第二章　树

本次课主要内容

(一)、生成树的计数

(二)、求生成树的方法

(三)、最小生成树

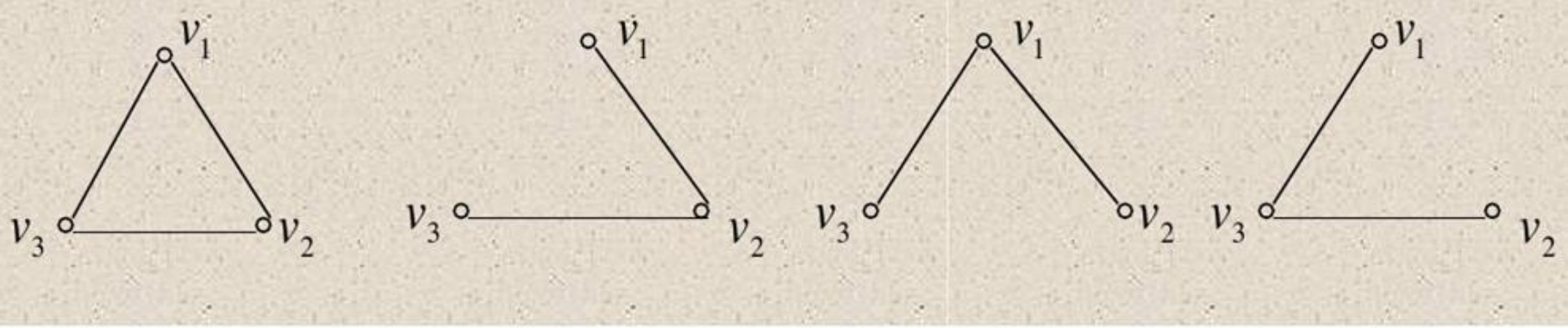
(四)、根树及其应用

- 前面我们介绍了生成树的存在性问题，既然我们已经知道了一个连通图中一定存在生成树，很自然我们就要去考察连通图的生成树的数目—生成树的计数

（一）、生成树的计数

- 设G是连通图，T，T’是G 的两个生成树， 如E(T)与E(T’)不同， 我们说这是两个不同的生成树。
- 记G的生成树的个数为 $\tau(G)$

如：



则：$\tau(K_3)=3$。

1、凯莱递推计数法

凯莱(Cayley 1821—1895): 剑桥大学数学教授，著名代数学家，发表论文数仅次于Erdos , Euler, Cauchy. 著名成果是1854年定义了抽象群，并且得到著名定理：任意一个群都和一个变换群同构。同时，他也是一名出色的律师，作律师14年期间，发表200多篇数学论文，著名定理也是在该期间发表的。

