $G = (V, E)$ 有 $n$ 个顶点，若存在以 $u$ 和 $v$ 为端点的道路，则存在以 $u$ 和 $v$ 为端点的长度小于等于 $n - 1$ 的初级道路。

**推论 1.2.5** 给定图 $G = (V, E)$ 有 $n$ 个顶点，若存在从 $u$ 到其自身的回路，则存在从 $u$ 到其自身的长度小于等于 $n$ 的初级回路。

在图论的证明中，很多时候需要利用极长道路的概念：

**定义 1.2.6** 给定无向图 $G = (V, E)$ ，称 $G$ 中的一条简单道路 $\Gamma = v_0e_1v_1e_2\cdots e_nv_n$ 是**极长道路**，如果不存在顶点 $v \in V$ ， $v \neq v_i (0 \leq i \leq n)$ 使得 $v$ 与 $\Gamma$ 的端点 $v_0$ 或 $v_n$ 邻接。进一步，若极长道路 $\Gamma$ 又是初级道路，则称为**极长初级道路**。

也就是说，**极长道路 $\Gamma$ 的两个端点只与在 $\Gamma$ 中的顶点相邻**。注意，只有在简单道路（则不重复边）的意义下才有所谓的极长道路。如果允许重复边，则任意一条边 $e = (u, v)$ 都可看作是一条无穷长的道路 $\Gamma = ueveue\cdots$ 。

**例子 1.2.7** 若简单图 $G$ 的最小度 $\delta(G) \geq 2$ ，则 $G$ 必含有回路。

**证明** 设 $\Gamma = v_0e_1v_1e_2\cdots e_nv_n$ 是 $G$ 中一条极长道路，由于 $d(v_0) \geq 2$ ，因此除 $v_1$ 之外，还有 $v_i \in \Gamma$ 使得 $v_0$ 与 $v_i$ 相邻，则 $\Gamma' = v_0v_1\cdots v_iv_0$ 是回路。□

**例子 1.2.8** 若 $\Gamma$ 是简单图 $G = (V, E)$ 的中含顶点数大于3的初级回路，如果存在边 $e = (u, v) \in E$ 满足 $u, v \in \Gamma$ 而 $e \notin \Gamma$ ，则称 $e$ 是 $\Gamma$ 的弦，而 $\Gamma$ 是**带弦回路**。证明若简单图 $G = (V, E)$ 的最小度 $\delta(G) \geq 3$ ，则 $G$ 必存在带弦的回路。

**证明** 设 $\Gamma = v_0e_1v_1e_2\cdots e_nv_n$ 是 $G$ 中一条极长初级道路，由于 $d(v_0) \geq 3$ ，因此除 $v_1$ 之外，还有 $v_i, v_j \in \Gamma$ 使得 $v_0$ 与 $v_i$ 和 $v_j$ 相邻，不妨假设 $i < j$ ，则 $\Gamma' = v_0v_1\cdots v_iv_i\cdots v_jv_0$ 是一条初级回路，而边 $e = (v_0, v_i)$ 是该回路的弦。□

**例子 1.2.9** 给定 $G = (V, E)$ 是简单图， $|V| = n \geq 4$ ， $|E| = m \geq 2n - 3$ ，证明 $G$ 有带弦回路。

**证明** 用数学归纳法，不难看到当 $n = 4$ 时，命题成立；假设命题对于 $n = k$ 时成立，也即对任意的图，如果其顶点数是 $k$ ，而边数大于等于 $2k - 3$ ，则它含有带弦回路。考虑 $n = k + 1$ ，即考虑对于顶点数是 $k + 1$ ，边数大于等于 $2(k + 1) - 3$ 的图 $G$ 。设 $\Gamma$ 是 $G$ 的一条极长初级道路， $v$ 是 $\Gamma$ 的一个端点，如果 $d(v) \geq 3$ ，则由上面的例子可得 $G$ 有带弦回路，否则（即 $d(v) < 3$ ）不难看到 $G - v$ 是有 $k$ 个顶点，边数大于等于 $2(k + 1) - 3 - 2 = 2k - 3$ 的图，根据归纳假设， $G - v$ 含有带弦回路，那么 $G$ 也必含有带弦回路。□

## 1.3 图的连通性

**定义 1.3.1** 给定无向图 $G = (V, E)$ ，顶点 $u, v \in V$ ，称 $u$ 和 $v$ 之间有道路，或称 $u$ 和 $v$ 是**可达**的，如果 $G$ 存在以 $u$ 和 $v$ 为端点的道路。进一步，我们约定 $u$ 和 $u$ 自己是可达的。

在 $V$ 上定义关系 $\leftrightarrow \subseteq V \times V$ ,  $u \leftrightarrow v$ 当且仅当 $u$ 和 $v$ 是可达的, 则 $\leftrightarrow$ 是等价关系, 称 $V$ 关于 $\leftrightarrow$ 的等价类 $[u]_{\leftrightarrow} \subseteq V$ 导出的子图为 $G$ 的**连通分支**, 我们将 $G$ 的连通分支数记为 $p(G)$ 。若 $p(G) = 1$ , 则称 $G$ 是**连通图**。

**引理 1.3.2** 给定简单图 $G = (V, E)$ ,  $|V| = n, |E| = m$ , 若 $G$ 是连通的, 则 $m \geq n - 1$ 。

**证明** 使用第二数学归纳法证明: 当 $n = 1$ 时, 显然 $m \leq 0$ , 类似地, 当 $n = 2$ 时, 要使得 $G$ 连通至少要有一条边。设 $n < k$ 时成立, 即任意少于 $k$ 个顶点的简单连通图的边数大于等于顶点数减一。

考虑任意含有 $k$ 个顶点的简单连通图 $G$ 。设 $v$ 是 $G$ 的任意顶点, 记 $p(G - v) = j$ ,  $G - v$ 的每个连通分支的顶点数分别是 $n_1, n_2, \dots, n_j$ , 显然对任意的 $1 \leq i \leq j$ 有 $1 \leq n_i < k$ , 因此 $G - v$ 的每个连通分支都是少于 $k$ 个顶点的连通简单图, 根据归纳假设,  $G - v$ 至少含有

$$\sum_{i=1}^j n_i - 1 = k - 1 - j$$

条边。而为保证 $G$ 是连通图,  $v$ 与 $G - v$ 的每个连通分支都至少有一条边相连, 因此 $G$ 至少含有 $k - 1 - j + j = k - 1$ 条边。  $\square$

这个引理给出了连通简单图的边数的一个下界, 也即使得简单图连通的在边数方面的一个必要条件。显然简单图的边数的上界在完全图达到, 利用这个上界, 我们可以证明使得简单图连通的一个边数方面的充分条件:

**例子 1.3.3** 给定简单图 $G = (V, E)$ ,  $|V| = n, |E| = m$ , 若 $m > \frac{1}{2}(n-1)(n-2)$ , 则 $G$ 连通。

**证明** 反证法: 若 $G$ 不连通, 则 $k = p(G) \geq 2$ , 设其 $k$ 个连通分支的顶点数分别为 $n_1, n_2, \dots, n_k$ , 则对任意的 $1 \leq i \leq k$ 有 $1 \leq n_i \leq n - 1$ , 且 $\sum_{i=1}^k n_i = n$ 。当 $G$ 的每个连通分支都是完全图时边数最大, 因此 $G$ 最多有

$$\sum_{i=1}^k \frac{n_i(n_i - 1)}{2} \leq \sum_{i=1}^k \frac{(n-1)(n_i - 1)}{2} = \frac{n-1}{2} \sum_{i=1}^k (n_i - 1) = \frac{(n-1)(n-k)}{2}$$

若 $k \geq 2$ , 则与 $m > \frac{1}{2}(n-1)(n-2)$ 矛盾, 因此 $G$ 只有一个连通分支, 即 $G$ 是连通图。  $\square$

下面从顶点度数的角度给出了简单图连通的一个充分条件:

**引理 1.3.4** 给定简单图 $G = (V, E)$ ,  $|V| = n$ , 若对任意的两个顶点 $u, v \in V$ 都有 $d(u) + d(v) \geq n - 1$ , 则 $G$ 连通。

**证明** 反证法: 若 $G$ 不连通, 则至少有两个连通分支, 从其中两个连通分支中分别取顶点 $u$ 和 $v$ , 设这两个连通分支的顶点数分别为 $n_u$ 和 $n_v$ , 则 $d(u) < n_u$ 且 $d(v) < n_v$ , 从而 $d(u) + d(v) < n_u + n_v \leq n$ , 这与 $d(u) + d(v) \geq n - 1$ 矛盾, 因此 $G$ 是连通图。  $\square$

有向图的连通性可类似定义:

**定义 1.3.5** 给定有向图 $G = (V, E)$ , 对 $G$ 的两个顶点 $u, v \in V$ , 若存在以 $u$ 为起点,  $v$ 为终点的有向道路, 则称 $u$ 可达 $v$ , 记为 $u \rightarrow v$ 。进一步记 $u \leftrightarrow v$ 为 $u \rightarrow v$ 且 $v \rightarrow u$ 。约定总有 $u \leftrightarrow u$ 。

若对 $G$ 的任意两个顶点 $u, v$ 都有 $u \leftrightarrow v$ , 则称 $G$ 是**强连通**的; 若有 $u \rightarrow v$ 或 $v \rightarrow u$ 之一成立, 则称 $G$ 是**单向连通**的; 若不考虑有向图 $G$ 中边的方向时是(无向)连通图, 则称 $G$ 是**弱连通**的。

有些连通图当删除若干个顶点或若干条边之后就不连通了,从某种意义上说,能够删除多少顶点或边使得图不连通,从另一个角度刻画了图的连通程度。

**定义 1.3.6** 给定无向图  $G = (V, E)$  (不一定是简单图),  $V' \subset V$  是顶点集  $V$  的真子集,若  $V'$  满足: (i)  $p(G - V') > p(G)$ , 即从  $G$  中删除  $V'$  中的顶点 (及其相关联的边) 之后的图的连通分支数比  $G$  大; (ii) 对  $V'$  的真子集  $V'' \subset V'$ , 都有  $p(G - V'') = p(G - V')$ , 即只删除  $V'$  中某些顶点 (及关联的边) 不会再增加图的连通分支数; 则称  $V'$  是图的 **点割集**。若点割集  $V' = \{v\}$  只有一个顶点, 则称  $v$  是图  $G$  的 **割点**。

**定义 1.3.7** 给定无向图  $G = (V, E)$  (不一定是简单图),  $|V| = n$ , 设  $G$  连通且不含完全图  $K_n$  为子图, 称:

$$\kappa(G) = \min\{|V'| \mid V' \text{ 是 } G \text{ 的点割集}\}$$

为  $G$  的 **点连通度**, 简称 **连通度**。规定完全图  $K_n$  的点连通度为  $n - 1$ , 非连通图的点连通度为 0。若  $\kappa(G) \geq k$ , 则称图  $G$  是  $k$  连通图。

实际上, 点连通度就是顶点数最少的点割集的顶点数目, 显然若图有割点的话, 则其点连通图就是 1。若  $G$  是  $k$  连通图, 则从  $G$  中任意删除  $k$  个顶点之后, 图仍然是连通的。也可从边的角度刻画图的连通程度:

**定义 1.3.8** 给定无向图  $G = (V, E)$  (不一定是简单图),  $E' \subset E$  是边集  $E$  的真子集, 若  $E'$  满足: (i)  $p(G - E') > p(G)$ , 即从  $G$  中删除  $E'$  中的边之后的图的连通分支数比  $G$  大; (ii) 对  $E'$  的真子集  $E'' \subset E'$ , 都有  $p(G - E'') = p(G - E')$ , 即只删除  $E'$  中某些边不会再增加图的连通分支数; 则称  $E'$  是图的 **边割集**。若边割集  $E' = \{e\}$  只有一条边, 则称  $e$  是图  $G$  的 **割边**, 或称为 **桥**。

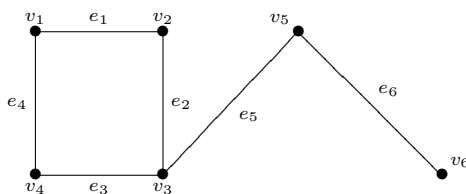
**定义 1.3.9** 给定无向图  $G = (V, E)$  (不一定是简单图),  $|V| = n$ , 设  $G$  连通, 称:

$$\lambda(G) = \min\{|E'| \mid E' \text{ 是 } G \text{ 的边割集}\}$$

为  $G$  的 **边连通度**, 简称 **连通度**。规定非连通图的点连通度为 0。若  $\lambda(G) \geq r$ , 则称图  $G$  是  $r$  边连通图。

类似地, 若  $G$  是  $r$  边连通图, 则从  $G$  中任意删除  $r$  条边之后, 它仍然是连通的。

**例子 1.3.10** 考虑下面的图:



显然  $\{v_2, v_4\}, \{v_3\}, \{v_5\}$  都是点割集, 其中  $v_3$  和  $v_5$  都是割点, 注意  $v_1$  和  $v_6$  都不在任何点割集中, 因此该图的点连通度是 1。

另一方面,  $\{e_6\}, \{e_5\}, \{e_2, e_3\}, \{e_1, e_2\}, \{e_3, e_4\}, \{e_1, e_3\}, \{e_2, e_4\}$  都是边割集, 其中  $e_6$  和  $e_5$  都是桥, 显然该图的边连通度是 1。

由于删除顶点要同时删除其关联的边, 因此不难证明如下定理:

**定理 1.3.11** 设 $G$ 是连通图, 则:  $\kappa(G) \leq \lambda(G) \leq \delta(G)$ 。

**证明** 设 $v$ 是 $G$ 中度数最少的顶点, 删除与 $v$ 相关联的边之后, 图就不连通, 因此 $\lambda(G) \leq \delta(G)$ 。下面再证 $\kappa(G) \leq \lambda(G)$ 。设 $E' = \{e_1, e_2, \dots, e_\lambda\}$ 是 $G$ 的一个最小边割集, 则 $G - E'$ 至少有两个不连通的顶点集 $V_1$ 和 $V_2$ 。这时再考虑图 $G$ , 若将 $V_1$ 中与 $e_1, e_2, \dots, e_{\lambda-1}$ 相关联的至多 $\lambda - 1$ 个顶点删除 (注意这也删除了边 $e_1, e_2, \dots, e_{\lambda-1}$ ), 又将 $V_2$ 中与 $e_\lambda$ 相关联的一个顶点删除 (这也删除了边 $e_\lambda$ ), 显然我们至多删除了 $\lambda(G)$ 个顶点, 但得到的图同样不连通, 因此 $\kappa(G) \leq \lambda(G)$ 。□

## 1.4 欧拉图与哈密顿图

欧拉图来源于“哥尼斯堡七桥问题”, 是图论研究的发源之一。

**定义 1.4.1** 无向连通图 $G = (V, E)$ 的一条经过所有边的简单回路 (道路) 称为 $G$ 的**欧拉回路 (道路)**。

**定理 1.4.2** 无向连通图 $G = (V, E)$ 存在欧拉回路的充要条件是 $G$ 的所有顶点的度数都为偶数。

**证明** 必要性。若 $G$ 中有欧拉回路 $C$ , 则 $C$ 过每一条边一次且仅一次。对任一顶点 $v$ 来说, 如果 $C$ 经由 $e$ 进入 $v$ , 则一定通过另一条边 $e'$ 离开 $v$ , 因此 $v$ 的度数是偶数。

充分性。由于 $G$ 的所有顶点度数都是偶数, 因此对 $G$ 的任意顶点 $v_0$ , 一定存在从 $v_0$ 出发的 $G$ 的一个简单回路 $C$ 。如果 $C$ 已经包含 $G$ 的所有边, 则得到一个欧拉回路, 否则考虑 $G - C$ , 它的所有顶点度数仍然是偶数, 而且一定存在一个顶点 $v$ 与 $C$ 有边相连, 而且顶点 $v$ 的度数大于0, 从而这一点开始又存在一个简单回路 $C'$ , 这样 $C \cup C'$ 就是一个新的回路, 比 $C$ 的边多, 一直这样扩大回路, 直到包含所有的边, 则得到一个欧拉回路。□

根据这个定理可找出一个所有顶点都是偶数的图的欧拉回路, 具体例子可参见戴一奇教材p16的例2.3.1。

**推论 1.4.3** 若无向连通图 $G$ 中只有2个度为奇数的顶点, 则 $G$ 存在欧拉道路。

欧拉回路有着十分有趣的应用, 例如戴一奇教材p17的例2.3.4给出了其在计算机科学中一个十分有趣的应用。我国图论专家管梅谷先生提出的中国邮路问题也与欧拉回路有着十分密切的联系。

哈密顿图来源于19世纪英国数学家哈密顿提出的周游世界的问题, 后来成为图论研究中一个重要问题, 并且有许多重要的应用。

**定义 1.4.4** 无向图的一条经过全部顶点的初级回路 (道路) 称为它的**哈密顿回路 (道路)**, 简称**H回路 (道路)**。

至今没有找到一个图存在哈密顿回路的充分且必要的条件, 但可以证明下面的充分条件:

**定理 1.4.5** 如果简单图 $G$ 的任意两个顶点 $v_i, v_j$ 的度数之和大于等于 $n - 1$ , 这里 $n$ 是图 $G$ 的顶点个数, 则 $G$ 存在哈密顿道路。

**证明** 参见戴一奇教材p16定理2.4.1的证明。先证 $G$ 是连通图。若 $G$ 非连通, 则至少有2个连通分支 $H_1, H_2$ , 其顶点数分别为 $n_1, n_2$ , 从中各取一个顶点 $v_1, v_2$ , 则 $d(v_1) \leq n_1 - 1$ , 而 $d(v_2) \leq n_2 - 1$ , 从而 $d(v_1) + d(v_2) \leq n_1 + n_2 - 2 \leq n - 1$ , 矛盾! 所以 $G$ 必然是连通图。

下面证明 $G$ 存在哈密顿道路。设 $\Gamma = v_{i1}v_{i2}\cdots v_{il}$ 是 $G$ 的一条极长初级道路，即 $\Gamma$ 的两个端点 $v_{i1}$ 和 $v_{il}$ 的相邻顶点都在 $\Gamma$ 中。若 $l = n$ ，则 $\Gamma$ 就是一条哈密顿道路。

若 $l < n$ ，我们可用反证法证明 $G$ 中存在经过顶点 $v_{i1}, v_{i2}, \dots, v_{il}$ 的回路。因为若不存在这样的回路，考察 $v_{i1}$ 和 $v_{il}$ 的相邻顶点，注意它们的相邻顶点都在 $\Gamma$ 中，若假设不存在这样的回路，那么对任意的 $v_{ip} \in \Gamma$ ， $v_{i1}$ 与 $v_{ip}$ 相邻意味着 $v_{il}$ 就不能与 $v_{ip-1}$ 相邻（否则 $v_{i1}v_{i2}\cdots v_{ip-1}v_{il}v_{il-1}\cdots v_{ip}$ 就是一条回路），于是当 $d(v_{i1}) = k$ 时， $d(v_{il}) \leq l - k - 1$ （其中减1因为不能与自身相邻，注意图 $G$ 是简单图），从而 $d(v_{i1}) + d(v_{il}) < n - 1$ ，与题设条件矛盾！这就证明了当 $l < n$ 时存在经过顶点 $v_{i1}, \dots, v_{il}$ 的回路 $C$ 。

由于 $G$ 是连通图，所以存在 $C$ 之外的顶点 $v_t$ 与 $C$ 的某个顶点 $v_{iq}$ 相邻，删除 $(v_{iq-1}, v_{iq})$ ，则 $\Gamma' = v_tv_{iq}\cdots v_{ip-1}v_{il}\cdots v_{iq-1}$ 是 $G$ 中比 $\Gamma$ 更长的初级道路，以 $\Gamma'$ 的两个端点可继续扩充，得到更长的初级道路，重复此过程，最后必得到一条包含 $G$ 的所有顶点的初级道路，即 $G$ 的哈密顿道路。□

**推论 1.4.6** 若简单图 $G$ 的任意两个顶点 $v_i, v_j$ 的度数之和大于等于 $n$ ，这里 $n$ 是图 $G$ 的顶点个数，则 $G$ 存在哈密顿回路。

**证明** 由上面的定理知其 $G$ 存在哈密顿道路 $H$ ，而且该道路是 $G$ 的一条极长道路，设其端点是 $v_1$ 和 $v_n$ ，若 $G$ 不存在包含 $H$ 中（也就是 $G$ 中）所有顶点的回路，那么与在上述定理中的证明类似， $d(v_1)$ 和 $d(v_n)$ 的和小于等于 $H$ 的长度减1，即 $d(v_1) + d(v_n) \leq n - 1$ ，与题设条件矛盾！所以 $G$ 一定存在哈密顿回路。□

哈密顿回路在生活以及一些研究中有十分重要的应用，其中旅行商问题与哈密顿回路的关系十分密切，而旅行商问题是一个典型的NP问题，在计算机科学中的可计算性理论及算法理论的研究中有着十分重要的地位。

## 1.5 邻接矩阵与可达矩阵

在对图进行描述或运算时，常常要采用图的矩阵表示。图的矩阵表示有两种，一种是邻接矩阵，一种是关联矩阵。邻接矩阵表示顶点之间的邻接关系，而关联矩阵表示顶点与边之间的关联关系。这一节只介绍邻接矩阵，关联矩阵将在树的一章介绍。

**定义 1.5.1** 给定有向图 $G = (V, E)$ 无多重边， $V = \{v_1, v_2, \dots, v_n\}$ 。有向图 $G$ 的**邻接矩阵** $A$ 是一个 $n$ 阶方阵 $A = [a_{ij}]_{n \times n}$ ，定义为：

$$a_{ij} = \begin{cases} 1 & \text{若 } \langle v_i, v_j \rangle \in E \\ 0 & \text{否则} \end{cases}$$

有向图邻接矩阵的行的非零元素个数是该行对应顶点的正度，列的非零元素个数是该列对应顶点的负度。

**定义 1.5.2** 给定无向图 $G = (V, E)$ 无多重边， $V = \{v_1, v_2, \dots, v_n\}$ 。无向图 $G$ 的**邻接矩阵** $A$ 是一个 $n$ 阶方阵 $A = [a_{ij}]_{n \times n}$ ，定义为：

$$a_{ij} = \begin{cases} 1 & \text{若 } (v_i, v_j) \in E \\ 0 & \text{否则} \end{cases}$$

显然无向图的邻接矩阵是一个对称矩阵，它的行的非零元素个数是该行对应顶点的度数。

对于有向图 $G = (V, E)$ 的邻接矩阵 $A$ ，可通过矩阵乘法计算 $A$ 的 $k$ 幂 $A^k$ ， $A^k$ 中的元素 $a_{ij}^{(k)}$ 表示顶点 $v_i$ 到 $v_j$ 的长度为 $k$ 的道路条数。如果我们只关心 $v_i$ 是否可达 $v_j$ ，则可在矩阵乘法中使用逻辑运算，即该用实数乘法时使用逻辑与，该用实数加法时使用逻辑或。我们定义有向图的可达矩阵：

**定义 1.5.3** 给定有向图 $G = (V, E)$ ， $V = \{v_1, v_2, \dots, v_n\}$ 。有向图 $G$ 的**可达矩阵** $P$ 是一个 $n$ 阶方阵 $P = [p_{ij}]_{n \times n}$ ，定义为：

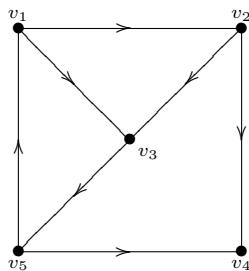
$$p_{ij} = \begin{cases} 1 & \text{若 } v_i \text{ 可达 } v_j \\ 0 & \text{否则} \end{cases}$$

利用邻接矩阵使用矩阵乘法计算可达矩阵虽然可行，但是计算起来比较复杂，一个更有效的计算方法是Warshall算法。Warshall 算法的基本思想是：

第 $k$ 次循环计算的 $p_{ij}^{(k)}$ 是顶点 $v_i$ 是否能经过 $v_1, v_2, \dots, v_k$ 到达 $v_j$ 。从而第 $n$ 次循环时， $p_{ij}^{(n)} = 1$ 表示顶点 $v_i$ 可达 $v_j$ 。

下面使用例子说明Warshall 算法计算可达矩阵的基本思想。

**例子 1.5.4** 参见戴一奇教材p13的例2.2.1。考虑下面的图：



Warshall算法的计算结果如下：

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \quad P^{(1)} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad P^{(2)} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$P^{(3)} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad P^{(4)} = P^{(3)} \quad P^{(5)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

注意Warshall算法的第 $k$ 次循环计算的是 $P^{(k)}$ ，其计算方法是将第 $k$ 行叠加第 $k$ 行中哪些不为0的行，这里叠加是逻辑或操作。例如，在计算 $P^{(3)}$ 时，第3列中有 $p_{13}, p_{23}, p_{53}$ 这几个元素为1，因此将第3行分别叠加到第1, 2, 5行，例如，对于第1行，令 $p_{1j} = p_{1j} \vee p_{3j} (j = 1, \dots, 5)$ 。上面将每一次循环对原来矩阵作了改变的数字用了黑体字标出。



## 第二章 树的基本概念

这一章我们考虑与树有关的基本概念,包括无向树的定义及基本性质、图的关联矩阵与生成树的计数、有向树(根树)和二叉树的性质,以及最优二叉树(Huffman树)的构造及应用等。

### 2.1 树的基本定义

**定义 2.1.1** 若简单连通图  $G = (V, E)$  没有回路,则称  $G$  是(无向)树。无向树中度数为1的顶点称为**树叶**,度数大于等于2的顶点称为**分支点**。只有一个顶点(无边)的简单图称为**平凡树**;若  $G$  含有多个无回路的连通分支,则称  $G$  是**森林**。

也就是说,(无向)树就是连通无回路的简单图。后面在提到树时,如果没有特别说明都是指无向树。在证明树的性质之前,我们先定义桥(或说割边),我们将看到树的每一条边都是桥。

**定义 2.1.2** 给定图  $G = (V, E)$ , 说边  $e \in E$  是  $G$  的**桥**(割边), 如果  $p(G - e) > p(G)$ , 即  $G$  中删除  $e$  之后会增加连通分支数。

显然,如果  $e$  是  $G$  的桥,则有  $p(G - e) = p(G) + 1$ , 即删除一条边,最多增加一个连通分支。另一方面,若  $e$  是  $G$  的桥,则  $e$  不属于  $G$  的任意回路。根据这些基本常识,下面证明树的基本性质。

**定理 2.1.3** 给定简单图  $G = (V, E)$ ,  $|V| = n, |E| = m$ , 下面各命题等价:

- (1)  $G$  是树(即  $G$  连通且无回路);
- (2)  $G$  的任意两个顶点存在惟一的道路;
- (3)  $G$  连通且任意边都是桥;
- (4)  $G$  无回路,但在任何两个不相邻顶点之间加一条新边,则得到惟一一个含新边的回路。

**证明** 我们采用循环论证法,即证明  $(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (1)$ 。

(1)  $\Rightarrow$  (2): 由于  $G$  是连通的,因此  $G$  的任意两个顶点之间都存在道路,而若  $G$  的两个顶点之间存在两个不同的道路,则这两个道路构成回路,与  $G$  无回路矛盾,因此  $G$  的任意两个顶点必存在惟一的道路。

(2)  $\Rightarrow$  (3): 当  $G$  的任意两个顶点存在惟一的道路时,显然  $G$  是连通的。对  $G$  的任意边  $e = (u, v)$ , 如果  $e$  不是  $G$  的桥,则  $G - e$  仍是连通的,即存在以  $u$  和  $v$  为端点的道路  $\Gamma$ , 那么  $G$  就存在以  $u$  和  $v$  的两条道路  $\Gamma$  和  $\Gamma' = uev$ , 矛盾,因此  $G$  的任意边必是桥。

(3)  $\Rightarrow$  (4): 因为  $G$  的任意边都是桥,因此  $G$  无回路,因此  $G$  连通且无回路,即  $G$  是树,从而由 (1)  $\Rightarrow$  (2) 得  $G$  的任意两个顶点之间只存在惟一的道路,从而若在  $G$  的两个不相邻顶点之间加一条

新边, 必得到含有该新边的回路, 且该回路是惟一的 (不然则这两个顶点之间原来就会有两条不同的道路, 矛盾! )。

(4)  $\Rightarrow$  (1): 只要证明 $G$ 是连通的, 对 $G$ 的任意两个顶点 $u, v$ , 如果 $u$ 和 $v$ 相邻, 则 $u$ 和 $v$ 是可达的; 如果 $u$ 和 $v$ 不相邻, 则在 $u$ 和 $v$ 之间增加一条新边可得到惟一个含该新边的回路, 这表明在 $G$ 中 $u$ 和 $v$ 也是可达的。因此 $G$ 的任意两个顶点都是可达的, 即 $G$ 是连通的。  $\square$

**定理 2.1.4** 给定简单图 $G = (V, E)$ ,  $|V| = n, |E| = m$ , 下面各命题等价:

- (1)  $G$ 是树 (即 $G$ 连通且无回路);
- (2)  $G$ 连通且任意边都是桥;
- (3)  $G$ 无回路且 $m = n - 1$ ;
- (4)  $G$ 连通且 $m = n - 1$ ;

**证明** 由于(1)和(2)的等价性上面已经证明, 因此这里只要证明(2)  $\Rightarrow$  (3)  $\Rightarrow$  (4)  $\Rightarrow$  (2)即可。

(2)  $\Rightarrow$  (3): 显然这时 $G$ 没有回路, 我们使用归纳法证明 $m = n - 1$ 。当 $n = 1$ 时,  $G$ 为平凡图 (平凡树),  $m = 0$ , 显然 $m = n - 1$ 。假定对任意少于 $k$ 个顶点的连通图 $G$ , 若 $G$ 的任意边都是桥时,  $G$ 的边数等于顶点数减一。考虑具有 $k$ 个顶点的图, 对 $G$ 的任意边 $e$ , 因为 $G$ 连通且 $e$ 是桥, 因此 $p(G - e) = 2$ , 设 $G$ 的两个连通分支分别是 $G_1$ 和 $G_2$ , 其顶点数分别是 $n_1$ 和 $n_2$ , 边数分别是 $m_1$ 和 $m_2$ , 则显然 $G_1$ 和 $G_2$ 的任意边仍然是桥, 且 $n_1 < k$ 和 $n_2 < k$ , 根据归纳假设有 $m_1 = n_1 - 1$ 且 $m_2 = n_2 - 1$ , 从而 $G$ 共有 $n_1 - 1 + n_2 - 1 + 1 = n_1 + n_2 - 1 = k - 1$ 条边, 命题成立。因此当 $G$ 连通且任意边是桥时,  $G$ 无回路且边数等于顶点数减1。

(3)  $\Rightarrow$  (4): 这只要证明 $G$ 是连通的, 用反证法, 设 $G$ 有 $k > 1$ 个连通分支, 每个连通分支的顶点数分别是 $n_1, \dots, n_k$ , 则 $G$ 的每个连通分支无回路, 即 $G$ 的每个连通分支是树, 从而 $G$ 的每个连通分支的边数分别是 $n_1 - 1, \dots, n_k - 1$ , 从而 $G$ 的边数 $m$ 为 $n_1 - 1 + n_2 - 1 + \dots + n_k - 1 = n - k$ , 当 $k > 1$ 时与 $m = n - 1$ 矛盾! 因此必有 $k = 1$ , 即 $G$ 是连通图。

(4)  $\Rightarrow$  (2): 只要证明当 $G$ 连通且 $m = n - 1$ 时,  $G$ 的任意边都是桥。对 $G$ 的任意边 $e$ , 若 $G - e$ 是连通的, 则由于 $G - e$ 的顶点数仍然是 $n$ , 而连通简单图的边数要大于顶点数减1, 因此 $G - e$ 的边数要大于等于 $n - 1$ , 但 $G$ 的边数 $m = n - 1$ , 从而 $G - e$ 的边数是 $n - 2$ , 因此必有 $G - e$ 是不连通的, 这就说明 $e$ 是桥。  $\square$

总的来说, 上面两个定理说明树具有这样的性质: (i) **连通无回路**; (ii) **边数等于顶点数减一**; (iii) **每条边都是桥**; (iv) **任意两个顶点之间有惟一的道路**; (v) **任何两个不相邻顶点之间加新边得惟一的回路**。我们通常用 $T = (V, E)$ 表示树。

**例子 2.1.5** 若 $T = (V, E)$ 是树, 且 $|V| = n \geq 2$ , 则 $T$ 至少有两片树叶, 即至少有两个度数为1的顶点。

**证明** 因为 $T$ 是连通图, 所以对 $T$ 的任意顶点 $v$ 都有 $d(v) \geq 1$ , 如果 $T$ 至少有 $n - 1$ 个顶点的度数大于等于2, 设 $T$ 的边数为 $m$ , 则

$$2m = \sum_{v \in V} d(v) \geq 2(n - 1) + 1$$

即 $m \geq n$ , 这与 $m = n - 1$ 矛盾, 因此 $T$ 至多只有 $n - 1$ 个顶点的度数大于等于2, 也即至少有两个顶点的度数为1。  $\square$



## 2.2 生成树

任何无向连通图都存在一棵树作为它的子图，我们称这种树为无向连通图的树，特别地，若这种树是图的生成子图（即顶点数一样的子图），则称其为生成树：

**定义 2.2.1** 给定无向连通图  $G = (V, E)$ （不一定是简单图），若  $T = (V, E')$  是  $G$  的生成子图且是树，则称  $T$  是  $G$  的**生成树**。若  $T$  是  $G$  的生成树，则对任意的边  $e \in E$ ，若  $e \in E'$ ，则称  $e$  是  $T$  的**树枝**，否则称  $e$  是  $T$  的**弦**，并称由  $G$  的顶点和  $T$  的弦构成的  $G$  的子图（即  $G - T$ ）称为  $T$  的**余树**，记为  $\bar{T}$ 。

注意  $G$  的生成树  $T$  的余数  $\bar{T}$  不一定连通，也不一定不含有回路，因此  $\bar{T}$  不一定是树。很容易证明：

**引理 2.2.2** 任意无向图  $G = (V, E)$  具有生成树当且仅当  $G$  是连通图。

**证明** 显然当  $G$  具有生成树时， $G$  是连通图，因此生成树本身是连通图。反之，当  $G$  是连通图时，若  $G$  无回路，则  $G$  就是自己的生成树；若  $G$  含有回路，任取一回路，随意地删除回路上的一条边，若再有回路则继续删除回路上的边，直到不含有任意回路为止，最后得到的图仍然是连通的（因为删除回路上的边不会增加连通分支），而且是无回路的，也即就是  $G$  的生成树。通常人们称这种产生生成树的方法为**破圈法**。□

**推论 2.2.3** 设图  $G$  是  $n$  个顶点  $m$  条边的无向连通简单图，则  $m \geq n - 1$ 。

**推论 2.2.4** 设图  $G$  是  $n$  个顶点  $m$  条边的无向连通简单图， $T$  是  $G$  的生成树，则  $T$  的余树  $\bar{T}$  有  $m - n + 1$  条边。

**推论 2.2.5** 设  $T$  是连通图  $G$  的一棵生成树， $\bar{T}$  是它的余树，则  $G$  的任意回路  $C$  必含  $\bar{T}$  中的边（即  $T$  的弦）。

**证明** 因为否则的话，回路  $C$  的边都在树  $T$  中，这与  $T$  无回路矛盾。□

更进一步可证明下面的定理：

**定理 2.2.6** 设  $T$  是连通图  $G$  的一棵生成树， $e$  为  $T$  的弦，则  $T \cup \{e\}$  必含有一个回路使得该回路除了含有弦  $e$  之外不再有  $T$  的其他弦。

**证明** 设  $e = (u, v)$ ，则  $T$  中有  $u$  到  $v$  的惟一道路  $\Gamma$ ，显然  $\Gamma \cup \{e\}$  是含有弦  $e$  的回路。□

显然不同的弦含的回路也不同，我们有如下的定义：

**定义 2.2.7** 设  $T$  是  $n$  个顶点  $m$  条边的无向连通简单图  $G$  的一棵生成树，设  $e'_1, e'_2, \dots, e'_{m-n+1}$  是  $T$  的弦，设  $C_r$  是  $T$  添加弦  $e'_r$  后产生的只含弦  $e'_r$ ，其他边都是树枝的回路，称  $C_r$  是  $G$  的对应  $T$  的弦  $e'_r$  的**基本回路**，称  $\{C_1, C_2, \dots, C_{m-n+1}\}$  是  $G$  对应  $T$  的**基本回路系统**。

注意，不同生成树的基本回路系统可能不同，但不同生成树的基本回路系统中回路数目是一样的，即都是  $m - n + 1$ 。从弦导出基本回路系统，从树枝则可得到基本割集系统。首先有如下定理：

**定理 2.2.8** 设  $T$  是连通图  $G$  的一棵生成树， $e$  是  $T$  的树枝，则  $G$  存在只含树枝  $e$ ，其他边都是弦的边割集，且不同的树枝对应的边割集也不同。

**证明** 根据树的基本性质,  $e$  是  $T$  的桥, 因此  $T - e$  有两个连通分支  $T_1, T_2$ , 令

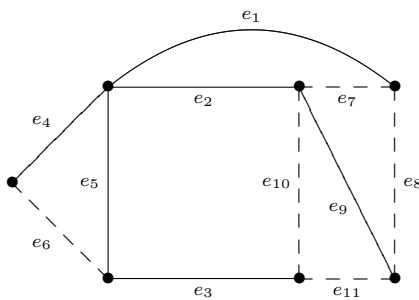
$$S_e = \{(u, v) \mid u \in T_1 \wedge v \in T_2\}$$

即  $S_e$  是哪些两个端点分属于  $T_1$  和  $T_2$  的边构成的边集, 显然  $S_e$  是边割集, 而且  $e \in S_e$ , 且  $S_e$  中的其他边都是弦 (否则作为树  $T$  的连通分支  $T_1$  和  $T_2$  有边相连)。□

**定义 2.2.9** 设  $T$  是有  $n$  个顶点的连通图  $G$  的一棵生成树。  $e_1, e_2, \dots, e_{n-1}$  是  $T$  的树枝。  $S_i$  是  $G$  的只含树枝  $e_i$  的边割集, 则称  $S_i$  是  $G$  的对应  $T$  由树枝  $e_i$  生成的 **基本割集**, 称  $\{S_1, S_2, \dots, S_{n-1}\}$  为  $G$  对应  $T$  的 **基本割集系统**。

同样, 不同生成树的基本割集系统也可能不同, 但不同生成树的基本割集系统中的边割集数目都是一样的, 即  $n - 1$ 。

**例子 2.2.10** 考虑下面的图 (参见耿素云教材[1]p309的图16.3):



其中实线给出的边构成了上图的生成树, 它的弦包含  $e_6, e_7, e_8, e_{10}, e_{11}$ , 对应的基本回路系统是:

$$C_6 = e_6 e_4 e_5$$

$$C_7 = e_7 e_2 e_1$$

$$C_8 = e_8 e_9 e_2 e_1$$

$$C_{10} = e_{10} e_3 e_5 e_2$$

$$C_{11} = e_{11} e_3 e_5 e_2 e_9$$

该生成树的树枝包括  $e_1, e_2, e_3, e_4, e_5, e_9$ , 对应的基本割集系统是:

$$S_1 = \{e_1, e_7, e_8\}$$

$$S_2 = \{e_2, e_7, e_9, e_{10}, e_{11}\}$$

$$S_3 = \{e_3, e_{10}, e_{11}\}$$

$$S_4 = \{e_4, e_6\}$$

$$S_5 = \{e_5, e_6, e_{10}, e_{11}\}$$

$$S_6 = \{e_9, e_8, e_{11}\}$$

为了得到连通图的所有生成树及其数目, 我们考虑图的关联矩阵:

**定义 2.2.11** 给定有向图  $G = (V, E)$ , 设  $V = \{v_1, v_2, \dots, v_n\}$ , 而  $E = \{e_1, e_2, \dots, e_m\}$ , 则  $G$  的关联矩阵  $B = [b_{ij}]$  是  $n \times m$  的矩阵, 对任意的  $1 \leq i \leq n, 1 \leq j \leq m$  有:

$$b_{ij} = \begin{cases} 1 & \text{若 } v_i \text{ 是 } e_j \text{ 的起点} \\ -1 & \text{若 } v_i \text{ 是 } e_j \text{ 的终点} \\ 0 & \text{否则} \end{cases}$$

显然, 任何一个有向图的关联矩阵具有如下特征: (i) 孤立点对应的行向量为零向量; (ii) 每个列向量有且仅有两个非零元 $-1$ 和 $1$ ; (iii) 如果该有向图不连通, 则通过适当重排之后, 其关联矩阵可变成对角分块矩阵。

关联矩阵的秩与有向图的弱连通性有密切关系, 为方便起见, 我们将有向图不考虑边之后得到的无向图称为该有向图的**基图**。

**定理 2.2.12** 弱连通有向图 $G = (V, E)$ 的关联矩阵 $B$ 的秩 $r(B) = |V| - 1$ 。

**证明 略**

□

有向图的关联矩阵划去一行向量后得到的矩阵称为该有向图的基本关联矩阵。

**定义 2.2.13** 给定有向图 $G = (V, E)$ , 设 $V = \{v_1, v_2, \dots, v_n\}$ , 而 $E = \{e_1, e_2, \dots, e_m\}$ 。其关联矩阵 $n \times m$ 矩阵 $B$ , 称划去 $B$ 中对应顶点 $v_k$ 对应的行向量得到的 $(n-1) \times m$ 矩阵 $B_k$ 为 $G$ 的对应 $v_k$ 的**基本关联矩阵**。

基本关联矩阵与有向图的基图中的回路有如下关系:

**定理 2.2.14** 给定有向图 $G = (V, E)$ , 设 $V = \{v_1, v_2, \dots, v_n\}$ , 而 $E = \{e_1, e_2, \dots, e_m\}$ 。  $B_k$ 是 $G$ 的基本关联矩阵。设 $\Gamma = v_{i1}e_{i1}v_{i2}e_{i2} \cdots e_{ij}v_{i1}$ 是 $G$ 的基图的一个简单回路, 则 $e_{i1}, \dots, e_{ij}$ 对应的 $B_k$ 的各列向量**线性相关**。

**证明 略**

□

根据这个性质, 我们可以证明:

**定理 2.2.15** 给定有向弱连通图 $G = (V, E)$ ,  $|V| = n$ , 其基本关联矩阵 $B_k$ 的任一 $(n-1)$ 阶子阵非零的充要条件是: 该子阵各列对应的图 $G$ 的边 (不考虑方向) 构成 $G$ 的基图的一棵生成树。

**证明 略**

□

根据在矩阵理论中的Binet-Cauchy定理, 我们可以对图的生成树进行计数:

**定理 2.2.16** 设 $B_k$ 是有向弱连通图 $G = (V, E)$ 的某一基本关联矩阵, 则 $G$ 的不同树的生成树数目是 (下面的 $B_i$ 是 $B_k$ 的某一 $n-1$ 阶子阵, 而下面的求和则取遍 $B_k$ 的所有可能 $n-1$ 阶子式):

$$|B_k B_k^T| = \sum_i |B_i| |B_i^T| = \sum_i |B_i|^2$$

如果要给出图 $G$ 的所有生成树, 则可在 $B_k$ 中用 $E$ 中相应的元素标记, 得到矩阵 $B_k^e = (b_{ij}^e)$ , 即令:

$$b_{ij}^e = \begin{cases} e_j & \text{若 } b_{ij} = 1 \\ -e_j & \text{若 } b_{ij} = -1 \\ 0 & \text{否则} \end{cases}$$

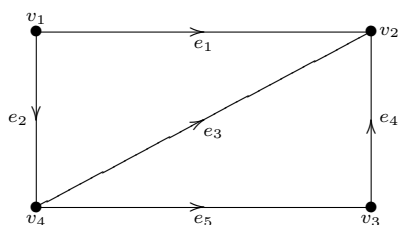
则 $|B_k^e B_k^T|$ 给出所有生成树的清单。

**证明 略**

□

我们以一个比较简单的例子来说明上述定理的使用：

**例子 2.2.17** 求下图的生成树的数目：



**解：**任取该图的一个基本关联矩阵，例如  $B_4$ ：

$$B_4 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

如果直接计算  $|B_4 B_4^T|$  则有：

$$|B_4 B_4^T| = \begin{vmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{vmatrix} = 8$$

如果利用Binet-Cauchy定理则共有  $C_5^3 = 10$  种可能，从而有：

$$\begin{aligned} |B_4 B_4^T| = & \begin{vmatrix} 1 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 0 & 0 \end{vmatrix}^2 + \begin{vmatrix} 1 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 0 & 1 \end{vmatrix}^2 + \begin{vmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{vmatrix}^2 + \\ & \begin{vmatrix} 1 & 0 & 0 \\ -1 & -1 & -1 \\ 0 & 0 & 1 \end{vmatrix}^2 + \begin{vmatrix} 1 & 0 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & -1 \end{vmatrix}^2 + \begin{vmatrix} 1 & 0 & 0 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \end{vmatrix}^2 + \\ & \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & -1 \end{vmatrix}^2 + \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{vmatrix}^2 + \\ & \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & -1 \end{vmatrix}^2 + \begin{vmatrix} 0 & 0 & 0 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \end{vmatrix}^2 \end{aligned}$$