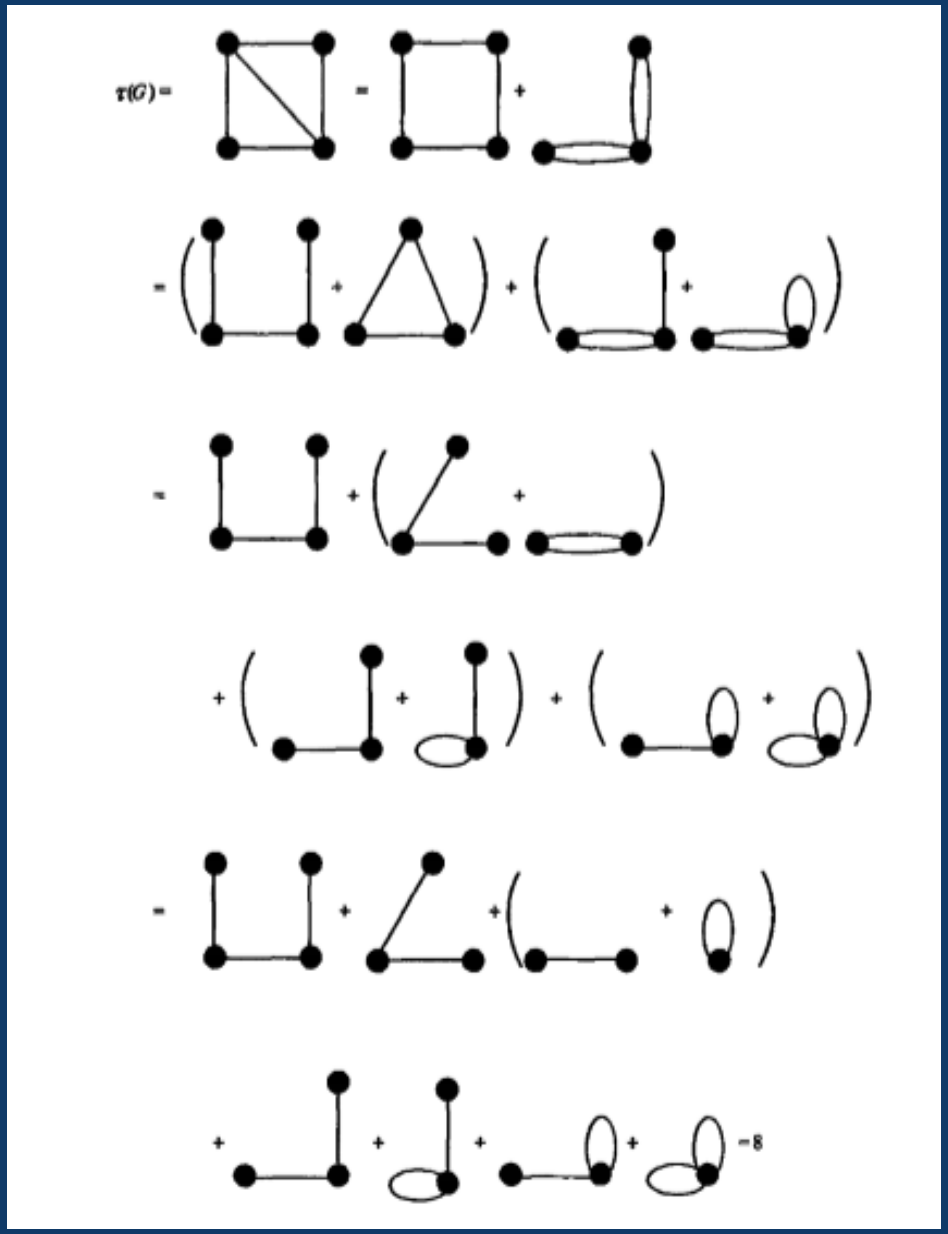
凯莱生成树递推计数公式是他在1889年建立的。

有趣的插曲：Erdos number：埃尔德什和别人合写的论文实在太多了，有人定义了埃尔德什数，简称埃数。埃尔德什的埃尔德什数为0，与他直接合作写论文的人的埃数为1，与埃数为1的人合写论文的人埃数为2，依此类推。https://www.oakland.edu/enp/ 查Erdos number

定理（Cayley）设e是连通图G的一条边，则有：

$\tau(G)=\tau(G-e)+\tau(Goe)$，其中Goe代表G中收缩边e，也就是前面定义的 G/e

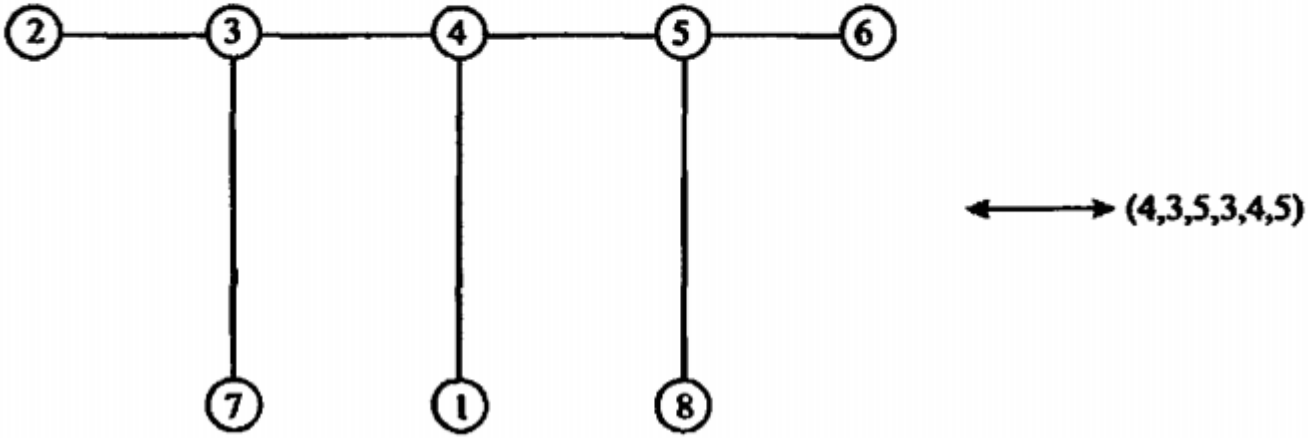
证明：对于$e\in E(G)$，G中不含边e的生成树的个数为τ（G-e），G中含边e的生成树个数为 τ（Goe），所以结论成立。