显然除了第一个和最后一个行列式为0之外，其他的行列式的值都等于1。

为了给出所有生成树，我们计算：

$$\begin{aligned}
 |B_4^e B_4^T| &= \begin{vmatrix} e_1 & e_2 & 0 & 0 & 0 \\ -e_1 & 0 & -e_3 & -e_4 & 0 \\ 0 & 0 & 0 & e_4 & -e_5 \end{vmatrix} \begin{vmatrix} 1 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{vmatrix} = \begin{vmatrix} e_1 + e_2 & -e_1 & 0 \\ -e_1 & e_1 + e_3 + e_4 & -e_4 \\ 0 & -e_4 & e_4 + e_5 \end{vmatrix} \\
 &= (e_1 + e_2)((e_1 + e_3 + e_4)(e_4 + e_5) - e_4^2) - e_1^2(e_4 + e_5) \\
 &= (e_1 + e_2)(e_1 e_4 + e_1 e_5 + e_3 e_4 + e_3 e_5 + e_4 e_5) - e_1^2 e_4 - e_1^2 e_5 \\
 &= e_1 e_3 e_4 + e_1 e_3 e_5 + e_1 e_4 e_5 + e_2 e_1 e_4 + e_2 e_1 e_5 + e_2 e_3 e_4 + e_2 e_3 e_5 + e_2 e_4 e_5
 \end{aligned}$$

这就得到了所有的生成树，即由边 $e_1, e_3, e_4$ 构成的生成树，由 $e_1, e_3, e_5$ 构成的生成树，等等。

最后，如果我们要求无向连通图的所有生成树，**只要将该无向图的每一条随意地给一个方向，然后利用上面的方法计算所得到的弱连通有向图的所有生成树即可。**

## 2.3 根树

**定义 2.3.1** 如果一个弱连通有向图不考虑边的方向是树，则称这个图是**有向树**。若有向树 $T$ 有且仅有一个顶点 $v$ 的入度为0，则称 $T$ 是以 $v$ 为**根**的**根树**。我们将根树中出度为0的顶点为**叶子**。

**定义 2.3.2** 对于根树 $T = (V, E)$ ， $T$ 的两个顶点 $u, v$ 。若有有向边 $e = \langle u, v \rangle \in E$ ，则称 $u$ 是 $v$ 的**父亲**，而 $v$ 是 $u$ 的**儿子**；若存在 $u$ 到 $v$ 的一条有向道路，则称 $u$ 是 $v$ 的**祖先**，而 $v$ 是 $u$ 的**后代**。

**定义 2.3.3** 对于根树 $T = (V, E)$ ，设其根为 $v$ ，则对 $T$ 的任意顶点 $u$ ，从 $v$ 到 $u$ 的惟一道路的长度称为 $u$ 的**深度**。定义 $T$ 的**高度**是从 $v$ 到 $T$ 的叶子最长道路的长度。

显然根节点的深度为0，我们称之为第0层节点；深度同为 $i$ 的节点称为第 $i$ 层节点，具有最大深度的节点的深度为树的深度（高度）。对于根树，如果将每个节点的儿子按照某种特定的次序（例如在画的时候从左至右的次序）排列，则称该根树为有序根树，如果有序根树的所有节点的出度不大于 $m$ ，则称为 **$m$ 元树**（或说 **$m$ 叉树**）；如果有序根树的所有节点的出度都是0或 $m$ ，则称为 **$m$ 元正则树**（或说 **$m$ 叉正则树**）。

根树在计算机科学及实际生活中有许多应用，例如语法树、判定树（决策树）等都是根树的例子。也许在计算机科学中用得最多的是二元（叉）树。

**定义 2.3.4** **二叉树**是二元有序树，即每个节点的出度不大于2，且每个节点的儿子按一定顺序排列的根树。每个节点的出度为0或2的二叉树称为**正则二叉树**。

通常我们将二叉树某个节点的两个儿子节点区分为左儿子和右儿子节点。不难看出，二叉树具有这样的特点：(i) 第 $k$ 层最多有 $2^k$ 个节点；(ii) 高度为 $k$ 的二叉树最多有 $2^{k+1} - 1$ 个节点。进一步， $m$ 元有序树都可转化为一棵二叉树：以某个结点的最左儿子作为该结点的左儿子，而以该结点的右兄弟作为该结点的右儿子。

**例子 2.3.5** 设 $T$ 是二叉树, 其出度为2的节点数目是 $n_2$ , 叶子结点的数目是 $n_0$ , 证明 $n_0 = n_2 + 1$ 。

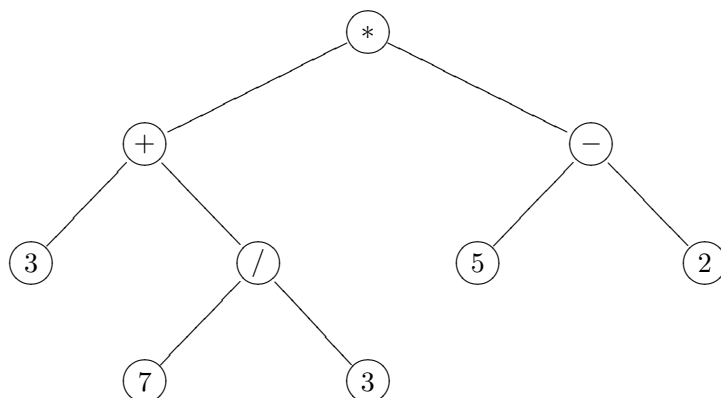
**证明** 对 $T$ 的结点数 $n$ 进行归纳, 显然当 $n = 1$ 时, 一个叶子结点, 零个出度为2的节点, 因此命题成立; 设对于任意小于 $k$ 个结点的二叉树叶子数等于出度为2的节点数加1。考虑有 $k$ 个结点的任意二叉树 $T$ , 若 $T$ 的根 $v$ 的出度为1, 则 $T$ 与 $T - v$ 的叶子数和出度为2的节点数都相同, 而根据归纳假设,  $T - v$ 的叶子数等于出度为2的节点数加1, 因此这时命题成立; 若 $T$ 的根 $v$ 的出度为2, 则 $v$ 的左子树 $T_1$ 和 $v$ 的右子树 $T_2$ 都是结点数小于 $k$ 的二叉树, 设其叶子数分别为 $n_{01}$ 和 $n_{02}$ , 出度为2的节点数为 $n_{21}$ 和 $n_{22}$ , 则根据归纳假设有:

$$n_{01} = n_{21} + 1 \quad n_{02} = n_{22} + 1$$

也即 $n_{01} + n_{02} = n_{21} + n_{22} + 2$ , 注意到这时 $T$ 恰好有 $n_{01} + n_{02}$ 个叶子节点, 而有 $n_{21} + n_{22} + 1$ 个出度为2的结点 (比 $T - v$ 多 $v$ 这个出度为2的节点), 因此这时命题也成立。□

二叉树的遍历是程序设计中常常遇到的一个问题, 数据课程中也常常研究二叉树的各种遍历方法。我们这里以一个简单的例子说明二叉树的中序、前序和后序遍历方法。

**例子 2.3.6** 考虑下面的二叉树:



对上述二叉树进行中序遍历则得到:

$$3 + 7/3 * 5 - 2$$

当然这不加上括号不会得到正确的算术表达式, 加上适当括号则得到 $(3 + 7/3) * (5 - 3)$ 。

对上述二叉树进行后序遍历则得到

$$373/+52-*$$

对这个串作一遍扫描即可计算表达式的值。对上述二叉树进行前序遍历则得到

$$* + 3/73 - 52$$

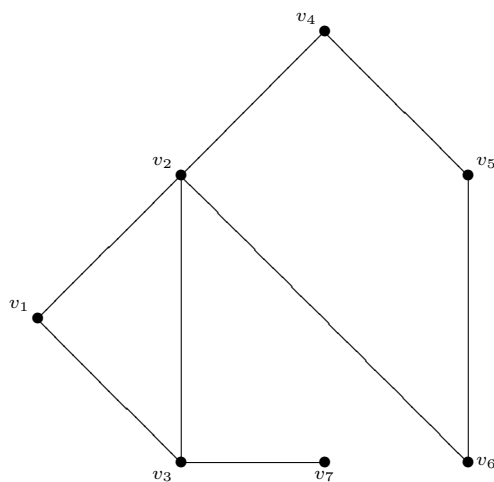
同样也可利用这个串计算该表达式的值, 但不常用。实际上, 二叉树的中序遍历常用于说明表达式的语法, 而后序遍历常用于计算表达式的值。

至于 $m$ 元树, 我们可以按照上述方法将 $m$ 元树转换为2元树再进行遍历。如果直接遍历 $m$ 元树, 从根节点出发, 有两种策略: 一是先深搜索: 先不断地搜索结点的儿子, 直到叶子结点, 再回溯到父

亲结点，再从该父亲结点的另外儿子开始继续策略；二是先广搜索：先搜索所有儿子结点，然后再根据一定的顺序（从左到右）选择一个儿子节点再继续搜索。

实际上，上面的中序遍历采用的就是先深搜索策略，而前序遍历采用的就是先广搜索策略。对于一般的图（有向图或无向图）的遍历也可采用这两种策略：先深搜索是从一个结点开始不断搜索其相邻结点，直到没有还未搜索过的相邻结点，则回溯到上一结点，而先广搜索是先搜索一个结点的所有相邻结点，然后选一个结点再继续搜索。

**例子 2.3.7** 考虑下图的顶点遍历（边的遍历与顶点遍历类似），也即按照某种策略访问下图的所有顶点：



我们分别使用先深搜索策略和先广搜索策略从 $v_1$ 开始遍历上图的所有顶点。在遍历时，我们总是使用序列 $T$ 表示哪些顶点已经被遍历，并假定我们用某种方式记住了顶点与顶点之间的相邻关系（例如使用邻接矩阵），因此我们总可很容易地得到某个顶点的相邻顶点。读者可将序列 $T$ 理解为一些元素的集合，但这些元素有顺序以便给出我们遍历顶点的顺序。

为了使用**先深策略**，我们还需要用一个**栈** $S$ 保存刚遍历过但还有相邻节点没有遍历的节点，这里大家无需深究栈到底是什么东西，你可将它理解为也是一个序列，即其中元素有顺序的集合，而且我们按照所谓的“**先进后出**”的策略访问其中的元素即可，下面的遍历过程将展示我们如何使用和访问栈。

我们从 $v_1$ 开始遍历，因此开始时 $T = \langle v_1 \rangle$ ， $S = \langle \rangle$ ，这里用 $\langle \rangle$ 表示空序列。下一步找到 $v_1$ 的某个相邻顶点，比如说 $v_2$ （当然 $v_3$ 也可，但那将得到一个不同的遍历顺序），将 $v_2$ 加入 $T$ ，而因为 $v_1$ 有相邻结点，从而也将其加入 $S$ （按栈的术语称为“压入栈”），得到：

$$T = \langle v_1, v_2 \rangle \quad S = \langle v_1 \rangle$$

再找 $v_2$ 的相邻顶点（当然是除 $T$ 中顶点之外的相邻顶点），比如说 $v_4$ ，将 $v_4$ 加入 $T$ ，而将 $v_2$ 压入栈 $S$ ，得到：

$$T = \langle v_1, v_2, v_4 \rangle \quad S = \langle v_1, v_2 \rangle$$

再找 $v_4$ 的除在 $T$ 中顶点之外的相邻顶点，上图只有 $v_5$ ，加入 $v_5$ 到 $T$ ，而加入 $v_4$ 压入栈 $S$ 得到：

$$T = \langle v_1, v_2, v_4, v_5 \rangle \quad S = \langle v_1, v_2, v_4 \rangle$$



类似地，下一步 $v_5$ 在 $T$ 中顶点之外的相邻顶点只有 $v_6$ ，将 $v_6$ 加入到 $T$ ，而将 $v_5$ 加入压入栈 $S$ 得到：

$$T = \langle v_1, v_2, v_4, v_5, v_6 \rangle \quad S = \langle v_1, v_2, v_4, v_5 \rangle$$

好了，对于 $v_6$ ，它的所有相邻顶点都已经在 $T$ 中（也即 $v_6$ 没有尚未访问的相邻顶点），这时就需要**回溯**，也即要返回到上一个我们访问过的节点，这里就体现了栈的作用，我们上一个访问的节点现在在 $S$ 的最后（按栈的术语就是在栈顶），也即这时返回到 $v_5$ 。注意， $v_5$ 是最后入栈的顶点，而弹出的时候最先弹出，这就是栈的所谓的“**后进先出**”的访问策略。

现在考虑 $v_5$ ，注意到它也没有不在 $T$ 中的相邻顶点，那么它留在 $S$ 中已经没有用了，将其删除（按栈的术语称为“弹出” $v_5$ ），则得到：

$$T = \langle v_1, v_2, v_4, v_5, v_6 \rangle \quad S = \langle v_1, v_2, v_4 \rangle$$

现在再考虑在栈顶的顶点（ $S$ 的最后） $v_4$ ，同样它也没有不在 $T$ 中的相邻顶点，继续弹出 $v_4$ ，得到：

$$T = \langle v_1, v_2, v_4, v_5, v_6 \rangle \quad S = \langle v_1, v_2 \rangle$$

现在再考虑在栈顶的顶点 $v_2$ ，它还存在不在 $T$ 中的相邻顶点 $v_3$ ，因此得到：

$$T = \langle v_1, v_2, v_4, v_5, v_6 \rangle \quad S = \langle v_1 \rangle$$

现在再考虑刚弹出的 $v_2$ ，它还存在不在 $T$ 中的相邻顶点 $v_3$ ，因此将 $v_3$ 加入 $T$ ，得到：

$$T = \langle v_1, v_2, v_4, v_5, v_6, v_3 \rangle \quad S = \langle v_1 \rangle$$

现在考虑 $v_3$ ，由于它有不在 $T$ 中的相邻顶点 $v_7$ ，因此将 $v_7$ 加入 $T$ ，而将 $v_3$ 压入栈 $S$ ，得到：

$$T = \langle v_1, v_2, v_4, v_5, v_6, v_3, v_7 \rangle \quad S = \langle v_1, v_3 \rangle$$

现在对于 $v_7$ 也没有不在 $T$ 中的相邻顶点，因此再回溯到 $v_3$ ，由于 $v_3$ 已经没有不在 $T$ 中的相邻顶点，将 $v_3$ 弹出：

$$T = \langle v_1, v_2, v_4, v_5, v_6, v_3, v_7 \rangle \quad S = \langle v_1 \rangle$$

这时栈顶的顶点 $v_1$ 也没有不在 $T$ 中的相邻顶点，再将 $v_1$ 弹出。好了，现在栈空了，表明我们整个遍历的过程就结束了，而顶点的访问顺序就在 $T$ 中。

若使用**先广策略**，则除了用 $T$ 记住遍历结点的顺序之外，这回使用**队列** $Q$ 来帮助我们，实际上队列也是一个序列，只是我们采用与栈不同的访问策略访问其中的元素，即使用“**先进先出**”的策略访问队列。

从 $v_1$ 开始遍历，同样将 $v_1$ 加入到 $T$ ，同样将 $v_1$ 加入到 $Q$ ，得到：

$$T = \langle v_1 \rangle \quad Q = \langle v_1 \rangle$$

下一步我们考虑 $Q$ 的最前面顶点 $v_1$ ，将 $v_1$ 的所有（不在 $T$ 中的）相邻顶点 $v_2, v_3$ 加入到 $T$ （即表示 $v_1$ 的所有相邻顶点都已经访问），同时在 $Q$ 中删除 $v_1$ （ $v_1$ 出队列），而将 $v_2, v_3$ 加入到 $Q$ （入队列），得到：

$$T = \langle v_1, v_2, v_3 \rangle \quad Q = \langle v_2, v_3 \rangle$$



继续考虑 $Q$ 的最前面顶点 $v_2$ ，将 $v_2$ 的所有（不在 $T$ 中的）相邻顶点 $v_4, v_6$ 加入到 $T$ 和 $Q$ ，同时删除 $v_2$ ，得到：

$$T = \langle v_1, v_2, v_3, v_4, v_6 \rangle \quad Q = \langle v_3, v_4, v_6 \rangle$$

注意将 $v_4$ 和 $v_6$ 加入 $Q$ 时是加入（入队列）到 $Q$ 的尾部，而删除（出队列）的 $v_2$ 是在 $Q$ 的最前面顶点，这就是队列的所谓“先进先出”访问策略（ $v_2$ 是先入队列的顶点，因此也是最先出队列的顶点）。

继续考虑 $Q$ 的最前面顶点 $v_3$ ，将 $v_3$ 的所有不在 $T$ 中的相邻顶点 $v_7$ 加入 $T$ 和 $Q$ ，并在 $Q$ 中删除 $v_3$ ，得到：

$$T = \langle v_1, v_2, v_3, v_4, v_6, v_7 \rangle \quad Q = \langle v_4, v_6, v_7 \rangle$$

继续考虑 $Q$ 的最前面顶点 $v_4$ ，将 $v_4$ 的所有不在 $T$ 中的相邻顶点 $v_5$ 加入 $T$ 和 $Q$ ，并在 $Q$ 中删除 $v_4$ ，得到：

$$T = \langle v_1, v_2, v_3, v_4, v_6, v_7, v_5 \rangle \quad Q = \langle v_6, v_7, v_5 \rangle$$

继续考虑 $RQ$ 的最前面顶点 $v_6$ ，它没有不在 $T$ 中的相邻顶点，直接在 $Q$ 中删除，得到：

$$T = \langle v_1, v_2, v_3, v_4, v_6, v_5 \rangle \quad Q = \langle v_7, v_5 \rangle$$

继续考虑 $RQ$ 的最前面顶点 $v_7$ ，它也没有不在 $T$ 中的相邻顶点，直接在 $Q$ 中删除，得到：

$$T = \langle v_1, v_2, v_3, v_4, v_6, v_5 \rangle \quad Q = \langle v_5 \rangle$$

继续考虑 $RQ$ 的最前面顶点 $v_5$ ，它也没有不在 $T$ 中的相邻顶点，直接在 $Q$ 中删除，得到了空队列，这表明所有的顶点已经访问到，遍历过程结束，而节点的遍历顺序也在 $T$ 中。

**练习 2.3.8** 读者应通过上述例子理解先深搜索和先广搜索的区别，并将上述例子中描述的方法提炼为一般的方法，并自己画一个图进行先深搜索和先广搜索。进一步，读者可通过这里例子，理解栈和队列的区别，为以后的数据课程学习打下基础。

## 2.4 哈夫曼树

这一章的最后，我们利用一种特殊的根树，即所谓的哈夫曼(Huffman)树考虑一类实际问题的解决。假定现在要在网络传送许多英文文件，需要对文件的内容进行编码。我们知道，对于英文而言，不是每个字母都是等概率出现的，例如 $c, e$ 这些字母通常出现得最多，而 $z, x$ 这些字母则出现得很少，所以如果我们都用同样长的二进制编码所有的英文字母，那么对文件编码得到二进制文件长度就不是最优的。

为简单起见，假设我们要传送的文件中只含有 $a, b, c, d, e, f, g, h$ 等八个字母，每个字母在文件中出现的概率分别为：

$a$ : 15%	$b$ : 12%	$c$ : 25%	$d$ : 8%
$e$ : 20%	$f$ : 6%	$g$ : 8%	$h$ : 6%

如果我们使用相同长度的二进制对这八个字母进行编码，那么每个字母需要三位二进制，从而若要传送长度为1万个字母的英文文件，那么总共要传送3万个二进制数。

下面我们利用Huffman树对这八个字母做一种不等长的编码，使得当要传送1万个字母时，传送的二进制数可以少于3万，从而节省了网络传送的费用。为此，我们先引入有关概念。

**定义 2.4.1** 给定有 $k$ 个叶子节点的二元树 $T = (V, E)$ , 若 $k$ 个叶子分别赋以非负实数权 $w_1, w_2, \dots, w_k$ , 并设这 $k$ 个叶子对应的高度分别是 $l_1, l_2, \dots, l_k$ , 则称

$$w(T) = \sum_{i=1}^k l_i * w_i$$

为二元树 $T$ 的**带权外部通路总长**。

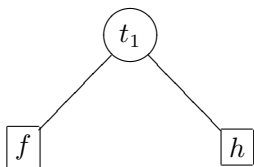
称在有 $k$ 个叶子节点并分别赋以非负实数权的二元树中, 具有最小的带权外部通路总长的二元数为**最优二元树**。

回到前面的问题, 我们将要编码的字母看作二元树的叶子, 字母在文件中出现的概率作为赋予叶子的权, 通过构造一个最优二元树可以确定这些字母的一个比较节省网络传送费用的二进制编码方案。Huffman给出了一个构造最优二元树的算法, 从而我们将利用Huffman算法构造出来的最优二元树称为**Huffman树**。

**例子 2.4.2** 我们利用上面的例子来说明Huffman算法的基本思想。首先将字母的权从小到大排序(省略百分比, 实际上, Huffman算法对于权也没有特别的要求, 只要非负即可):

$$6(f) \quad 6(h) \quad 8(g) \quad 8(d) \quad 12(b) \quad 15(a) \quad 20(e) \quad 25(c)$$

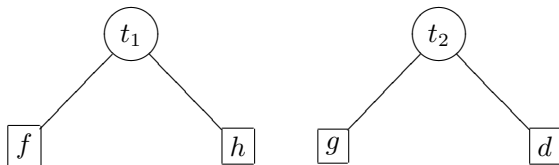
然后我们开始构造二元树。取所有权中最小的两个权, 以这两个权所对应的字母作为儿子, 新添加一个节点 $t_1$ 作为根, 得到一个子树(注意, 我们特意用方框表示叶子节点, 而用圆框表示分支节点):



为了便于后面的构造, 我们令 $t_1$ 的赋为它的两个儿子节点的权的和, 即为12, 而且在上面的权序列中, 将 $f$ 和 $h$ 的权删除, 而将 $t_1$ 的权加入, 并仍保持从小到大排序, 即得到如下的权序列:

$$8(g) \quad 8(d) \quad 12(t_1) \quad 12(b) \quad 15(a) \quad 20(e) \quad 25(c)$$

下一步跟前面类似, 我们再选两个权最小的字母 $g$ 和 $d$ 作为儿子, 并新添加一个节点 $t_2$ 作为根, 得到一个子树(为清楚起见, 我们将先前构造的树 $t_1$ 也画在下面):

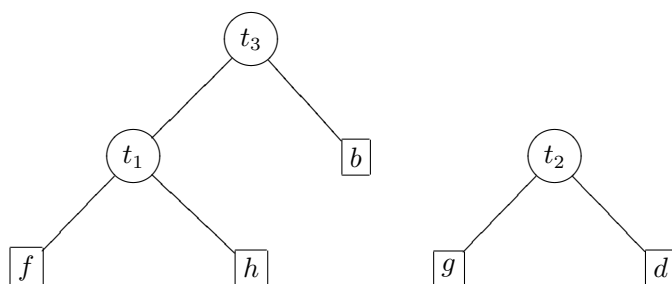


同样地, 在权序列中删除 $g$ 和 $d$ 的权, 并令 $t_2$ 的权为它们的和(即16), 并加入到权序列, 保持从小到大的序, 即得到:

$$12(t_1) \quad 12(b) \quad 15(a) \quad 16(t_2) \quad 20(e) \quad 25(c)$$

这时权最小的是 $t_1$ 和 $b$ 的权, 也没有关系, 我们将 $t_1$ 看作一个单独的字母好了, 取这两个权对应的节点(这时称节点更为恰当了)作为儿子, 添加一个新节点 $t_3$ 作为根, 也就是说, 现在我们得到如下的