定理 (Cayley) $\tau(K_n)=n^{n-2}$

证 令 $V(K_n)=\{1,2,3,\cdots,n\}$. 于是 $V(K_n)$ 中元素为分量构成的 $n-2$ 个分量的向量共计 n^{n-2} 个. 下面建立这 n^{n-2} 个向量与 K_n 的生成树集合间的一一对应.

任取 K_n 的一个生成树 T, 设 s_1 是 T 上最小(指此顶所对应的$\{1,2,\cdots,n\}$中的数最小)的叶, t_1 为 s_1 的邻顶, 把 s_1 从 T 上删除; 设 s_2 是 $T-s_1$ 上最小的叶, t_2 为 s_2 在 $T-s_1$ 中的邻顶. 依此类推, 即得一个由 $V(K_n)$ 中的数为分量的 $n-2$ 个分量的向量$(t_1,t_2,\cdots,t_{n-2})$, T 上还剩下一个 K_2, 见图 2.2.



反之，任给定向量$(t_1, t_2, \cdots, t_{n-2})$，$t_i \in V(K_n)$，$i=1,2,\cdots,n-2$. 我们来找出一个与此向量对应的 K_n 的生成树：s_1 是 $V(K_n)$ 中不在$(t_1, t_2, \cdots, t_{n-2})$中的最小顶，把 s_1 与 t_1 之间连一条边；s_2 是不在$(t_2, t_3, \cdots, t_{n-2})$中的 $V(K_n)-\{s_1\}$ 中的最小顶，把 s_2 与 t_2 之间连一边. 依此类推，得 $n-2$ 条边 $s_1t_1, s_2t_2, \cdots, s_{n-2}t_{n-2}$. 再连接 $V(K_n)-\{s_1, s_2, \cdots, s_{n-2}\}$ 中的两个顶，则得一棵生成树. 证毕.

注：定理的证明有好几种不同方法。1967年，加拿大的Moon用了10种不同方法证明，之后有人给出了更多证明方法。

例 证明：若e为K_n的一条边，则：

$$\tau(K_n - e) = (n-2)n^{n-3}$$

证：若e为K_n的一条边，由Kn中的边的对称性以及每棵生成树的边数为n-1，K_n的所有生成树的总边数为：

$$(n-1)n^{n-2}$$

所以，每条边所对应的生成树的棵数为：

$$\frac{(n-1)n^{n-2}}{\frac{1}{2}n(n-1)} = 2n^{n-3}$$

所以，K_n - e 对应的生成树的棵数为：

$$\tau(K_n - e) = n^{n-2} - 2n^{n-3} = (n-2)n^{n-3}$$

凯莱公式的缺点之一是计算量很大，其次是不能具体指出每棵生成树。

关联矩阵计数法 ------有兴趣的同学可以去查资料

（二）、求生成树的方法

- 广度优先搜索
- 深度优先搜索

- 广度优先和深度优先算法其实只是“生长”法的特例------对不？
- 前面我们讲题目中的破圈法，避圈法也是求生成树的方法

（三）、 最小生成树

定义 设T是无向连通带权图$G=\langle V, E, W\rangle$的一棵生成树，

T的各边权之和称为T的权，记作$W(T)$。

G的所有生成树中权最小的生成树称为G的最小生成树。

思考： 避圈法+greedy 或 破圈法+greedy 求出最小生成树

（一）、克鲁斯克尔（Kuruskal）算法

克鲁斯克尔（Kuruskal）：1928年生， 一家三兄弟都是数学家，1954年在普林斯顿大学获博士学位， 导师是Erdos. 大部分研究工作是数学和语言学， 主要在贝尔实验室工作。1956年发表包含克鲁斯克尔算法的论文， 使他名声大振。