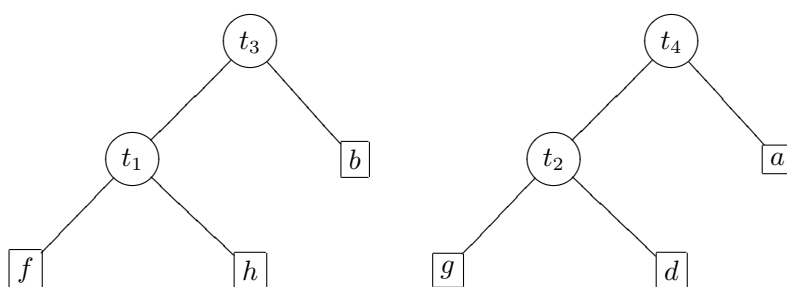
树（准确地说，是森林）。注意，以 $t_3$ 是以 $t_1$ 作为左儿子还是以 $b$ 作为左儿子没有什么关系，随便选择即可，下面为了画图的方便，选择 $t_1$ 作为左儿子：



同样地，在权序列中删除 $t_1$ 和 $b$ 的权，并令 $t_3$ 的权为它们的和（即24），并加入到权序列，保持从小到大的序，即得到：

$$15(a) \quad 16(t_2) \quad 20(e) \quad 24(t_3) \quad 25(c)$$

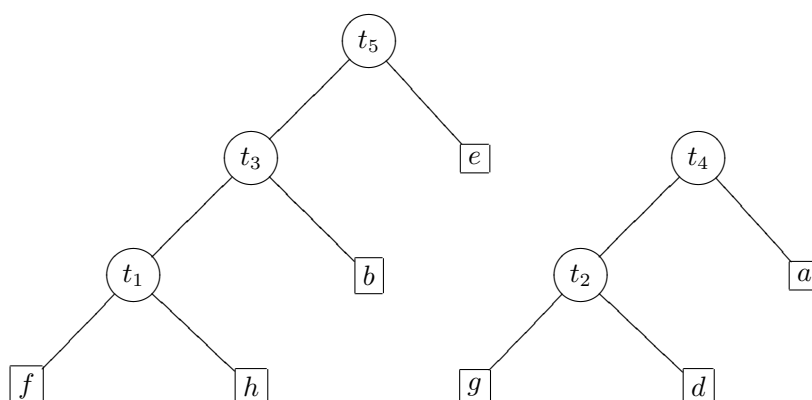
继续选择两个最小的权对应的节点（即 $a$ 和 $t_2$ ）作为儿子，添加一个新节点 $t_4$ 作为根，也就是说，得到如下的森林：



删除 $a$ 和 $t_2$ 对应的权，加入 $t_4$ 对应的权（即 $a$ 和 $t_2$ 的权之和31），并保持权序列的从小到大顺序得到：

$$20(e) \quad 24(t_3) \quad 25(c) \quad 31(t_4)$$

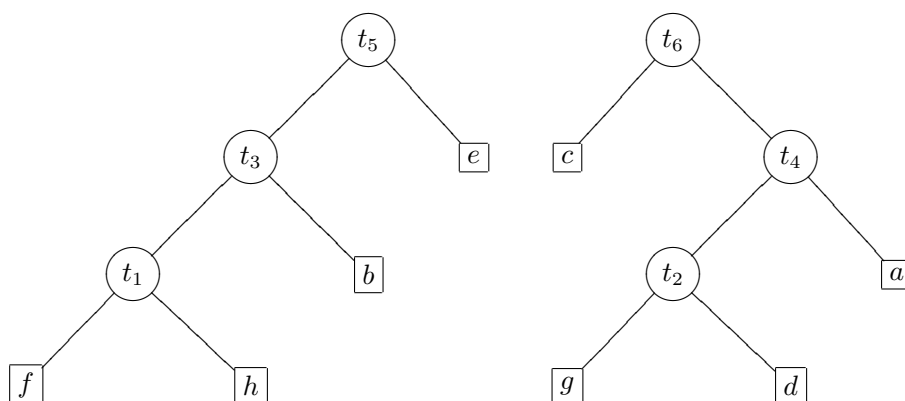
继续选择两个最小的权对应的节点（即 $e$ 和 $t_3$ ）作为儿子，添加新节点 $t_5$ 作为根，得到如下森林：



删除 $e$ 和 $t_3$ 对应的权，加入 $t_5$ 对应的权（即 $e$ 和 $t_3$ 的权之和44），保持权序列的从小到大顺序得到：

$$25(c) \quad 31(t_4) \quad 44(t_5)$$

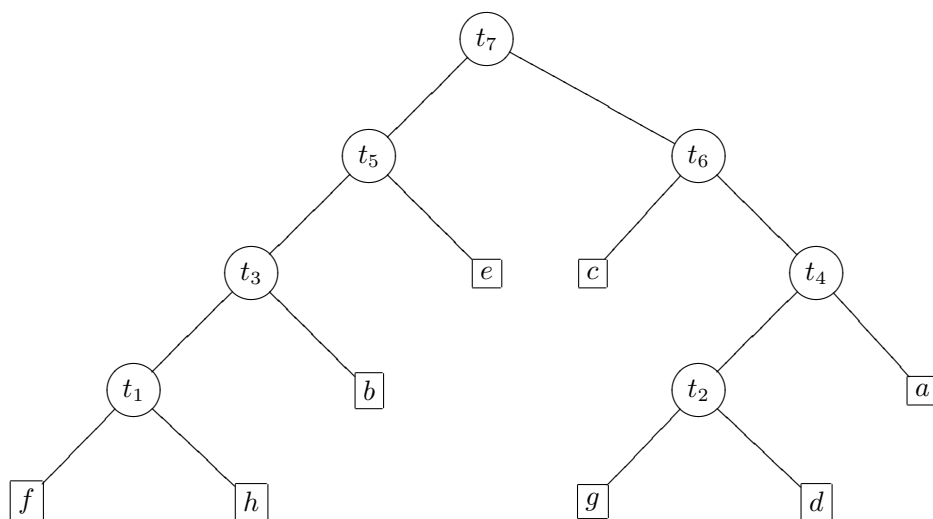
构造一个新的节点 $t_6$ ，以 $c$ 和 $t_4$ 对应的节点作为儿子，得到：



删除 $c$ 和 $t_4$ 对应的权，加入 $t_6$ 对应的权（即 $c$ 和 $t_4$ 对应的权之和56），保持权序列的从小到大顺序得到：

$$44(t_5) \quad 56(t_6)$$

构造一个新的节点 $t_7$ ，以 $t_5$ 和 $t_6$ 对应的节点作为儿子，得到：



在权序列中删除 $t_5$ 和 $t_6$ 对应的权之后，再加入 $t_7$ 对应的权（即 $t_5$ 和 $t_6$ 的权之和100）后得到的权序列中只有一个节点（即 $t_7$ ），这表明我们的最优二元树的构造已经完成。但要注意，[这个最优二元树 \$T\$ 的外部通路总长并不是 \$t\_7\$ 的权](#)，而是：

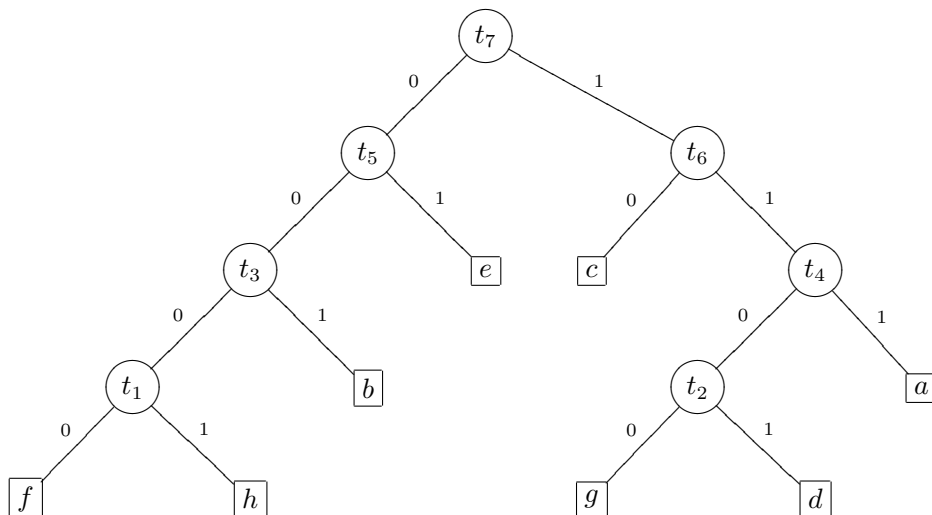
$$w(T) = 6 * 4 + 6 * 4 + 8 * 4 + 8 * 4 + 12 * 3 + 15 * 3 + 20 * 2 + 25 * 2 = 283$$

**练习 2.4.3** 根据上述例子的描述，抽象出Huffman算法的一般描述（或者说，根据这个例子，理解戴一奇教材p57页对Huffman算法的描述）。

**定理 2.4.4** 由Huffman算法得到的二元树是最优二元树。

**证明 略**，参见戴一奇教材p58页的定理3.6.1。 □

**例子 2.4.5** 继续上面的例子, 根据上面构造得到的Huffman树, 我们可以对要传送的字母 $a, b, c, d, e, f, g, h$ 进行编码。编码方法很简单, 对树的每一条边赋一个二进制数: 如果某条边是从父亲节点连向左儿子节点, 则赋二进制数0; 如果是从父亲节点连向右儿子节点, 则赋二进制数1。对于上面的Huffman树, 我们得到:



我们令每个叶子节点对应的字母的编码就是从根节点到该叶子节点所经过的边上的二进制数构成的串, 也即得到:

$a$  的编码是: 111       $b$  的编码是: 001       $c$  的编码是: 10       $d$  的编码是: 1101  
 $e$  的编码是: 01       $f$  的编码是: 0000       $g$  的编码是: 1100       $h$  的编码是: 0001

与前面所给出的每个字母的出现频率相比较, 不难发现, 出现频率越高的字母的编码越短。因此, 如果传送具有1万个字母的英文文件, 该文件只出现这8个字母, 而且每个字母出现的概率正如上面所给出的那样, 也就是说, 在1万个字母中,  $a$ 大概出现15%, 也即1500个,  $b$ 大概出现12%, 也即1200个等等, 那么传送这1万个字母英文文件需要传送的二进制编码数目是:

$$\begin{aligned} W &= 600 * 4 + 600 * 4 + 800 * 4 + 800 * 4 + 1200 * 3 + 1500 * 3 + 2000 * 2 + 2500 * 2 \\ &= w(T) * 10000 / 100 = 28300 \end{aligned}$$

比使用等长的三位编码要传送的二进制数3万要少1700个。

要使用不等长的二进制编码进行传输, 我们还需要解决一个问题, 即传送信息的分割, 也即那几个二进制位是代表一个字符。很显然如果使用等长编码的话, 例如3位编码, 那么文件接受方在接收到数据之后, 每三位三位进行译码即可, 但如果是使用不等长的编码进行传送, 那么接受方接收到数据之后该如何译码呢? 也就是说, 我们需要一个能够正确分割接收到的信息并进行译码的方法。解决这个问题的方法就是让对要传送的字符的编码满足**前缀码**的要求。

**定义 2.4.6** 设 $a_1 a_2 \cdots a_n$ 是长度为 $n$ 的串, 我们称其子串 $a_1, a_1 a_2, a_1 a_2 a_3, \cdots, a_1 a_2 \cdots a_{n-1}$ 分别为该符号串长度为1, 2, 3,  $\cdots, n-1$ 的**前缀**。也就是说, 符号串 $\beta$ 是符号串 $\alpha$ 的**前缀**, 当且仅当 $\beta$ 是 $\alpha$ 的从第一个字符开始的子符号串。

**定义 2.4.7** 设  $A = \{\beta_1, \beta_2, \dots, \beta_n\}$  是一些符号串构成的集合, 如果  $A$  满足, 对任意的  $i \neq j, 1 \leq i, j \leq n$ , 都有  $\beta_i$  不是  $\beta_j$  的前缀, 也即  $A$  中的任意两个符号串都不互为前缀, 则称  $A$  是**前缀码**。进一步, 若前缀码  $A$  中任意的串  $\beta_i$  都只有两种符号构成, 则称  $A$  是**二元前缀码**。

**例子 2.4.8** 对于串 110010101, 其前缀有 1, 11, 110, 1100, 11001 等等, 而 10010 虽然是它的子串, 但不是它的前缀。下面的符号串集合不是前缀码:

$$A = \{11, 100, 1101, 001, 0100, 0111, 0011\}$$

因为其中 11 是 1101 的前缀, 001 是 0011 的前缀。而我们前面根据 Huffman 树得到的编码集 (符号串集)

$$H = \{111(a), 001(b), 10(c), 1101(d), 01(e), 0000(f), 1100(g), 0001(h)\}$$

就是前缀码。一般地, 如果按照上面的编码方法, 即将连向左儿子的边编码为 0, 右儿子的边编码为 1, 叶子结点的编码为根到该叶子结点的路径上的二进制串, 那么任意一棵二元树的叶子结点的编码构成的集合一定是前缀码, 因为根到任何叶子结点的路径都是惟一的, 绝不会是根到另外一个叶子结点的一部分。

对于二元前缀码, 由于任何一个码都不是另外一个码的前缀, 从而对于一些字母的编码, 如果传送正确的话, 那么在接收方都能正确地译码, 例如我们要传送的字符串是 *faceaddheadgeaf*, 根据上面的编码我们要传送如下的二进制串:

$$000011110011111101110100010111111011100011110000$$

接收方只要知道每个字母的编码, 就能正确地将上述二进制串翻译回英文字符串。例如, 看到第一个二进制 0 时, 那么可能的字符是 *b, e, f, h*, 再看到下一个 0 时, 可能的字符只有 *b, f, h*, 再看到一个 0, 可能的字符剩下 *f, h*, 再看到一个 0, 就只有 *f* 了, 而且由于 0000 不会是哪任何其他字符编码的前缀, 因此 0000 只可能译码成 *f*。

如果用于编码字符的不是前缀码, 那么就无法正确分割信息并进行译码, 例如, 若利用上述编码集  $A$  进行编码, 传送如下的二进制串:

$$1101110010011$$

那么当接收方接收到 11 时, 它不知道是将 110111 分割成 110111 呢, 还是分割成 110111, 而使用前缀码进行编码则只可能存在惟一的一种分割方式。

**作业:** 戴一奇教材[2]p66习题三的第1, 4(a), 14(a)题。

## 第三章 路径问题

这一章集中考虑几个与图的路径有关的几个问题, 包括有向图中任意两点之间的最短路径, 图的最小生成树, 以及有向网络关键路径问题, 这些问题都是在实际应用中经常遇到的问题, 例如求网络的关键路径是项目管理 (包括软件开发项目) 中的重要问题之一。

### 3.1 最短路径

**定义 3.1.1** 给定有向图  $G = (V, E)$ , 设  $V = \{v_1, v_2, \dots, v_n\}$ 。如果给每一条有向边  $e = \langle v_i, v_j \rangle$  赋一个非负实数权  $w_{ij}$ , 则称图  $G$  为**有向网络**。

假定我们这一节考虑的有向网络都是简单有向图, 即没有多重边, 也没有环。

**定义 3.1.2** 给定有向网络  $G = (V, E)$ , 设  $V = \{v_1, v_2, \dots, v_n\}$ 。定义  $G$  的**距离矩阵**  $D_{n \times n} = [d_{ij}]$ , 其中

$$d_{ij} = \begin{cases} w_{ij} & \text{若存在有向边 } e = \langle v_i, v_j \rangle \in E \\ \infty & \text{否则} \end{cases}$$

**定义 3.1.3** 给定有向网络  $G = (V, E)$ , 设  $V = \{v_1, v_2, \dots, v_n\}$ 。定义有向路径  $\Gamma = v_{i_1}v_{i_2}\dots v_{i_k}$  的**带权路径长度**为:

$$d(\Gamma) = \sum_{j=1}^{k-1} w_{i_j i_{j+1}}$$

若定点  $v_i$  可达  $v_j$ , 则称  $v_i$  到  $v_j$  的所有有向路径中具有最小带权路径长度的路径为  $v_i$  到  $v_j$  的**最短路径**,  $v_i$  到  $v_j$  的最短路径的带权路径长度称为  $v_i$  到  $v_j$  的**最短距离**。

显然我们有:

**定理 3.1.4** 给定有向网络  $G = (V, E)$ , 设  $V = \{v_1, v_2, \dots, v_n\}$ 。若路径  $v_1, v_2, \dots, v_{k-1}, v_k$  是  $v_1$  到  $v_k$  的最短路径, 则路径  $v_1, v_2, \dots, v_{k-1}$  是  $v_1$  到  $v_{k-1}$  的最短路径。

根据这一点, Dijkstra 设计了一个求  $v_1$  到其他各顶点的最短距离。我们以一个例子说明 Dijkstra 算法的基本思想。

**例子 3.1.5** 考虑戴一奇教材p25图2.23所示的有向网络。其距离矩阵为：

$$\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{bmatrix} \infty & 7 & 1 & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 6 & \infty & \infty & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & \infty \end{bmatrix}$$

我们利用Dijkstra算法求 $v_1$ 到各点的最短路径。需要三个集合（或数组）来记录求解过程中的信息。

(1)  $\bar{S}$ 是记录尚未求得最短路径顶点的集合；

(2)  $U = (u_i)_{1 \leq i \leq n}$ 是以顶点编号（顶点 $v_i$ 的编号是 $i$ ）为下标的数组， $u_i$ 记录 $v_1$ 到 $v_i$ 的最短距离；

(3)  $Q = (q_i)_{1 \leq i \leq n}$ 也是以顶点编号为下标的数组， $q_i$ 记录 $v_1$ 到 $v_i$ 的最短路径中的 $v_i$ 的直接前趋。

开始时，自然 $\bar{S} = \{2, 3, 4, 5, 6\}$ （表示 $v_2$ 到 $v_6$ 都没有求得最短路径），而 $u_1 = 0$ ， $u_i = d_{1i}$ （ $2 \leq i \leq n = 6$ ），也即：

$$U = \langle \infty, 7, 1, \infty, \infty, \infty \rangle$$

而对任意的 $1 \leq i \leq n$ ，都令 $q_i = 1$ ，即

$$Q = \langle 1, 1, 1, 1, 1, 1 \rangle$$

在 $U$ 中那些上未求得最短路径顶点的编号所对应的 $u_i$ 值中的最小者 $u_j$ ，即令

$$u_j = \min_{i \in \bar{S}} u_i$$

具体到上面的有向网络的距离矩阵，我们得到最小的是 $u_3 = 1$ ，这表示 $v_1$ 到 $v_3$ 的最短路径就是边 $e = \langle v_1, v_3 \rangle$ ，其长度就是1，因此将3从 $\bar{S}$ 中删除，而且修改 $U$ 中的 $u_i$ 值，只针对 $v_3$ 的邻接顶点即可，也就是说， $v_1$ 原来不能到达的顶点，现在可能可经过 $v_3$ 到达了，或者说虽然原来 $v_1$ 能到达的顶点的路径长度可能比以 $v_3$ 为中间顶点的路径长度要短。

在这里，我们看到 $v_3$ 的邻接顶点有 $v_2, v_5, v_6$ ，比较 $u_2 = 7$ 和 $u_3 + d_{32} = 7$ ，因此仍保持 $u_2$ 的值为7；比较 $u_5 = \infty$ 和 $u_3 + d_{35} = 3$ ，因此将 $u_5$ 修改为3，相应地将 $q_5$ 的值改为3（即 $v_1$ 经过 $v_3$ 可以比较快地到达 $v_5$ ）；比较 $u_6 = \infty$ 和 $u_3 + d_{36} = 8$ ，因此将 $u_6$ 修改为8，相应地将 $q_6$ 的值也修改为3，因此得到：

$$\bar{S} = \{2, 4, 5, 6\} \quad U = \langle 0, 7, 1, \infty, 3, 8 \rangle \quad Q = \langle 1, 1, 1, 1, 3, 3 \rangle$$

下一步我们考虑 $u_2, u_4, u_5, u_6$ 中的最小者（注意不要再考虑 $u_3$ ，因为 $v_3$ 已经不属于 $\bar{S}$ ），显然最小者是 $u_5 = 3$ ，因此将5从 $\bar{S}$ 中删除，并修改 $v_5$ 的邻接顶点 $v_2$ 和 $v_4$ 所对应的 $u_2$ 和 $u_4$ 的值，即比较 $u_2 = 7$ 和 $u_5 + d_{52} = 6$ ，将 $u_2$ 的值修改为6， $q_2$ 的值修改为5；比较 $u_4 = \infty$ 和 $u_5 + d_{54} = 8$ ，将 $u_4$ 的值修改为8， $q_4$ 的值修改为5，因此得到：

$$\bar{S} = \{2, 4, 6\} \quad U = \langle 0, 6, 1, 8, 3, 8 \rangle \quad Q = \langle 1, 5, 1, 5, 3, 3 \rangle$$

下一步考虑 $u_2, u_4, u_6$ 中的最小者，即 $u_2 = 6$ ，将2从 $\bar{S}$ 中删除，并修改 $v_2$ 的邻接点 $v_4$ 和 $v_6$ 所对应的 $u_4$ 和 $u_6$ 的值，即比较 $u_4 = 8$ 和 $u_2 + d_{24} = 10$ ，仍保持 $u_4$ 和 $q_4$ 的值；比较 $u_6 = 8$ 和 $u_2 + d_{26} = 7$ ，修改 $u_6$ 的值为7， $q_6$ 的值为2，因此得到：

$$\bar{S} = \{4, 6\} \quad U = \langle 0, 6, 1, 8, 3, 7 \rangle \quad Q = \langle 1, 5, 1, 5, 3, 2 \rangle$$



最后考虑 $u_4, u_6$ 中的最小者, 即 $u_6 = 7$ , 将6从 $\bar{S}$ 中删除, 并修改 $v_6$ 的邻接顶点, 但 $v_6$ 的邻接顶点只有 $v_5$ , 而 $v_5$ 已经考虑, 所以无需修改 $U$ 和 $Q$ 。最后考虑删除 $u_4$ 时,  $u_4$ 的邻接顶点都已经考虑过, 因此这时不会对 $U$ 和 $Q$ 产生影响, 从而最后得到的 $U$ 和 $Q$ 如下:

$$U = \langle 0, 6, 1, 8, 3, 7 \rangle \quad Q = \langle 1, 5, 1, 5, 3, 2 \rangle$$

$U$ 给出了 $v_1$ 到各顶点之间的最短距离, 即 $v_1$ 到 $v_i (2 \leq i \leq 6)$ 的最短距离是 $u_i$ , 而 $Q$ 可给出 $v_1$ 到各顶点的最短路径, 例如 $v_1$ 到 $v_2$ 的最短路径可这样得到:  $q_2$ 的值是5, 而 $q_5$ 的值是3,  $q_3$ 的值是1, 因此 $v_1$ 到 $v_2$ 的最短路径是 $v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_2$ 。总之,  $v_1$ 到各顶点之间的最短路径和最短距离如下:

$v_1$ 到 $v_2$ 的最短距离是 6,	相应的路径是	$v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_2$
$v_1$ 到 $v_3$ 的最短距离是 1,	相应的路径是	$v_1 \rightarrow v_3$
$v_1$ 到 $v_4$ 的最短距离是 8,	相应的路径是	$v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_4$
$v_1$ 到 $v_5$ 的最短距离是 3,	相应的路径是	$v_1 \rightarrow v_3 \rightarrow v_5$
$v_1$ 到 $v_6$ 的最短距离是 7,	相应的路径是	$v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_2 \rightarrow v_6$

**练习 3.1.6** 根据上述例子理解戴一奇教材p25对Dijkstra算法的抽象描述。

显然利用Dijkstra算法, 我们可以求得任意两个顶点之间的最短距离。如果要直接求有向网络中任意两个顶点之间的最短距离, 与利用邻接矩阵计算有向图的可达矩阵类似, 我们也可通过修改矩阵乘法计算任意两个顶点之间的最短距离, 进一步, 也可通过修改计算可达矩阵的Warshall算法计算任意两个顶点之间的最短距离。

**例子 3.1.7** 同样考虑戴一奇教材p25图2.23所示的有向网络。其距离矩阵 $D$ 为:

$$\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} \begin{bmatrix} \infty & 7 & 1 & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 6 & \infty & \infty & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & \infty \end{bmatrix}$$

我们通过修改矩阵乘法计算任意两个顶点之间的最短距离。

我们知道, 距离矩阵 $D = [d_{ij}]$ 中的元素 $d_{ij}$ 给出了 $v_i$ 到 $v_j$ 的长度为1的带权路径长度, 而计算矩阵乘法 $D \times D = D^{(2)} = [d_{ij}^{(2)}]$ 时 $d_{ij}^{(2)}$ 的值是通过考察对任意的顶点 $v_k$ ,  $v_i$ 经过 $v_k$ 再到 $v_j$ 的长度为2的路径, 如果要考虑最小的带权路径长度, 我们在计算 $d_{ij}^{(2)}$ 时可使用如下方法:

$$d_{ij}^{(2)} = \min\{d_{ij}, \min_{1 \leq k \leq n} (d_{ik} + d_{kj})\}$$

这里 $+$ 是普通实数加法, 但约定对任意的实数 $r$ 有 $r + \infty = \infty$ 。实际上,  $d'_{ij} = \min_{1 \leq k \leq n} (d_{ik} + d_{kj})$ 是 $v_i$ 到 $v_j$ 的长度为2的最小带权路径长度, 从而 $d_{ij}^{(2)} = \min\{d_{ij}, d'_{ij}\}$ 就是 $v_i$ 到 $v_j$ 长度不超过2的最小带权路径长度。类似地计算 $D^{(3)} = D^{(2)} \times D$ 等等, 一直到 $D^{(n-1)}$ , 得到长度不超过 $n-1$ 的最小带权路径长度。显然, 任意两个顶点之间的最短路径长度不会超过 $n-1$ , 因此 $D^{(n-1)}$ 将给出任意两个顶点之间的最短距离。

根据这个方法, 我们针对上述有向网络进行计算, 得到:

$$D^{(2)} = \begin{bmatrix} \infty & 7 & 1 & 11 & 3 & 8 \\ \infty & \infty & \infty & 4 & 4 & 1 \\ \infty & 5 & \infty & 7 & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & 4 \\ \infty & 6 & \infty & 8 & 3 & \infty \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} \infty & 6 & 1 & 8 & 3 & 8 \\ \infty & 7 & \infty & 4 & 4 & 1 \\ \infty & 5 & \infty & 7 & 2 & 6 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & 7 & 4 \\ \infty & 6 & \infty & 8 & 3 & 7 \end{bmatrix}$$

$$D^{(4)} = D^{(5)} = \begin{bmatrix} \infty & 6 & 1 & 8 & 3 & 7 \\ \infty & 7 & \infty & 4 & 4 & 1 \\ \infty & 5 & \infty & 7 & 2 & 6 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & 7 & 4 \\ \infty & 6 & \infty & 8 & 3 & 7 \end{bmatrix}$$

最后矩阵 $D^{(5)}$ 的对角线数字, 如果不是 $\infty$ , 则意味着 $v_i$ 到 $v_i$ 有一条最短的回路, 不过这对于求两顶点之间的最短距离没有什么意义, 因为我们可以约定任意顶点到它自己的最短距离是0。

**例子 3.1.8** 上述计算虽然直接, 但比较繁琐, Warshall 的计算可达矩阵的方法也可用来计算任意两个顶点之间的最短距离。Warshall 算法的基本思想是:

第 $k$ 次循环计算的 $d_{ij}^{(k)}$ 是顶点 $v_i$ 到 $v_j$ 之间可能经过 $v_1, v_2, \dots, v_k$ 这 $k$ 个顶点的路径中带权长度最短的路径长度, 从而第 $n$ 次循环时,  $d_{ij}^{(n)}$ 将给出顶点 $v_i$ 到 $v_j$ 之间可能经过所有顶点的路径中带权长度最短的路径长度, 也即 $d_{ij}^{(n)}$ 是顶点 $v_i$ 到 $v_j$ 之间的最短距离。

同样考虑戴一奇教材p25图2.23所示的有向网络。其距离矩阵 $D$ 为:

$$\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} \begin{bmatrix} \infty & 7 & 1 & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 6 & \infty & \infty & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & \infty \end{bmatrix}$$

第一次循环是对任意的 $1 \leq i \leq n, 1 \leq j \leq n$ , 用 $d_{ij}$ 与 $d_{i1} + d_{1j}$ 之间的小者修正 $d_{ij}$ , 由于上述有向网

络对任意的 $i$ 都有 $d_{i1} = \infty$ ，因此第一次循环不会对距离矩阵作任何修改，即

$$D^{(1)} = D = \begin{bmatrix} \infty & 7 & 1 & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 6 & \infty & \infty & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & \infty \end{bmatrix}$$

第二次循环是对任意的 $1 \leq i \leq n, 1 \leq j \leq n$ ，用 $d_{ij}^{(1)}$ 与 $d_{i2}^{(1)} + d_{2j}^{(1)}$ 之间的小者修正 $d_{ij}$ 得到 $d_{ij}^{(2)}$ ，为此，只要将第2行中非 $\infty$ 的 $d_{2j}^{(1)}$ 与第2列中非 $\infty$ 的 $d_{i2}^{(1)}$ 相加，并与 $d_{ij}^{(1)}$ 比较即可得到 $d_{ij}^{(2)}$ ，也即，令：

$$d_{ij}^{(2)} = \min\{d_{ij}^{(1)}, d_{i2}^{(1)} + d_{2j}^{(1)}\}$$

针对上述有向网络，我们得到：

$$D^{(2)} = \begin{bmatrix} \infty & 7 & 1 & 11 & \infty & 8 \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 6 & \infty & 10 & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & 4 \\ \infty & \infty & \infty & \infty & 3 & \infty \end{bmatrix}$$

$$D^{(4)} = D^{(3)} = \begin{bmatrix} \infty & 7 & 1 & 11 & 3 & 8 \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 6 & \infty & 10 & 2 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & 4 \\ \infty & \infty & \infty & \infty & 3 & \infty \end{bmatrix}$$

$$D^{(5)} = \begin{bmatrix} \infty & 6 & 1 & 8 & 3 & 7 \\ \infty & \infty & \infty & 4 & \infty & 1 \\ \infty & 5 & \infty & 7 & 2 & 6 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & \infty & 4 \\ \infty & 6 & \infty & 8 & 3 & 7 \end{bmatrix}$$

$$D^{(6)} = \begin{bmatrix} \infty & 6 & 1 & 8 & 3 & 7 \\ \infty & 7 & \infty & 4 & 4 & 1 \\ \infty & 5 & \infty & 7 & 2 & 6 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & \infty & 5 & 7 & 4 \\ \infty & 6 & \infty & 8 & 3 & 7 \end{bmatrix}$$

得到的结果 $D^{(6)}$ 与上面得到的 $D^{(5)}$ 相同，都给出了任意两顶点之间的最短距离，但显然，使用Warshall算法计算更为简单。

## 3.2 最小生成树

上一节考虑的是有向网络, 类似地, 我们也可定义无向网络:

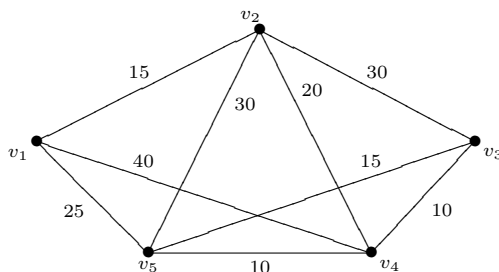
**定义 3.2.1** 给定无向图  $G = (V, E)$ , 设  $V = \{v_1, v_2, \dots, v_n\}$ 。如果给每一条有向边  $e = \langle v_i, v_j \rangle$  赋一个非负实数权  $w_{ij}$ , 则称图  $G$  为 **无向网络**。

同样我们考虑的无向网络都是简单无向图, 即没有多重边, 也没有环。这一节要考虑的主题是求无向连通网络的最小生成树:

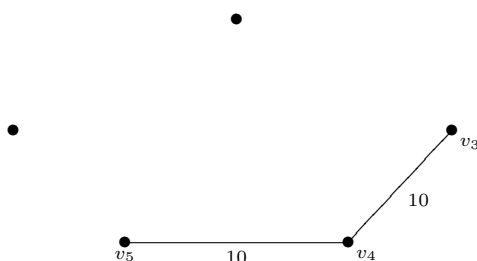
**定义 3.2.2** 无向连通网络  $G$  的所有生成树中树枝权值之和最小的生成树称为  $G$  的 **最小生成树** (或最短树)。

Kruskal 提供了一个求无向连通网络  $G = (V, E)$  的最小生成树的算法, 其基本思想是将  $G$  的所有边按权值从小到大排序, 然后依次考察每条边作为最小生成树  $T$  的候选树枝, 如果某条边不与已经选中的边构成回路, 则合乎要求, 否则就被放弃而考虑下一条边, 直到所有的边考虑完毕, 或更简单地, 只要  $T$  中已经有  $|V| - 1$  条边即可终止算法。

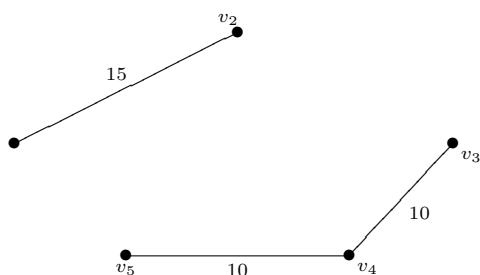
**例子 3.2.3** 我们以例子来说明 Kruskal 算法。考虑戴一奇教材 P59 的图 3.24 的无向网络:



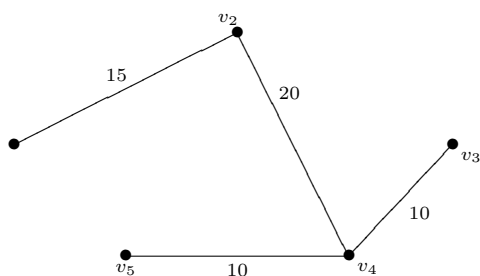
根据 Kruskal 算法的选边的顺序如下。首先选择权最小的边  $(v_5, v_4)$ , 然后选择边  $(v_4, v_3)$ , 这两个边的权都是 10, 而且不构成回路, 我们初步得到的生成树:



下一步考虑的边的权是 15, 这时如果加入边  $(v_5, v_3)$ , 则会构成回路, 所以下一步只能加入边  $(v_1, v_2)$ , 得到的生成树如下:



下一步考虑的边的权是20，即加入边 $(v_2, v_4)$ ，这时就已经得到了 $4 = |V| - 1$ 条边，从而算法终止，得到如下的最小生成树：

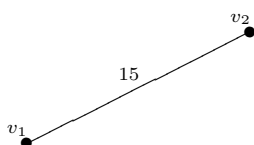


Prim提供了另一种得到最小生成树的算法，其基本思想是：首先任选一个顶点 $v_0$ 构成顶点集 $V'$ ，然后不断在 $V - V'$ 中选一条到 $V'$ 中某顶点（例如是 $v$ ）最短的边（例如是 $(v, u)$ ），也即

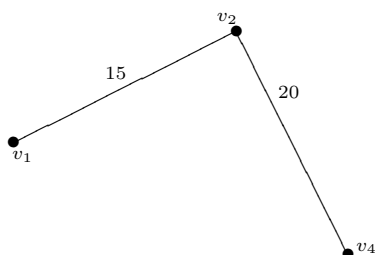
$$w(v, u) = \min_{t \in V', w \in V - V'} w(t, w)$$

将边 $(v, u)$ 加入到树 $T$ ，并令 $V' = V' \cup \{u\}$ ，直到 $V' = V$ 。

**例子 3.2.4** 同样考虑戴一奇教材P59的图3.24的无向网络，使用Prim算法选择顶点和边的顺序如下：假定第一步选择顶点 $v_1$ ，令 $V' = \{v_1\}$ ，与 $v_1$ 距离最短的边是 $(v_1, v_2)$ ，加入边 $(v_1, v_2)$ 到生成树 $T$ ，加入顶点 $v_2$ 到 $V'$ ，得到：

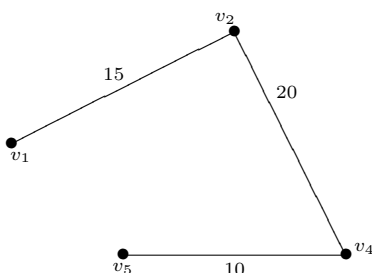


显然与 $v_1, v_2$ 距离最短的边是 $(v_2, v_4)$ ，加入边 $(v_2, v_4)$ 到生成树 $T$ ，加入顶点 $v_4$ 到 $V'$ ，得到：

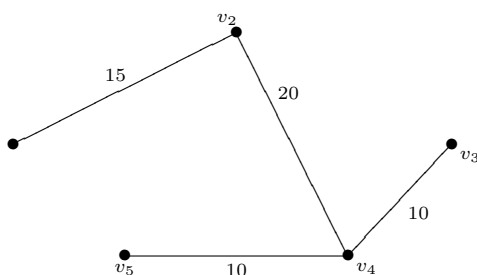


下一步与 $v_1, v_2, v_4$ 距离最短的边是 $(v_5, v_4)$ （或 $(v_3, v_4)$ 也类似），加入边 $(v_5, v_4)$ 到生成树 $T$ ，加入顶

点 $v_5$ 到 $V'$ , 得到:



下一步与 $v_1, v_2, v_4, v_5$ 距离最短的边是 $(v_3, v_4)$ , 加入边 $(v_3, v_4)$ 到生成树, 加入顶点 $v_3$ 到 $V'$ , 得到:



这时 $V' = V$ , 算法终止, 我们看到得到的生成树与Kruskal算法的结果一样。

**练习 3.2.5** 根据上面的例子, 理解教材中对Kruskal算法和Prim算法的一般描述, 并思考这两个算法的关键步骤 (或最复杂的步骤, 最费时间的步骤) 各是什么。

### 3.3 关键路径

在许多工程项目管理中, 我们都会遇到与关键路径有关的问题。通常一项工程都会分为很多工序, 这些工序之间有一些约束, 最简单的约束是时间约束, 例如工序 $j$ 必须在工序 $i_1, i_2, \dots, i_k$ 完成之后才能进行等等。

我们以后将在软件工程课程中也会遇到与关键路径有关的问题, 因为按照软件生命周期, 软件开发项目通常分为需求分析、总体设计、详细设计、实现、单元测试、集成测试、验收测试等各个阶段, 而且当软件规模比较大的时候, 又会分成几个子系统, 每个子系统都有这些阶段, 从而个子系统的各阶段之间存在复杂的时间约束关系, 软件项目经理通常需要找出其中的关键步骤, 从而控制软件开发的进度。戴一奇的教材p29的例2.7.1给出了一个建筑方面的例子。

我们先引入一些概念来对这种与关键路径有关问题进行建模。

**定义 3.3.1 PERT图**是一个有向连通网络 $G = (V, E)$ , 且:

- (1) 每条边表示一个作业 (工序), 边的非负实数权表示完成该作业所需的时间;
- (2) 每个顶点表示以该顶点为起点的作业的开始, 也表示以该顶点为终点的作业结束;
- (3) **作业开始条件**: 某条有向边 $e = \langle v, u \rangle$ 所代表的作业可以开始当且仅当以 $v$ 为终点的有向边所代表的作业全部完成;
- (4)  $G$ 没有有向回路, 也没有环;

(5)  $G$ 中有且仅有一个顶点的入度为0, 称该顶点为**源点**; 同时 $G$ 中有且仅有一个顶点的出度为0, 称该顶点为**汇点**。

实际上, 我们有:

**引理 3.3.2** 没有回路的有向连通网络 $G$ 存在入度为0的顶点, 同时也存在出度为0的顶点。

**证明** 考虑 $G$ 的一条极长有向道路 $\Gamma$ , 设其起点为 $v_1$ , 终点为 $v_n$ 。若 $d^-(v_1) \neq 0$ , 则存在边 $(v_i, v_1)$ , 若 $v_i \in \Gamma$ , 则 $G$ 存在有向回路, 若 $v_i \notin \Gamma$ , 则 $\Gamma$ 不是极长道路, 总之必有 $d^-(v_1) = 0$ 。类似地也有 $d^+(v_n) = 0$ 。□

**定理 3.3.3** 设 $G = (V, E)$ 不存在有向回路, 可以将 $G$ 的顶点重新编号为 $v'_1, v'_2, \dots, v'_n$ 使得对任意边 $(v'_i, v'_j) \in E$ , 都有 $i < j$ 。

根据这个定理, 我们总是假定PERT图的顶点编号满足上述定理给出的条件。对于PERT图, 我们主要关心其关键路径, 以及每个作业的最早启动时间和最晚启动时间:

**定义 3.3.4** 给定PERT图 $G = (V, E)$ ,  $G$ 中从源点到汇点最长带权路径称为**关键路径**。关键路径的长度 $T$ 为完成整个任务(即包括所有作业)所必需的最少时间。关键路径上的边所代表的作业称为**关键作业**。

对于某个作业 $e = \langle v_k, - \rangle$ , 源点到该作业起点 $v_k$ 的最长路径长度 $\pi_k$ 称为该作业的**最早启动时间**。设 $v_k$ 到**汇点**的最长路径长度为 $\pi'_k$ , 则该作业可最晚于 $\tau_k = T - \pi'_k$ 时间启动而不影响整个任务完成的预期时间 $T$ ,  $\tau_k$ 称为该作业的**最晚启动时间**。

很显然, 作业 $e = \langle v_k, - \rangle$ 是关键作业当且仅当 $\tau_k = \pi_k$ , 因为这时 $e$ 是处于从源点到汇点最长带权路径上。所以, 我们可通过求每个作业的最早启动时间和最晚启动时间而得到PERT图的关键路径。

**定理 3.3.5** 设PERT图 $G = (V, E)$ 的顶点编号 $v_1, v_2, \dots, v_n$ 使得边 $e = \langle v_i, v_j \rangle \in E$ 蕴涵 $i < j$ 。令:

$$\pi_1 = 0 \quad \pi_j = \max\{\pi_l + w(v_l, v_j) \mid 1 \leq l < j\}$$

则 $\pi_i (1 \leq i \leq n)$ 是作业 $e = \langle v_i, v_j \rangle$ 的最早启动时间。这里 $w(v_l, v_j)$ 定义为:

$$w(v_l, v_j) = \begin{cases} \text{边}\langle v_l, v_j \rangle\text{的非负实数权} & \text{若}\langle v_l, v_j \rangle \in E \\ -\infty & \text{否则} \end{cases}$$

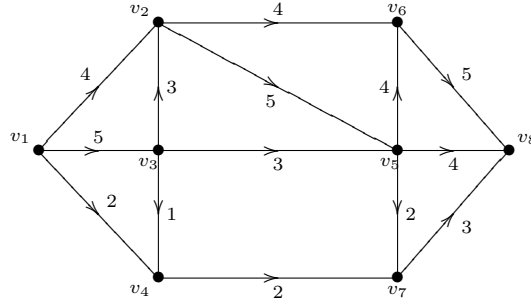
令:

$$\tau_n = \pi_n \quad \tau_k = \min\{\tau_l, \tau_l - w(v_k, v_l) \mid k < l \leq n\}$$

则 $\tau_i (1 \leq i \leq n)$ 是作业 $e = \langle v_i, v_j \rangle$ 的最晚启动时间。

我们无需详细证明上述定理, 下面以一个简单的例子说明如何给出PERT图中每个作业的最早启动时间和最晚启动时间, 从而得到整个PERT图的关键路径。

**例子 3.3.6** 考虑下面的PERT图:



注意顶点可看作一种状态,在该状态,所有以它为终点的作业都已经完成,而所有以它为起点的作业准备启动。为计算每个作业(边)的最早启动时间和最晚启动时间,我们实际上计算每个顶点的最早启动时间和最晚启动时间,也即,某顶点的最早启动时间(最晚启动时间)就是所有以该顶点为起点的作业的最早启动时间(最晚启动时间)。

我们先计算每个顶点的最早启动时间。最早启动时间的计算是从源点开始计算,源点的最早启动时间为0,对于顶点 $v_i$ ,只有**在所有以 $v_i$ 为终点的边 $\langle v_l, v_i \rangle$ 的起点 $v_j$ 的最早启动时间都计算出来之后**,才能计算 $v_i$ 的最早启动时间,而且 $v_i$ 的最早启动时间 $\pi_i$ 是:

$$\pi_i = \max\{\pi_j + w(v_j, v_i) \mid \langle v_j, v_i \rangle \in E\}$$

对于上述PERT图,首先 $\pi_1 = 0$ ,下一步,我们看到,只有 $v_3$ 能够计算最早启动时间,且:

$$\pi_3 = \max\{\pi_1 + w(v_1, v_3)\} = \max\{0 + 5\} = 5$$

再下一步,我们看到现在可以计算 $v_2$ 和 $v_4$ 的最早启动时间:

$$\pi_2 = \max\{\pi_1 + w(v_1, v_2), \pi_3 + w(v_3, v_2)\} = \max\{4, 5 + 3\} = 8$$

$$\pi_4 = \max\{\pi_1 + w(v_1, v_4), \pi_3 + w(v_3, v_4)\} = \max\{2, 5 + 1\} = 6$$

再下一步,我们看到现在可以计算 $v_5$ 的最早启动时间:

$$\pi_5 = \max\{\pi_3 + w(v_3, v_5), \pi_2 + w(v_2, v_5)\} = \max\{5 + 3, 8 + 5\} = 13$$

再下一步,我们看到现在可以计算 $v_6, v_7$ 的最早启动时间:

$$\pi_6 = \max\{\pi_2 + w(v_2, v_6), \pi_5 + w(v_5, v_6)\} = \max\{8 + 4, 13 + 4\} = 17$$

$$\pi_7 = \max\{\pi_4 + w(v_4, v_7), \pi_5 + w(v_5, v_7)\} = \max\{6 + 2, 13 + 2\} = 15$$

最后可计算 $v_8$ 的最早启动时间:

$$\begin{aligned} \pi_8 &= \max\{\pi_5 + w(v_5, v_8), \pi_6 + w(v_6, v_8), \pi_7 + w(v_7, v_8)\} \\ &= \max\{13 + 4, 17 + 5, 15 + 3\} = 22 \end{aligned}$$

这就得到源点 $v_1$ 到汇点 $v_8$ 的最长带权路径长度为22,这也是完成整个任务的必需时间。

现在可以计算每个顶点的最晚启动时间。最晚启动时间的计算是从汇点开始计算,汇点的最晚启动时间等于其最早启动时间,对于以顶点 $v_i$ ,只有**在所有以 $v_i$ 为起点的边 $\langle v_i, v_j \rangle$ 的终点 $v_j$ 的最晚启动时间都计算出来之后**,才能计算 $v_i$ 的最晚启动时间,而且 $v_i$ 的最晚启动时间 $\tau_i$ 是:

$$\tau_i = \min\{\tau_j - w(v_i, v_j) \mid \langle v_i, v_j \rangle \in E\}$$



对于上述PERT图，首先 $\tau_8 = \pi_8 = 22$ ，下一步，我们看到，可以计算 $v_6, v_7$ 的最晚启动时间：

$$\tau_6 = \min\{\tau_8 - w(v_6, v_8)\} = \min\{22 - 5\} = 17$$

$$\tau_7 = \min\{\tau_8 - w(v_7, v_8)\} = \min\{22 - 3\} = 19$$

再下一步，我们看到现在可以计算 $v_5$ 的最晚启动时间：

$$\begin{aligned}\tau_5 &= \min\{\tau_6 - w(v_5, v_6), \tau_7 - w(v_5, v_7), \tau_8 - w(v_5, v_8)\} \\ &= \min\{17 - 4, 19 - 2, 22 - 4\} = 13\end{aligned}$$

再下一步，我们看到可以计算 $v_2, v_4$ 的最晚启动时间：

$$\tau_2 = \min\{\tau_6 - w(v_2, v_6), \tau_5 - w(v_2, v_5)\} = \min\{17 - 4, 13 - 5\} = 8$$

$$\tau_4 = \min\{\tau_7 - w(v_4, v_7)\} = \min\{19 - 2\} = 17$$

再下一步，我们看到可以计算 $v_3$ 的最晚启动时间：

$$\begin{aligned}\tau_3 &= \min\{\tau_2 - w(v_3, v_2), \tau_4 - w(v_3, v_4), \tau_5 - w(v_3, v_5)\} \\ &= \min\{8 - 3, 17 - 1, 13 - 3\} = 5\end{aligned}$$

最后可以计算源点 $v_1$ 的最晚启动时间（实际上，源点的最晚启动时间应该是0）：

$$\begin{aligned}\tau_1 &= \min\{\tau_2 - w(v_1, v_2), \tau_3 - w(v_1, v_3), \tau_4 - w(v_1, v_4)\} \\ &= \min\{8 - 4, 5 - 5, 17 - 2\} = 0\end{aligned}$$

这样我们得到了各个顶点的最早启动时间和最晚启动时间：

顶点 $V$	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$	$v_7$	$v_8$
最早启动时间 $\pi$	0	8	5	6	17	13	15	22
最晚启动时间 $\tau$	0	8	5	17	17	13	19	22
缓冲时间 $\tau$	0	0	0	11	0	0	4	0

根据各个顶点的最早启动时间和最晚启动时间可以确定PERT图的关键路径，即**凡是最早启动时间等于最晚启动时间的顶点就是处于关键路径的顶点**。对于上述PERT图，我们得到的关键路径就是：

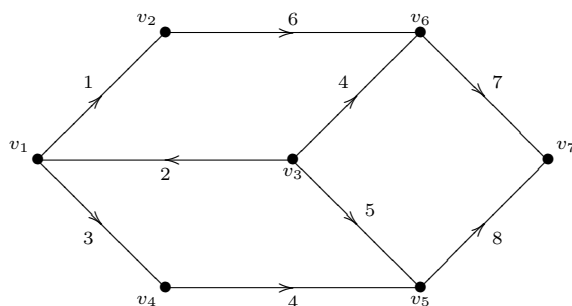
$$\Gamma = v_1 v_3 v_2 v_6 v_5 v_8$$

处于关键路径上的作业（边）就是关键作业，这些作业的完成时间必须按照预期的时间完成，否则就会影响整个任务的完成时间，而处于非关键路径上的作业 $\langle v_i, v_j \rangle$ 就允许有**延误时间**，其可以延误的时间等于 $\tau_j - \pi_i - w(v_i, v_j)$ ，例如，作业 $\langle v_1, v_4 \rangle$ 的允许延误时间就是 $\tau_4 - \pi_1 - w(v_1, v_4) = 15$ ，也就是说，该作业即使比原来预计的2个时间再晚15个时间也能保证整个任务在22个时间内完成（当然前提是其他作业，特别是作业 $\langle v_4, v_7 \rangle$ 不再延迟）。而作业 $\langle v_4, v_7 \rangle$ 的允许时间是 $\tau_7 - \pi_4 - w(v_4, v_7) = 11$ 。有时，我们简单地讨论顶点的**缓冲时间**，顶点 $v_i$ 的缓冲时间等于 $\tau_i - \pi_i$ ，各顶点的缓冲时间如上表所示。

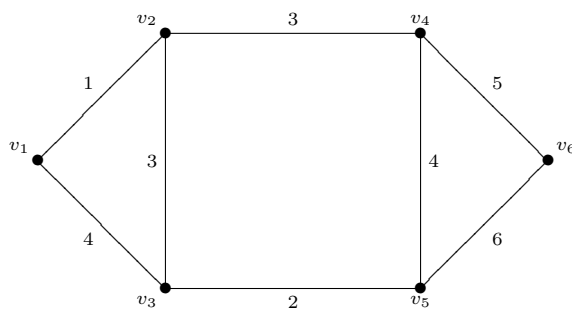
戴一奇教材介绍了PT和PERT两种图，这两种图的主要差别在于PT图以顶点代表工序，以该顶点为起点的边的权都是该工序的完成时间，而PERT图以边代表工序，边的权就是该工序的完成时间。PERT图通常比PT图更为简单，但PT图可描述对工序的更多约束。这里只讨论PERT图的关键路径计算，对于PT图，其关键路径的计算类似。

## 作 业

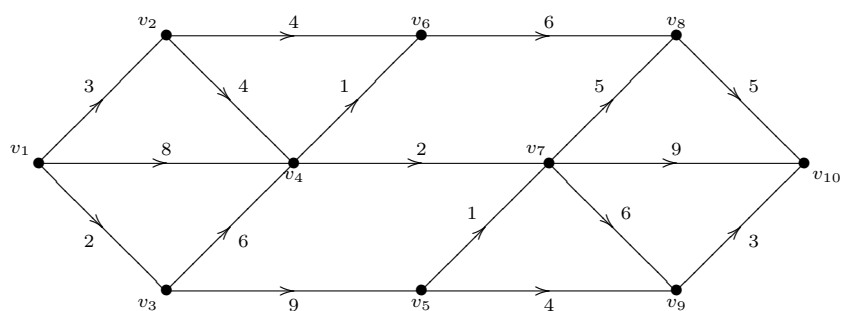
**作业 3.1** 写出下面的带权有向图的距离矩阵, 并使用Dijkstra算法求 $v_1$ 到各点的最短举例, 然后用Warshall算法求任意两点之间的最短距离:



**作业 3.2** 分别使用Kruskal算法和Prim算法求下图的最小生成树:



**作业 3.3** 对于下面的PERT图, 计算每个顶点的最早启动时间和最晚启动时间, 并给出其关键路径:



## 第四章 平面图与着色

在实际问题中有时要涉及到图的平面性的讨论，比如印刷电路板的设计、大规模集成电路的布局布线等，都离不开图的平面性研究。著名的四色猜想也属于平面性范畴。这一章讨论平面图的基本性质，以及与平面图相关的着色问题。

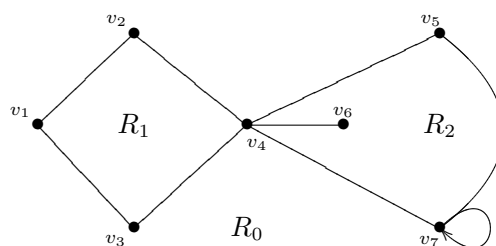
### 4.1 平面图及其性质

**定义 4.1.1** 给定图  $G = (V, E)$ ，若能将其画在平面上，且任意两条边的交点只能是  $G$  的顶点，则称  $G$  **可嵌入平面**，或称  $G$  **是可平面的**。可平面图在平面上的一个嵌入称为一个**平面图**。

实际上，树就是一类重要的平面图，因为树没有回路，也就是说，树没有一个封闭的区域，因此总能通过适当排列其顶点，使得任意两条边之间都不相交。

**定义 4.1.2** 设  $G$  是平面图，由它的若干边包围而成，且其中不含图的顶点和边的区域称为  $G$  的一个**面或域**。包围域  $R$  的所有边组成的回路称为该域的**边界**，回路长度称为该域的**度**，记为  $\deg(R)$ 。由平面图的边包围且无穷大的域称为平面图的外部面，其他的面都称为**内部面**。

**例子 4.1.3** 考虑下面的平面图：



这个平面图共有四个面，其中  $R_0$  是外部面， $R_3$  由  $v_7$  上的环围成，上面没有标出，这些面的边界和度分别是：

$R_0$ 的边界：	$v_1v_2v_4v_5v_7v_7v_4v_3v_1$	$\deg(R_0) = 8$
$R_1$ 的边界：	$v_1v_2v_4v_3v_1$	$\deg(R_1) = 4$
$R_2$ 的边界：	$v_4v_5v_7v_4v_6v_4$	$\deg(R_2) = 5$
$R_3$ 的边界：	$v_7v_7$	$\deg(R_3) = 1$

注意  $v_4$  和  $v_6$  之间的边作为  $R_2$  的边界需要计算两次。

类似于握手定理, 由于平面图的任意边都作为两个面的边界, 因此我们有:

**定理 4.1.4** 平面图所有面的度之和等于边数的两倍。

在平面图研究中, 最重要的是**欧拉公式**:

**定理 4.1.5** 设平面连通图 $G$ 有 $n$ 个顶点,  $m$ 条边,  $d$ 个域, 则有 $n - m + d = 2$ 。

**证明**  $G$ 是连通图, 有生成树 $T$ , 它包含 $n - 1$ 条边, 不产生回路, 因此对 $T$ 而言只有一个外部面。由于 $G$ 是平面图, 每加入一条余树边, 它一定不与其他边相交, 也就是说一定是跨在某个面的内部, 把该面分成两个面, 因此加入 $m - n + 1$ 条余树边就得到 $m - n + 2$ 个面。□

**推论 4.1.6** 设平面图 $G$ 的连通分支数为 $k$ , 并有 $n$ 个顶点,  $m$ 条边,  $d$ 个域, 则有 $n - m + d = k + 1$ 。

下面考虑极大平面图:

**定义 4.1.7** 设 $G = (V, E)$ 为简单平面图,  $|V| > 3$ , 若对任意 $v_i, v_j \in V$ 且 $(v_i, v_j) \notin E$ , 有 $G' = (V, E \cup (v_i, v_j))$ 为非平面图, 即在 $G$ 的任意两个不相邻顶点之间加一条新边都得到非平面图, 则称 $G$ 为一个**极大平面图**。

根据这个定义, 我们看到, 这里的“极大性”是针对固定顶点数的图的边的数目而言。很容易证明极大平面图的如下性质:

- 引理 4.1.8**
- (1) 极大平面图是连通图。
  - (2) 极大平面图没有桥。
  - (3) 极大平面图的每个面都由3条边组成。
  - (4) 极大平面图有 $3d = 2m$  ( $d$ 为面数目,  $m$ 为边数目)。
  - (5) 对于极大平面图 $G = (V, E)$ , 若 $|V| > 4$ , 则对任意的顶点 $v_i \in V$ , 有 $v_i$ 的度数大于等于3。

**证明**

(1) 若极大平面图不是连通图, 则在分别取两个连通分支上的一个顶点, 将这两个顶点相连不会增加面数, 也就是说不会改变图的平面性, 从而与极大性相矛盾。

(2) 若 $e = (v, u)$ 是极大平面图的一个桥(割边), 则 $v$ 的一个相邻的顶点 $v'$ 与 $u$ 不相邻, 但显然在 $v'$ 与 $u$ 之间增加一条边不会改变图的平面性, 从而与极大性矛盾!

(3) 极大平面图的每个面都由3条边组成, 否则可以在面内增加边而不改变平面性, 再由面的度数等于边数两倍, 就有 $3d = 2m$ 。

(4) 由极大平面图的每个面都由3条边组成, 及面的度数之和等于边数两倍可得。

(5) 留作练习。

□

进一步利用欧拉公式, 我们有:

**定理 4.1.9** 设极大平面图 $G$ 有 $n$ 个顶点,  $m$ 条边,  $d$ 个面, 则有 $m = 3n - 6$ 且 $d = 2n - 4$ 。

**证明** 将 $2m = 3d$ 代入 $n - m + d = 2$ 得 $n - \frac{3}{2}d + d = 2$ , 即 $d = 2n - 4$ , 从而 $m = 3n - 6$  □

**推论 4.1.10** 设简单平面图 $G$ 有 $n$ 个顶点,  $m$ 条边,  $d$ 个面, 则有 $m \leq 3n - 6$ 且 $d \leq 2n - 4$ 。

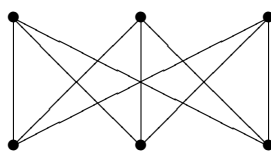
**定理 4.1.11** 简单平面图 $G$ 中存在度小于6的顶点。

**证明** 如果 $G$ 的任意度都大于等于6, 即 $G$ 的总度数大于等于 $6n$ , 由边数等于顶点度数两倍得, 边数大于等于 $3n$ , 与 $m \leq 3n - 6$ 矛盾!  $\square$

库拉图斯基定理是在理论上判别一个图是否是平面图的重要定理。为介绍该定理, 我们先定义二部图的概念:

**定义 4.1.12** 给定图 $G = (V, E)$ , 如果 $V$ 能分为两个不相交的非空子集 $V_1, V_2$ , 即 $V_1 \neq \emptyset, V_2 \neq \emptyset, V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$ , 且对 $G$ 的任意边 $e = (u, v) \in E$ , 都有 $u \in V_1$ 且 $v \in V_2$ , 或者 $v \in V_1$ 且 $u \in V_2$ , 也即 $G$ 的任意边的两个端点都在不同的子集, 则称 $G$ 是**二部图**。若 $|V_1| = s, |V_2| = r$ , 且 $V_1$ 的任意顶点都与 $V_2$ 的任意顶点有边, 则称该二部图为**完全二部图**, 记为 $K_{s,r}$ 。

对于平面图而言, 最重要的是 $K_{3,3}$ :



**定理 4.1.13** 完全图 $K_5$ 和完全二部图 $K_{3,3}$ 都不是平面图。

**定义 4.1.14**  $K_5$ 和 $K_{3,3}$ 分别记为 $K^{(1)}$ 和 $K^{(2)}$ 图, 在 $K^{(1)}$ 和 $K^{(2)}$ 图上任意增加一些度为2的顶点之后得到的图称为 $K^{(1)}$ 和 $K^{(2)}$ 型图, 统称为 **$K$ 型图**。

**定理 4.1.15 库拉图斯基定理:** 图 $G$ 是可平面的当且仅当 $G$ 不存在 $K$ 型子图。

这个定理只是在理论上给出了一个图是可平面的充分必要条件, 但实际上, 很难应用该定理判断一个图是否是可平面的, 目前判断一个图是否是可平面的算法比较复杂, 这里不再详细讨论。最后我们讨论平面图的对偶图的概念。

**定义 4.1.16** 给定图 $G = (V, E)$ , 满足下列条件的图 $G^* = (V^*, E^*)$ 称为图 $G$ 的**对偶图**:  $G$ 的任一域 $f_i$ 内有且仅有一点 $v_i^*$ ; 对 $G$ 的域 $f_i, f_j$ 的共同边界 $e_k$ , 画一条 $e_k^* = (v_i^*, v_j^*)$ 且只与 $e_k$ 交于一点; 若 $e_k$ 完全处于 $f_i$ 中, 则 $v_i^*$ 有一自环 $e_k^*$ 且与 $e_k$ 相交于一点。上述过程称为求对偶图的 **$D$ 过程**, 得到的对偶图称为原图的拓扑对偶。

对平面图 $G$ ,  $D$ 过程构造的 $G^*$ 是唯一的; 对于非平面图,  $D$ 过程可能不成立; 对平面图 $G$ ,  $D$ 过程构造的 $G^*$ 也是平面图; 不论图 $G$ 是否连通,  $D$ 过程得到的 $G^*$ 是连通的; 若图 $G$ 连通, 且存在 $G^*$ , 则 $(G^*)^* = G$ ; 对图 $G$ , 若存在 $G^*$ , 则 $G$ 中回路对应于 $G^*$ 中割集,  $G$ 中割集对应于 $G^*$ 中回路; 若 $G \simeq G^*$ , 称 $G$ 为**自对偶图**。

**定理 4.1.17** 图 $G$ 有对偶图当且仅当 $G$ 是平面图。

**定理 4.1.18** 对图 $G$ 施行 $D$ 过程得到 $G^*$ , 设 $n, m, d$ 和 $n^*, m^*, d^*$ 分别表示 $G$ 和 $G^*$ 的结点数、边数及域数, 则有:

$$m^* = m \quad n^* = d \quad \deg(v_i^*) = \deg(f_i)$$

**定理 4.1.19** 设 $G$ 为平面图, 施行 $D$ 过程得到 $G^*$ , 设 $n, m, d$ 和 $n^*, m^*, d^*$ 分别表示 $G$ 和 $G^*$ 的结点数、边数及域数,  $k$ 为 $G$ 的连通分支数, 则有:  $d^* = n - k + 1$

## 4.2 图的着色

图的着色包括顶点着色、边着色以及平面图的面着色, 这里只讨论简单无向图的顶点着色。图的着色可用于解决一些实际问题。

**例子 4.2.1** 某系二年级学生共选修全校性的选修课程 $n$ 门, 期末考试前要将这 $n$ 门课程先考完, 要求每天每个学生至多只能参加一门课程的考试, 问至少需要几天才能使每个学生将所选的课程都考完?

为解决这个问题, 我们可以使用图来建立一个模型。设 $V = \{c_1, c_2, \dots, c_n\}$ 为课程集合, 设 $S(c_i)$ 为学习课程 $c_i$ 的学生集合, 若 $S(c_i) \cap S(c_j) \neq \emptyset, i \neq j$ , 我们就在 $c_i$ 和 $c_j$ 之间连一条边, 这样得到一个以 $V$ 为顶点集的无向图, 对这个图的顶点进行着色, 使得相邻顶点着不同颜色, 那么需要的最小颜色数就是使每个学生都考完其选修课程所需要的天数。

我们先讨论图的顶点着色的一些基本性质, 然后再以一个更具体的例子说明这一类问题的求解方法。戴一奇教材p83页例4.6.1则给出了一个货物存储的问题, 此问题与考试安排问题类似, 也可用图的着色来解决。

**定义 4.2.2** 图 $G = (V, E)$ 的一个 **$k$ 顶点着色**指用 $k$ 种颜色对 $G$ 的各顶点的一种分配方案。若**着色使得相邻顶点的颜色都不同**, 则称该着色正常, 或称 $G$ 存在一个正常的 $k$ 顶点着色 (或称一个 **$k$ 着色**)。此时称 $G$ 为 **$k$ -可着色的**。

**定义 4.2.3** 给定图 $G = (V, E)$ , 使该图 **$k$ -可着色**的最小 $k$ 值称为 $G$ 的**色数**, 记为 $\gamma(G)$ 。若 $\gamma(G) = k$ , 称 $G$ 为 **$k$ 色图**。

显然我们可得到一些图的色数:

- 例子 4.2.4**
- (1) 若图 $G$ 是零图 (即只有一个顶点没有边的图), 则 $\gamma(G) = 1$ ;
  - (2) 若图 $G$ 是 $n$ 个顶点的完全图, 因为任意一个顶点都与 $n - 1$ 个顶点相邻, 因此 $\gamma(G) = n$ ;
  - (3) 若图 $G$ 是二部图, 则由于任意边都是连接两个不同集合中的顶点, 因此 $\gamma(G) = 2$ ;
  - (4) 若图 $G$ 是 $2n$ 个顶点的回路, 则 $\gamma(G) = 2$ ;
  - (5) 若图 $G$ 是 $2n + 1$ 个顶点的回路, 则 $\gamma(G) = 3$ ;

**引理 4.2.5** 若图 $T$ 是 $n(\geq 2)$ 个顶点的树, 则 $\gamma(T) = 2$ 。

**证明** 任取树 $T$ 的一个顶点 $v_0$ , 定义:

$$V_1 = \{u \mid v_0 \text{ 到 } u \text{ 的惟一路径长度为奇数}\} \quad V_2 = \{v \mid v_0 \text{ 到 } v \text{ 的惟一路径长度为偶数}\}$$

那么显然 $V_1$ 之中的任意两个顶点都不相邻, 因为若 $v_i, v_j \in V_1$ ,  $v_0$ 到 $v_i$ 的惟一路径是 $\Gamma_i$ ,  $v_0$ 到 $v_j$ 的惟一路径是 $\Gamma_j$ , 显然 $\Gamma_i \neq \Gamma_j$ , 若还有边 $e = (v_i, v_j) \in E$ , 则 $\Gamma_i, \Gamma_j$ 以及边 $e$ 构成回路, 与 $T$ 是树矛盾, 类似地 $V_2$ 之中的任意两个顶点也都不相邻, 从而 $T$ 是二部图, 即 $\gamma(G) = 2$ 。□

**定理 4.2.6** 给定  $G = (V, E)$  是非空图 (即  $|V| > 1, |E| > 0$ )，则  $\gamma(G) = 2$  当且仅当它没有奇数长的回路。

**证明** 如果  $G$  有奇数长的回路，则显然  $\gamma(G) \geq 3$ ，因此  $\gamma(G) = 2$  意味着图  $G$  没有奇数长的回路；反之，若它没有奇数长回路，若  $G$  有多个连通分支，则不同连通分支之间的着色不会互相影响，因此不妨假设它只有一个连通分支。

任取树  $G$  的一个顶点  $v_0$ ，定义：

$$V_1 = \{u \mid v_0 \text{ 到 } u \text{ 的最短路径长度为奇数}\} \quad V_2 = \{v \mid v_0 \text{ 到 } v \text{ 的最短路径长度为偶数}\}$$

那么显然  $V_1$  之中的任意两个顶点都不相邻，因为若  $v_i, v_j \in V$ ， $v_0$  到  $v_i$  的最短路径是  $\Gamma_i$ ， $v_0$  到  $v_j$  的最短路径是  $\Gamma_j$ ，显然  $\Gamma_i \neq \Gamma_j$ ，若还有边  $e = (v_i, v_j) \in E$ ，则  $\Gamma_i, \Gamma_j$  以及边  $e$  构成奇数长的回路，矛盾！类似地  $V_2$  之中的任意两个顶点也都不相邻，从而  $G$  是二部图，即  $\gamma(G) = 2$ 。□

下面的定理给出了一个图的色数的上界：

**定理 4.2.7** 对任意图  $G = (V, E)$ ， $\gamma(G) \leq \Delta(G) + 1$ ， $\Delta(G)$  是  $G$  中度数最大的顶点的度数。

**证明** 对  $G$  的顶点数作归纳。 $n = 1$  时命题显然成立。假设  $n = k - 1$  时命题成立，即对任意具有  $k - 1$  个顶点的图  $G$ ， $\gamma(G) \leq \Delta(G) + 1$ 。对于具有  $k$  个顶点的图  $G$ ，任取一个顶点  $v$ ，图  $G - v$  具有  $k - 1$  个顶点，根据归纳假设有  $\gamma(G - v) \leq \Delta(G - v) + 1$ 。由于  $\Delta(G - v) \leq \Delta(G)$ ，因此  $\gamma(G - v) \leq \Delta(G) + 1$ ，即  $\Delta(G) + 1$  种颜色可对  $G - v$  着色。放回  $v$ ，由于  $v$  的度数小于  $\Delta(G)$ ，与  $v$  相邻的顶点之多占用  $\Delta(G)$  种颜色，因此用  $\Delta(G) + 1$  种颜色也可对  $G$  着色。□

对于一个具体的图  $G$ ，如何确定它的色数呢？下面介绍色数的一种求解方法。先引入一个定义：

**定义 4.2.8** 给定图  $G = (V, E)$ ，设  $i, j$  是图  $G$  的两个不相邻的顶点。定义  $\overline{G}_{ij} = G + e_{ij}$ ，这里  $e_{ij} = (i, j)$ ，即  $\overline{G}_{ij}$  是  $G$  在顶点  $i$  和  $j$  之间增加一条新边而得到的图。

定义  $\overset{\circ}{G}_{ij}$  也是一个简单图，其顶点集  $\overset{\circ}{V}_{ij} = V - \{i, j\} + \{ij\}$ ，边集：

$$\overset{\circ}{E}_{ij} = E - \{(k, i) \mid (k, i) \in E\} - \{(k, j) \mid (k, j) \in E\} + \{(k, ij) \mid (k, i) \in E \vee (k, j) \in E\}$$

直观地说， $\overset{\circ}{G}_{ij}$  是将  $G$  中顶点  $i, j$  收缩成一个顶点  $ij$ ，原先与  $i$  或  $j$  相邻的顶点都与顶点  $ij$  相连，并合并多重边而得到的简单图。

**定理 4.2.9** 设  $i, j$  是简单图  $G$  的两个不相邻的顶点，则：

$$\gamma(G) = \min\{\gamma(\overline{G}_{ij}), \gamma(\overset{\circ}{G}_{ij})\}$$

**证明** 对  $G$  中顶点的任何着色，顶点  $i$  和  $j$  要么着以同色，要么着以异色，二者必居其一。设  $i, j$  着相同颜色时  $G$  的最少着色数是  $\gamma(G(i, j \text{ 同色}))$ ， $i, j$  着不同颜色时  $G$  的最少着色数是  $\gamma(G(i, j \text{ 异色}))$ ，则显然：

$$\gamma(G) = \min\{\gamma(G(i, j \text{ 同色})), \gamma(G(i, j \text{ 异色}))\}$$

而显然又有：

$$\gamma(G(i, j \text{ 同色})) = \gamma(\overset{\circ}{G}_{ij}) \quad \gamma(G(i, j \text{ 异色})) = \gamma(\overline{G}_{ij})$$

□



根据上述定理, 我们通过不断在 $G$ 的不相邻顶点之间增加边, 以及收缩不相邻顶点而将图 $G$ 换成完全图, 而得到任意图 $G$ 的色数。例子可见戴一奇教材p86的例4.6.4。

给定一个图, 最多使用 $k$ 种颜色对其进行着色, 使得其相邻顶点着不同颜色, 那么会有多少种不同的着色方案呢? 显然方案数与 $k$ 有关, 可以证明方案数是 $k$ 的多项式, 因此称为色多项式:

**定义 4.2.10** 图 $G$ 使用 $k$ 种颜色进行正常着色的方案数称为图 $G$ 的**色多项式**, 记为 $f(G, k)$ 。

**例子 4.2.11** 不难得到一些特殊图的色多项式:

- (1) 对于零图:  $G = (V, E)$ ,  $n = |V|$ ,  $|E| = 0$ , 其色多项式显然是:  $f(G, k) = k^n$ ;
- (2) 对于树 $T$ :  $T = (V, E)$ ,  $n = |V|$ , 根节点可在 $k$ 种颜色中任取, 非根节点选取与其父亲节点不同的颜色, 也就是说, 剩下的 $n - 1$ 个顶点都有 $k - 1$ 种选择, 因此 $f(G, k) = k(k - 1)^{n-1}$ ;
- (3) 对于完全图 $K_n$ :  $f(K_n, k) = k(k - 1)(k - 2) \cdots (k - n + 1)$ 。

类似求图 $G$ 的色数的方法, 我们有如下定理:

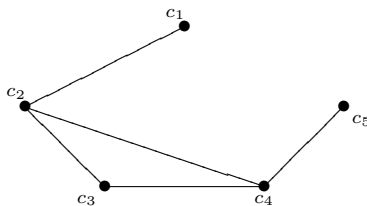
**定理 4.2.12** 给定图 $G$ ,  $i, j$ 是其两个不相邻的顶点, 则:

$$f(G, k) = f(\overline{G}_{ij}, k) + f(\overset{\circ}{G}_{ij}, k)$$

利用这个公式, 以及完全图的色多项式, 可以求任意图的色多项式, 例子可参见戴一奇教材p87的例4.6.5。这里我们回到这一节一开始提出的考试安排问题:

**例子 4.2.13** 假定有5门课程 $c_1, c_2, c_3, c_4, c_5$ , 已知有的学生既选 $c_1$ 又选 $c_2$ , 有的学生既选 $c_2$ 又选 $c_3$ , 有的既选 $c_3$ 又选 $c_4$ , 有的既选 $c_4$ 又选 $c_2$ , 也有的既选 $c_4$ 又选 $c_5$ , 问至少要安排多少天考试, 在安排最少天数的情况下至多有多少种安排方案?

**解:** 我们以 $c_1, c_2, c_3, c_4, c_5$ 为顶点, 若有学生既选课程 $c_i$ 又选 $c_j$ 则在 $c_i$ 和 $c_j$ 之间连一条边, 这样根据题目的陈述, 我们得到如下无向简单图:



很容易知道上图的色数 $\gamma(G) = 3$ , 而色多项式是:

$$f(G, k) = k^5 - 5k^4 + 9k^3 - 7k^2 + 2k$$

从而 $f(G, 3) = 24$ , 也就是说至少要安排3天考试, 至多有24种安排方案。

**作业:** 戴一奇教材p87至p88的习题四: 1、2、13



## 第五章 支配集、覆盖集、独立集与匹配

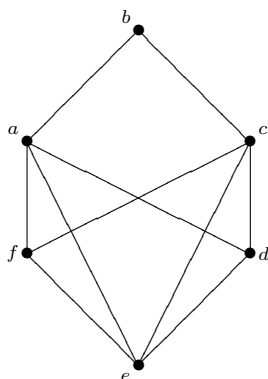
这一章讨论支配集、覆盖集、独立集以及匹配，这些问题的共同点是在图中找出满足一定性质顶点集或边集。目前还没有有效的算法给出任意图的支配集、覆盖集和独立集等，相关问题的算法及其复杂度研究为理论计算机，特别是可计算性理论提供了诸多素材。匹配问题则有着丰富的应用背景，也有比较有效的算法，因此这一章将重点讨论与匹配问题有关的算法。

这一章讨论的图都是无向简单图，如果没有特别说明，下面定义和定理中所说的图都是指无向简单图。这一章的大部分内容来自耿素云、屈婉玲编著的教材：《离散数学》（修订版），高等教育出版社，2004年。

### 5.1 支配集、点独立集和点覆盖集

**定义 5.1.1** 给定图  $G = (V, E)$ ,  $V^* \subseteq V$ , 若对于任意的  $v_i \in V - V^*$ , 都存在  $v_j \in V^*$  使得  $(v_i, v_j) \in E$ , 则称  $V^*$  是  $G$  的**支配集**。若  $V^*$  是  $G$  的支配集, 且  $V^*$  的任意真子集都不是  $G$  的支配集, 则称  $V^*$  是  $G$  的**极小支配集**。顶点数最少的支配集称为**最小支配集**, 最小支配集的顶点个数称为图  $G$  的**支配数**, 记为  $\gamma_0(G)$ 。

**例子 5.1.2** 考虑下图：



上图的支配集有：

$$\{a, c\}, \{b, e\}, \{c, f\}, \{b, d, f\}, \{a, b, c\}, \{a, b, c, d, e, f\}, \dots$$

极小支配集有：

$$\{a, c\}, \{b, e\}, \{c, f\}, \{b, d, f\}$$

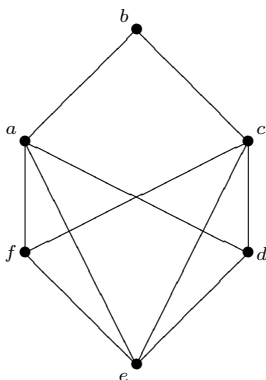
而最小支配集有:

$$\{a, c\}, \{b, e\}, \{c, f\}, \{b, d, f\}$$

注意, 最小支配集不是惟一的, 图 $G$ 的支配数 $\gamma_0(G) = 2$ 。

**定义 5.1.3** 给定 $G = (V, E)$ ,  $V^* \subseteq V$ , 若 $V^*$ 中任意两个顶点都不相邻, 则称 $V^*$ 为 $G$ 的(点)独立集。若 $V^*$ 再加入任何顶点都不再是独立集, 则称 $V^*$ 为极大点独立集, 顶点数最多的点独立集称为最大点独立集, 其顶点数称为图 $G$ 的独立数, 记为 $\beta_0(G)$ 。

**例子 5.1.4** 考虑例子5.1.2中的图:



其独立集有:

$$\{b, d\}, \{b, f\}, \{a, c\}, \{b, d, f\}, \dots$$

其极大独立集有:

$$\{b, d\}, \{a, c\}, \{b, d, f\}$$

而 $\{b, d, f\}$ 是最大独立集, 因此图 $G$ 的独立数是 $\beta_0(G) = 3$ 。

独立集与支配集有如下联系:

**定理 5.1.5** 给定图 $G = (V, E)$ 无孤立顶点, 则 $G$ 的极大点独立集都是 $G$ 的极小支配集。

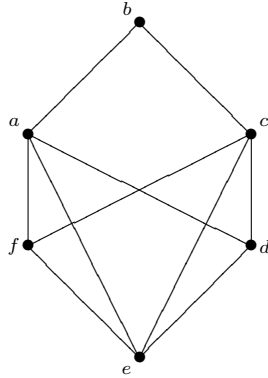
**证明** 设 $V^*$ 是 $G$ 的一个极大点独立集, 我们证明它也是 $G$ 的极小支配集。首先证明它是 $G$ 的支配集, 用反证法。若 $V^*$ 不是 $G$ 的支配集, 即存在顶点 $u \in V - V^*$ , 使得不存在顶点 $v \in V^*$ , 使得有边 $(u, v) \in E$ , 也即 $u$ 不与 $V^*$ 的任何顶点相邻, 从而 $V^* \cup \{u\}$ 也还是独立集, 与 $V^*$ 是极大独立集矛盾, 因此 $V^*$ 必是 $G$ 的支配集。

其次证明 $V^*$ 是极小支配集, 这显然成立, 因为对 $V^*$ 的任意真子集 $V_1^* \subset V^*$ , 在 $V^* - V_1^*$ 中的顶点都不与 $V_1^*$ 中的顶点相邻, 因此 $V_1^*$ 不可能是支配集。□

上述定理的逆不成立, 对于例子5.1.2中的图 $G$ ,  $\{c, f\}$ 是极小支配集, 但它不是点独立集, 更不是极大点独立集。

**定义 5.1.6** 给定图 $G = (V, E)$ ,  $V^* \subseteq V$ , 若对于任意的边 $e \in E$ , 都存在顶点 $v \in V^*$ , 使得 $v$ 与 $e$ 相关联, 则称 $V^*$ 为 $G$ 的点覆盖集, 或简称点覆盖。若 $V^*$ 是 $G$ 的点覆盖, 而 $V^*$ 的任何真子集都不是 $G$ 的点覆盖, 则称 $V^*$ 为极小点覆盖, 顶点个数最少的点覆盖称为最小点覆盖, 其顶点个数称为图 $G$ 的点覆盖数, 记为 $\alpha_0(G)$ 。

例子 5.1.7 考虑例子5.1.2中的图：



其点覆盖有：

$$\{a, b, c, d, e\}, \{a, b, c, e\}, \{a, c, e\}, \dots$$

极小点覆盖有：

$$\{a, c, e\}, \{b, d, e, f\}, \{a, c, d, f\}$$

而 $\{a, c, e\}$ 是图 $G$ 的最小点覆盖，因此图 $G$ 的点覆盖数 $\alpha_0(G) = 3$ 。

点覆盖集和点独立集有如下联系：

**定理 5.1.8** 给定图 $G = (V, E)$ 无孤立点， $V^* \subset V$ ，则 $V^*$ 是 $G$ 的点覆盖当且仅当 $\overline{V^*} = V - V^*$ 是 $G$ 的独立集。

**证明** 先证 $V^*$ 是 $G$ 的点覆盖蕴涵 $\overline{V^*}$ 是 $G$ 的独立集，用反证法。若 $\overline{V^*}$ 不是 $G$ 的独立集，即存在两个顶点 $u, v \in \overline{V^*}$ ， $u, v$ 相邻，这表明边 $e = (u, v)$ 的两个端点都不在 $V^*$ ，这与 $V^*$ 是点覆盖集矛盾！

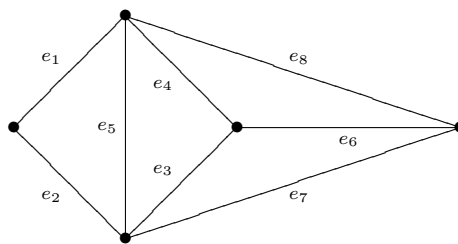
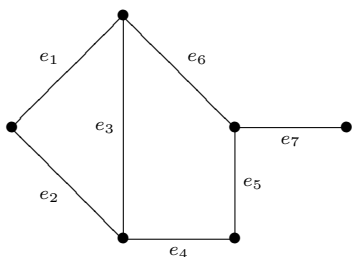
再证 $\overline{V^*}$ 是 $G$ 的独立集蕴涵 $V^*$ 是点覆盖集。由 $\overline{V^*}$ 是 $G$ 的独立集，表明 $G$ 的任意边的两个端点都至少有一个属于 $V^*$ （否则这两个端点在 $\overline{V^*}$ 中相邻），也即 $V^*$ 是点覆盖集。□

**推论 5.1.9** 给定 $G = (V, E)$ 无孤立点， $|V| = n$ ，则 $V^*$ 是 $G$ 的极小点覆盖当且仅当 $\overline{V^*} = V - V^*$ 是 $G$ 的极大独立集，从而有 $\alpha_0 + \beta_0 = n$ 。

## 5.2 边覆盖与匹配

**定义 5.2.1** 给定图 $G = (V, E)$ ， $E^* \subseteq E$ ，若对 $G$ 的任意顶点 $v \in V$ ，都存在边 $e \in E^*$ 与 $v$ 关联，则称 $E^*$ 为图 $G$ 的**边覆盖**。设 $E^*$ 为边覆盖，若 $E^*$ 的任何真子集都不是边覆盖，则称 $E^*$ 为**极小边覆盖**。边数最小的边覆盖称为图 $G$ 的**边覆盖数**，记为 $\alpha_1(G)$ 。

例子 5.2.2 考虑下面两个图：



对于左边的图， $\{e_1, e_4, e_7\}$ ,  $\{e_2, e_5, e_6, e_7\}$ 等都是极小边覆盖，其中 $\{e_1, e_4, e_7\}$ 是最小边覆盖，该图的边覆盖数是3。对于右边的图， $\{e_1, e_3, e_6\}$ ,  $\{e_2, e_4, e_8\}$ 等都是极小边覆盖，而且也都是最小边覆盖，其边覆盖数也是3。

**定义 5.2.3** 给定图 $G = (V, E)$ ,  $E^* \subseteq E$ , 若 $E^*$ 中的任意两条边都不相邻，即任意两条边都不存在公共端点，则称 $E^*$ 为 $G$ 的**边独立集**，也称 $E^*$ 为 $G$ 的**匹配**。若在 $E^*$ 中再添加任意边后都不再是匹配，则称 $E^*$ 为**极大匹配**。边数最多的匹配称为**最大匹配**，其边数称为**边独立数**或**匹配数**，记为 $\beta_1(G)$ 。

**例子 5.2.4** 考虑例子5.2.2中的两个图。对于左边的图， $\{e_2, e_6\}$ ,  $\{e_3, e_5\}$ ,  $\{e_1, e_4, e_7\}$ 等都是极大匹配，其中 $\{e_1, e_4, e_7\}$ 是最大匹配，其匹配数是3。对于右边的图， $\{e_1, e_3\}$ ,  $\{e_2, e_4\}$ ,  $\{e_4, e_7\}$ 等都是极大匹配，同时也都是最大匹配，其匹配数是2。

匹配问题有很强的应用背景，例如在国际会议中，需要两人一组编成讨论小组进行讨论，有的与会人员会一门外语，有的与会人员会很多门外语，假设要求小组中的两人要会一门共同的外语。我们以人为顶点，以能够编成一组的两个人之间连一条边，则编组方案就是求图的匹配，能将最多人进行编组的方案就是最大匹配。

匹配问题中应用得最多的是二部图的匹配，因为常见的任务分配问题都与二部图的匹配问题有关。例如， $m$ 项工作准备分配给 $n$ 个人做，以工作为一个顶点集，以人员为另一个顶点集，两个顶点之间有边表明某项工作可以由某人完成。如果要求每个人至多从事一项工作，一项工作也只能由一个人承担，则工作安排方案就是求二部图的匹配。

为了更好地研究匹配，下面引入更多有关匹配的概念。

**定义 5.2.5** 设 $M$ 是图 $G = (V, E)$ 的一个匹配。

- (1) 若 $e = (u, v) \in M$ ，则称 $u$ 和 $v$ 被 $M$ 所匹配；
- (2) 对于顶点 $v$ ，若存在边 $e \in M$ 使得 $e$ 与 $v$ 关联，则称 $v$ 为 **$M$ 饱和点**，否则称 $v$ 为 **$M$ 非饱和点**；
- (3) 若 $G$ 的每个顶点都是 $M$ 饱和点，则称 $M$ 为**完美匹配**；

对于例子5.2.2的两个图，左边图的完美匹配是 $\{e_1, e_4, e_7\}$ ，右边图不存在完美匹配，对于它的任意一个最大匹配，例如 $M = \{e_2, e_4\}$ ，通过增加关联非饱和顶点的边可产生最小边覆盖，例如 $M \cup \{e_6\}$ ,  $M \cup \{e_7\}$ ,  $M \cup \{e_8\}$ 都是最小边覆盖；而对于它的任意一个最小边覆盖，例如 $W = \{e_1, e_3, e_6\}$ ，从中删除相邻边的其中一条边都可得到最大匹配，例如 $W - \{e_3\}$ ,  $W - \{e_6\}$ 都是最大匹配。一般地说，我们有：

**定理 5.2.6** 给定图 $G = (V, E)$ 无孤立点， $|V| = n$ ：

- (1) 设 $M$ 为 $G$ 的一个最大匹配，对于 $G$ 的每个 $M$ 非饱和点都取一条与其关联的边组成边集 $N$ ，则 $W = M \cup N$ 是 $G$ 的最小边覆盖；

(2) 设 $W_1$ 是 $G$ 的一个最小边覆盖, 若 $W_1$ 中存在相邻的边就删除其中的一条, 继续这个过程, 直到无相邻的边, 设删除的边构成集合 $N_1$ , 即 $M_1 = W_1 - N_1$ 是 $G$ 的最大匹配;

(3)  $G$ 的边覆盖数 $\alpha_1$ 与匹配数 $\beta_1$ 满足:  $\alpha_1 + \beta_1 = n$ 。

**证明** 因为 $M$ 是最大匹配, 所以 $M$ 中的边数等于 $\beta_1$ , 那么 $G$ 有 $n - 2\beta_1$ 个 $M$ 非饱和点。显然 $N$ 中的边只关联一个 $M$ 非饱和点 (否则的话 $M$ 就不是最大匹配), 因此 $N$ 中的边数为 $n - 2\beta_1$ 。显然 $W = M \cup N$ 是边覆盖, 且:

$$|W| = |M| + |N| = \beta_1 + n - 2\beta_1 = n - \beta_1$$

另一方面, 由 $W_1$ 是最小边覆盖,  $W_1$ 中的任意边至多只有一个端点与 $W_1$ 中的其他边关联 (否则从 $W_1$ 删除这条边仍然是边覆盖, 与 $W_1$ 是最小边覆盖矛盾), 因而在由 $W_1$ 构造 $M_1$ 时, 每删除相邻两条边中的一条时, 产生并只产生一个 $M_1$ 非饱和点, 也就是说, 删除的边数等于 $M_1$ 非饱和点数, 即:

$$|N_1| = |W_1| - |M_1| = n - 2|M_1|$$

整理后得到 $|W_1| = \alpha_1 = n - |M_1|$ 。显然 $M_1$ 是匹配, 即 $|M_1| \leq \beta_1$ , 而 $W$ 是覆盖, 即 $|W| \geq \alpha_1$ , 从而:

$$\alpha_1 = n - |M_1| \geq n - \beta_1 = |W| \geq \alpha_1$$

这表明 $|M_1| = \beta_1$ , 即 $M_1$ 是最大匹配, 而且 $|W| = \alpha_1$ , 即 $W$ 是最小边覆盖, 且 $\alpha_1 + \beta_1 = n$ 。□

**推论 5.2.7** 给定图 $G = (V, E)$ 无孤立点,  $|V| = n$ 。  $M$ 是 $G$ 的匹配,  $W$ 是 $G$ 的边覆盖, 则 $|M| \leq |W|$ , 等号成立时 $M$ 是 $G$ 的完美匹配而 $W$ 是 $G$ 的最小边覆盖。

**证明** 上面定理表明最小边覆盖删除相邻边之一后可得到最大匹配, 因此 $\beta_1 \leq \alpha_1$ , 从而 $|M| \leq \beta_1 \leq \alpha_1 \leq |W|$ 。当等号成立时, 说明 $M$ 是最大匹配而 $W$ 是最小边覆盖 (否则与 $\beta_1 \leq \alpha_1$ 矛盾), 从而 $\alpha_1 = |W| = |M| = \beta_1$ , 从而 $\alpha_1 + \beta_1 = 2\beta_1 = n$ , 从而 $G$ 没有 $M$ 非饱和点 (因为 $M$ 非饱和点个数等于 $n - 2\beta_1$ ), 这表明 $M$ 是完美匹配。□

下面我们考虑怎样判断一个匹配是否是最大匹配, 首先我们定义匹配的可增广的交错路径:

**定义 5.2.8** 称 $G$ 中在 $M$ 和 $E - M$ 中交替取边的回路称为 $M$ 的**交错回路**。称 $G$ 中在 $M$ 和 $E - M$ 中交替取边的**初级道路**称为 $M$ 的**交错路径**, 起点和终点都是 $M$ 非饱和点的交错路径称为**可增广交错路径**。

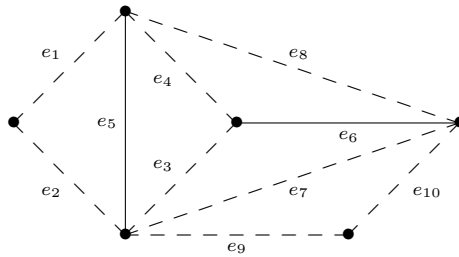
**引理 5.2.9** 设 $M$ 是图 $G$ 的匹配,  $\Gamma$ 是 $M$ 的可增广交错路径, 则 $M' = \Gamma \oplus M$ 也是 $G$ 的匹配, 且 $|M'| = |M| + 1$ 。这里 $\oplus$ 是异或操作, 也即 $M'$ 是取在 $M$ 或在 $\Gamma$ 中, 但不同时在 $M$ 和 $\Gamma$ 中的边构成的集合。

**证明** 对 $M'$ 中的任意两条边 $u, v$ , 有四种情况:

- (1)  $u \in M - \Gamma$ 且 $v \in M - \Gamma$ , 那么由于 $M$ 是匹配, 从而 $u, v$ 不相邻;
- (2)  $u \in \Gamma - M$ 且 $v \in \Gamma - M$ , 那么由于 $\Gamma$ 是交错路径, 在 $M$ 和 $E - M$ 中交替取边, 而且不存在重复的顶点 (是初级道路), 从而 $u, v$ 不相邻;
- (3)  $u \in \Gamma - M$ 而 $v \in M - \Gamma$ , 那么注意到 $u$ 的两个端点都是 $M$ 饱和点, 因此 $v$ 不与 $u$ 相邻;
- (4)  $v \in \Gamma - M$ 而 $u \in M - \Gamma$ , 与上一情况类似,  $v$ 与 $u$ 不相邻。

因此 $M'$ 中的任意两条边都不相邻, 也即 $M'$ 是图 $G$ 的匹配。又 $\Gamma$ 的两个端点都是 $M$ 非饱和点, 从而 $\Gamma$ 中在 $M$ 中的边比不在 $M$ 中的边少一条, 从而 $|M'| = |M| + 1$ 。□

**例子 5.2.10** 考虑下面的图:



其中  $M = \{e_5, e_6\}$  是匹配,  $\Gamma_1 = e_1e_5e_3e_6e_8$  是  $M$  的交错路径, 但不是可增广交错路径, 因为它的两个端点不都是  $M$  非饱和点, 而  $\Gamma_2 = e_1e_5e_3e_6e_{10}$  是  $M$  的可增广交错路径, 显然  $M' = \Gamma_2 \oplus M = \{e_1, e_3, e_{10}\}$  是比  $M$  更大的匹配。

上面的引理实际上给出了在图中寻找最大匹配的方法, 即从任意一个初始匹配开始 (例如只有一条边的匹配), 通过寻找该匹配的可增广交错路径可得到一个更大的匹配, 直到得到一个不存在可增广交错路径的匹配。下面的定理说明了不存在可增广交错路径的匹配确实是最大匹配:

**定理 5.2.11**  $M$  是  $G$  的最大匹配当且仅当  $G$  不含  $M$  可增广的交错路径。

**证明** 由上面的引理我们知道, 当  $M$  是最大匹配时, 它不含  $M$  可增广交错路径。下面只要证明若匹配  $M$  不含可增广交错路径时是最大匹配。设  $M_1$  是  $G$  的最大匹配, 只要证明  $|M| = |M_1|$  即可。

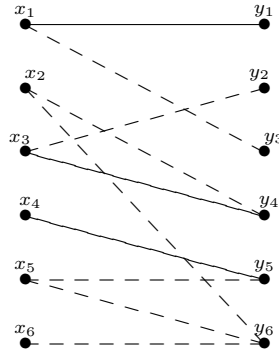
为此考虑  $M_1$  和  $M$  的对称差得到的边集导出的子图 (即这些边, 以及与这些边关联的顶点构成的图), 设该子图是  $H$ 。如果  $H = \emptyset$  是空图, 即  $M = M_1$ , 于是  $M$  就是最大匹配。若  $H \neq \emptyset$ , 则  $H$  的各连通分支只可能是下面三种情况 (否则与  $M$  和  $M_1$  都是匹配矛盾):

- (1) 孤立点: 当  $(v_i, v_j) \in M \cap M_1$  时, 出现孤立点  $v_i$  和  $v_j$ ;
- (2) 一个初级回路: 那么由于  $M$  和  $M_1$  都是匹配, 它们之中的边不相邻, 因此该回路必然在  $M$  和  $M_1$  中交替取边, 也即在这个回路中  $M$  和  $M_1$  的边相等;
- (3) 一条初级道路: 那么同样该初级道路是在  $M$  和  $M_1$  交替取边的交错路径, 而且这个道路的两个端点所关联的边不可能同时是  $M$  中的边 (这与  $M_1$  是最大匹配无可增广交错路径矛盾), 也不可能同时是  $M_1$  中的边 (这与  $M$  无可增广交错路径矛盾), 因此这个道路中  $M$  和  $M_1$  的边数也相等。

总之, 无论何种情况, 总有  $M$  的边数与  $M_1$  的边数相等, 也即  $|M| = |M_1|$ , 也即  $M$  是最大匹配。□

上面的定理表明可通过寻找匹配的可增广交错路径扩大匹配, 但在一般图中, 寻找可增广交错路径并不是一件简单任务, 所以我们下面只讨论在二部图中寻找最大匹配的算法。一个有效的算法称为**匈牙利算法**, 其基本思想是在给定一个初始匹配之后, 通过不断地寻找其可增广交错路径而扩大该匹配, 直到不存在可增广交错路径。

**例子 5.2.12** 我们使用戴一奇教材[2]p90的例子5.1.3来说明匈牙利算法的基本思想。考虑下面的二部图  $G = (X, Y, E)$ , 其中  $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ ,  $Y = \{y_1, y_2, y_3, y_4, y_5, y_6\}$ 。在图中我们用实线给出了初始的匹配  $M = \{(x_1, y_1), (x_3, y_4), (x_4, y_5)\}$ 。匈牙利算法的基本思想是寻找匹配的可增广交错路径, 从而扩大该匹配。



为了寻找匹配的可增广交错路径, 对 $X$ 中的每个顶点进行标记, 饱和点标记1, 没有搜索的点标记0, 无法扩大匹配的顶点标记为2。对于上述初始匹配, 初始标记是将 $M$ 饱和点标记1, 其他顶点都标记为2, 即:

$$\begin{array}{lll} X: \text{标记1的顶点: } \{x_1, x_3, x_4\} & \text{标记0的顶点: } \{x_2, x_5, x_6\} & \text{标记2的顶点: } \emptyset \\ Y: \text{标记1的顶点: } \{y_1, y_4, y_5\} & \text{标记0的顶点: } \{y_2, y_3, y_6\} & \text{标记2的顶点: } \emptyset \end{array}$$

在 $X$ 中选一个标记为0的顶点 $u$ , 寻找以该顶点为起点的可增广交错路径。寻找的方法是使用集合 $U$ 记录 $X$ 中可能在该交错路径中的顶点, 用 $V$ 记录 $Y$ 中可能在该交错路径中的顶点。这两个集合中的顶点的添加方式满足:

1.  $V$ 的初始值是空集,  $V$ 中的顶点都是 $U$ 中顶点的相邻顶点, 且除了最后一个添加的顶点可能是非饱和点之外, 其他顶点都是饱和顶点, 如果最后发现一个非饱和顶点, 则找到一个可增广交错路径, 否则初始选的顶点 $v$ 无法扩大匹配;
2.  $U$ 中的初始顶点是 $u$ , 其后加入的顶点都是根据 $V$ 中的顶点 $v$ , 选择一条在匹配 $M$ 中的边 $(v, u')$ 而加入 $u'$ 。

具体来说, 对于上面的初始匹配, 我们选择标记为0的顶点 $x_2$ , 记 $U = \{x_2\}$ ,  $V = \emptyset$ 。考虑 $x_2$ 的相邻顶点 $y_4$ 和 $y_6$ , 由于 $y_6$ 标记为0, 是 $M$ 非饱和顶点, 优先考虑 $y_6$ , 就得到一条可增广交错路径 $(x_2, y_6)$ , 将这条边加入 $M$ , 扩大 $M$ 为 $M = \{(x_1, y_1), (x_2, y_6), (x_3, y_4), (x_4, y_5)\}$ 。将 $x_2$ 和 $y_6$ 的标记改为1, 得到:

$$\begin{array}{lll} X: \text{标记1的顶点: } \{x_1, x_2, x_3, x_4\} & \text{标记0的顶点: } \{x_5, x_6\} & \text{标记2的顶点: } \emptyset \\ Y: \text{标记1的顶点: } \{y_1, y_4, y_5, y_6\} & \text{标记0的顶点: } \{y_2, y_3\} & \text{标记2的顶点: } \emptyset \end{array}$$

对于扩大后的匹配, 选择标记为0的顶点 $x_5$ , 令 $U = \{x_5\}$ ,  $V = \emptyset$ 。考虑 $x_5$ 的相邻顶点 $y_5, y_6$ , 由于它们都已经标记1, 因此随意选择一个顶点, 例如 $y_5$ 加入到 $V$ , 然后选择 $y_5$ 关联的在 $M$ 中的边 $(x_4, y_5)$ , 将 $x_4$ 加入 $U$ , 得到:

$$U = \{x_5, x_4\} \quad V = \{y_5\}$$

下一步考虑与 $x_5, x_4$ 相邻的所有顶点 $y_5, y_6$ , 由于 $y_5$ 已经在 $V$ 中, 这一次考虑 $y_6$ , 它的标记也为1, 将其加入 $V$ , 并选择 $y_6$ 关联的在 $M$ 中的边 $(x_2, y_6)$ , 将 $x_2$ 加入 $U$ , 得到:

$$U = \{x_5, x_4, x_2\} \quad V = \{y_5, y_6\}$$

再考虑与 $x_5, x_4, x_2$ 相邻且不在 $V$ 中的顶点 $y_4$ ,  $y_4$ 的标记也为1, 将其加入 $V$ , 并选择 $y_4$ 关联的在 $M$ 中的边 $(x_3, y_4)$ , 将 $x_3$ 加入 $U$ , 得到:

$$U = \{x_5, x_4, x_2, x_3\} \quad V = \{y_5, y_6, y_4\}$$

考虑与 $U$ 中顶点相邻且不在 $V$ 中的顶点 $y_2$ ,  $y_2$ 标记为0, 是 $M$ 非饱和顶点, 因此存在以 $x_5$ 为起点的可增广路径, 该路径可通过回溯加入 $y_2$ 的过程得到:

$$\Gamma = y_2 x_3 y_4 x_2 y_6 x_5$$

将 $M$ 扩大为 $M = M \oplus \Gamma = \{(x_1, y_1), (x_2, y_4), (x_3, y_2), (x_4, y_5), (x_5, y_6)\}$ 。将 $x_5$ 和 $y_2$ 的标记改为1, 得到:

$X$ : 标记1的顶点:  $\{x_1, x_2, x_3, x_4, x_5\}$       标记0的顶点:  $\{x_6\}$       标记2的顶点:  $\emptyset$

$Y$ : 标记1的顶点:  $\{y_1, y_2, y_4, y_5, y_6\}$       标记0的顶点:  $\{y_3\}$       标记2的顶点:  $\emptyset$

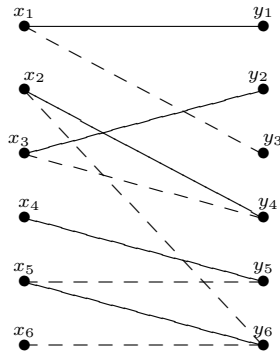
对于扩大后的匹配, 选择标记为0的顶点 $x_6$ , 令 $U = \{x_6\}, V = \emptyset$ , 考虑 $x_6$ 的相邻顶点 $y_6$ ,  $y_6$ 已经标记为1, 将其加入 $V$ , 并选择 $y_6$ 关联的在 $M$ 中的边 $(x_5, y_6)$ , 将 $x_5$ 加入 $U$ , 得到:

$$U = \{x_6, x_5\} \quad V = \{y_6\}$$

考虑与 $U$ 中的顶点相邻且不在 $V$ 中的顶点 $y_5$ ,  $y_5$ 也标记为1, 将其加入 $V$ , 并选择 $y_5$ 关联的在 $M$ 中的边 $(x_4, y_5)$ , 将 $x_4$ 加入 $U$ , 得到:

$$U = \{x_6, x_5, x_4\} \quad V = \{y_6, y_5\}$$

这时 $U$ 中的顶点没有不在 $V$ 中的相邻顶点, 这意味着以 $x_6$ 为起点的交错路径都只会以饱和点结束, 因此不存在以 $x_6$ 为起点的可增广交错路径, 这表明 $x_6$ 无法扩大匹配, 标记为2。到此, 所有在 $X$ 中的顶点已经被标记为1或2, 算法结束, 而在 $Y$ 中仍标记为0的顶点也是无法参与匹配的顶点。最后得到的最大匹配就是:  $M = \{(x_1, y_1), (x_2, y_4), (x_3, y_2), (x_4, y_5), (x_5, y_6)\}$ , 下图用实线给出了该匹配:



### 5.3 二部图中的匹配

下面我们转为讨论二部图中的匹配, 因为二部图中的匹配问题有更强的应用背景, 而且寻找二部图的最大匹配也比寻找一般图的最大匹配简单。首先回忆二部图的概念:



**定义 5.3.1** 给定图  $G = (V, E)$ , 如果  $V$  能分为两个不相交的非空子集  $V_1, V_2$ , 即  $V_1 \neq \emptyset, V_2 \neq \emptyset, V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$ , 且对  $G$  的任意边  $e = (u, v) \in E$ , 都有  $u \in V_1$  且  $v \in V_2$ , 或者  $v \in V_1$  且  $u \in V_2$ , 也即  $G$  的任意边的两个端点都在不同的子集, 则称  $G$  是**二部图**。若  $|V_1| = s, |V_2| = r$ , 且  $V_1$  的任意顶点都与  $V_2$  的任意顶点有边, 则称该二部图为**完全二部图**, 记为  $K_{s,r}$ 。

我们通常记二部图为  $G = (V_1, V_2, E)$ , 以明确给出它的两个不相交的非空顶点子集。在二部图中, 我们通常考虑其完备匹配:

**定义 5.3.2** 给定二部图  $G = (V_1, V_2, E)$ , 设  $|V_1| \leq |V_2|$ , 若  $M$  是  $G$  的最大匹配, 且  $|M| = |V_1|$ , 则称  $M$  为  $V_1$  到  $V_2$  的**完备匹配** (或称完全匹配)。

直观地说, 图的匹配就是将顶点进行某种配对, 最大匹配就是参与的顶点数尽可能多的匹配, 完美匹配就是所有顶点都参与的匹配, 而对于二部图而言, 我们往往更关注其中的一个顶点集 (当然是顶点数集少的那个顶点集) 是否都参与了匹配, 因此有所谓的完备匹配。显然在上述定义中, 当  $|V_1| = |V_2|$  时, 因为二部图的每条边一定分别关联  $V_1$  和  $V_2$  的一个顶点, 因此完备匹配就是完美匹配, 而当  $|V_1| < |V_2|$  时, 二部图不可能有完美匹配, 这时完备匹配只是最大匹配而已。

对于二部图, 我们可以判断其是否存在完备匹配:

**定理 5.3.3 Hall定理:** 给定二部图  $G = (V_1, V_2, E)$ , 设  $|V_1| \leq |V_2|$ ,  $G$  存在  $V_1$  到  $V_2$  的完备匹配当且仅当  $V_1$  中任意  $k$  ( $k = 1, 2, \dots, |V_1|$ ) 个顶点至少与  $V_2$  中的  $k$  个顶点相邻。该条件常称为二部图存在完备匹配的**相异性条件**。

**证明** 若  $G$  中存在  $V_1$  的  $k$  个顶点之多与  $V_2$  中的  $k-1$  个顶点相邻, 显然, 这  $V_1$  的这  $k$  个顶点不可能都由  $V_2$  的顶点与它匹配, 从而  $G$  不存在从  $V_1$  到  $V_2$  的完备匹配。

下面我们证, 当  $V_1$  的任意  $k$  个顶点至少与  $V_2$  的  $k$  个顶点相邻时,  $G$  的最大匹配就是  $V_1$  到  $V_2$  的完备匹配。反证法, 设  $M$  是  $G$  的最大匹配, 但不是  $G$  的完备匹配, 也即存在  $v_0 \in V_1$  是  $M$  非饱和点, 那么首先在  $V_2$  中存在与  $v_0$  相邻的顶点, 因为根据相异性条件,  $v_0$  不是孤立点, 其次与  $v_0$  相邻的  $V_2$  中的顶点都是  $M$  饱和点, 因为若有顶点  $u \in V_2$  与  $v_0$  相邻, 且是  $M$  非饱和点, 则  $M \cup \{(v_0, u)\}$  仍是  $G$  的匹配, 与  $M$  是最大匹配矛盾。

现在考虑从  $v_0$  出发的尽可能长的交错路径, 由于  $M$  是最大匹配, 因此这些交错路径都不是可增广交错路径, 即每条路径的另一个端点必然是  $M$  饱和点, 于是这些交错路径的长度为偶数长, 从而由二部图的性质, 这些路径的端点全部在  $V_1$  中。定义:

$$S = \{u \mid u \in V_1 \text{ 且 } u \text{ 在从 } v_0 \text{ 出发的交错路径上}\}$$

$$T = \{u \mid u \in V_2 \text{ 且 } u \text{ 在从 } v_0 \text{ 出发的交错路径上}\}$$

由于所有交错路径的端点全在  $S$  中,  $S$  中除了  $v_0$  之外, 其它顶点均与  $T$  中的顶点配对, 所以  $|S| = |T| + 1$ 。显然与  $S$  中顶点相邻的顶点都在  $T$  中, 从而  $V_1$  中的  $|S| = |T| + 1$  个顶点只与  $V_2$  中  $|T|$  个顶点相邻, 与相异性条件矛盾! 所以  $V_1$  中不存在  $M$  非饱和点, 即  $M$  是完备匹配。□

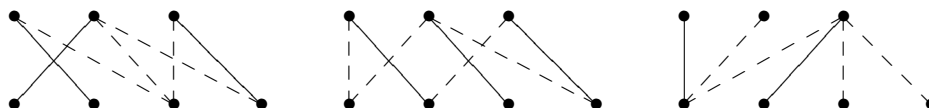
由上述定理, 我们还可以得到一个更容易判断的条件, 称为二部图完备匹配的 **$t$ 条件**:

**推论 5.3.4** 给定二部图  $G = (V_1, V_2, E)$ , 设  $|V_1| \leq |V_2|$ 。若  $V_1$  中每个顶点至少关联  $t$  ( $t \geq 1$ ) 条边, 而  $V_2$  中每个顶点至多关联  $t$  条边, 则  $G$  存在  $V_1$  到  $V_2$  的完备匹配。

**证明** 由于 $V_1$ 中每个顶点至少关联 $t$ 条边,从而 $V_1$ 的任意 $k(1 \leq k \leq |V_1|)$ 个顶点至少关联 $kt$ 条边,又由于 $V_2$ 的每个顶点之多关联 $t$ 条边,所以 $kt$ 条边至少关联 $V_2$ 中 $k$ 个顶点,根据相异性条件, $G$ 存在 $V_1$ 到 $V_2$ 的完备匹配。□

显然 $t$ 条件比相异性条件要强,也就是说,满足相异性条件不一定满足 $t$ 条件。

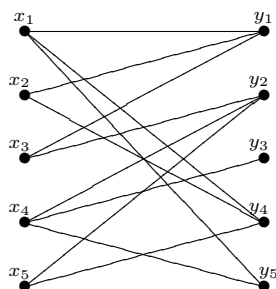
**例子 5.3.5** 考虑下面三个图:



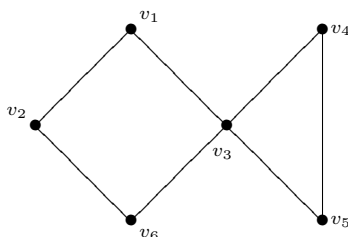
上面最左边的图满足相异性条件,但不满足 $t$ 条件,中间的图既满足相异性条件,又满足 $t$ 条件,这时 $t = 2$ ,因此这两个图都有完备匹配,正如图中实线所示。上面最右边的图不满足相异性条件,有 $V_1$ (上排顶点)的两个顶点只与 $V_2$ (下排顶点)的一个顶点相邻,因此它不存在完备匹配。

## 作业

**作业 5.1** 戴一奇教材p114的习题五的第一题:给出初始匹配 $M = \{(x_1, y_1), (x_4, y_2)\}$ ,求下面二部图的最大匹配:



**作业 5.2** 考虑下面的图:



- (1) 给出上图的两个不同的极小支配集、一个最小支配集及支配数 $\gamma_0$ ;
- (2) 给出上图的两个不同的极大点独立集、一个最大点独立集及点独立数 $\beta_0$ ;
- (3) 给出上图的两个不同的极小点覆盖集、一个最小点覆盖集及点覆盖数 $\alpha_0$ 。

**作业 5.3** 现有4名教师,张、王、李、赵,要求他们去教4门课程:数学、物理、英语和化学。已知张能胜任数学和化学;王能胜任物理和英语;李能胜任数学、物理和英语,而赵只能胜任英语。请问应该如何安排,才能使每位教师都能教一门自己能够胜任的课程,并且每门课都有人教?

## 参考文献

- [1] 耿素云、屈婉玲. 离散数学 (修订版). 高等教育出版社, 2004年1月.
- [2] 戴一奇、胡冠章、陈卫. 图论与代数结构. 清华大学出版社, 1995.