Kruskal算法（避圈法 +greedy）

(1) 设n阶无向连通带权图$G=<V, E, W>$有m条边。不妨设G中没有环（否则，可以将所有的环先删去），将m条边按权从小到大排序：$e_1, e_2, \ldots, e_m$。

(2) 取e_1在T中。

(3) 依次检查$e_2, \ldots, e_m$，若e_j $(j \geqslant 2)$与已在T中的边不构成圈，取e_j也在T中，否则弃去e_j。

(4) 当T所含的边数为n-1时算法停止得到的T为G的最小生成树。

Kruskal算法的正确性

定理 不妨设$e_1, e_2, \ldots, e_{n-1}$是kruskal算法获得的边， 则边导出子图G[{$e_1, e_2, \ldots, e_{n-1}$}]是G的最小生成树。

证明： 记T*= G[{$e_1, e_2, \ldots, e_{n-1}$}]. 显然T*无圈，因此T*是森林。现证T*连通。反证：假设它有w个连通分支(w≥2)，则|V(T*)|=|E(T*)|+w>|E(T*)|+1=|V(G)|，

与T*是G的子图矛盾。 故T连通。

Kruskal算法的正确性

下证其最优性。 反证法， 假设T*不是最小权生成树（下称最优树). 对G的任何一棵生成树， 记

f(T)=min{ i |$e_i \in \{e_1, e_2, \ldots, e_{n-1}\}$且$e_i \notin T$}

在G的所有最优树中选取使得f(T)最大的，记为T’ (T’不会是T*)。

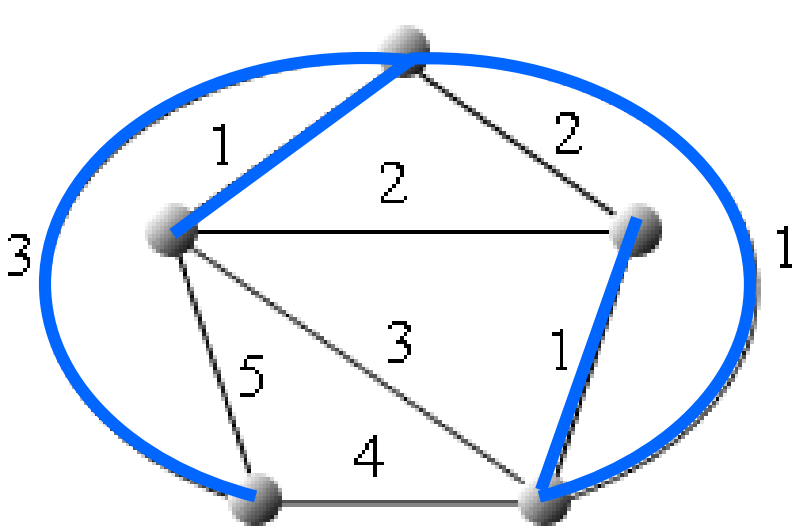
设f(T’)=k 。 由于f(T’)的定义， $e_1, e_2, \ldots, e_{k-1}$ 既在T*上又在T’上。且T’+e_k含有一个圈C 。 C上必有一条边e'_k, 而$e_k \notin T^*$ 。显然T_1=T’+ e_k -e'_k也是一棵生成树，且w(T_1)=w(T’)+w(e_k)-w(e'_k) 。按照算法，e_k是使得G[{$e_1, e_2, \ldots, e_{k-1}, e_j$ }]中无圈的边中权最小的。 注意G[{$e_1, e_2, \ldots, e_{k-1}$,$e'_k$}]也是T’ 的子图，也无圈。故由算法规则知道 w(e'_k) ≥w(e_k). 因此w(T_1)≤w(T’), 这说明T_1也是最优树， 但f(T_1)>k=f(T’),与取T’的取法矛盾。 证毕。

管梅谷的破圈法

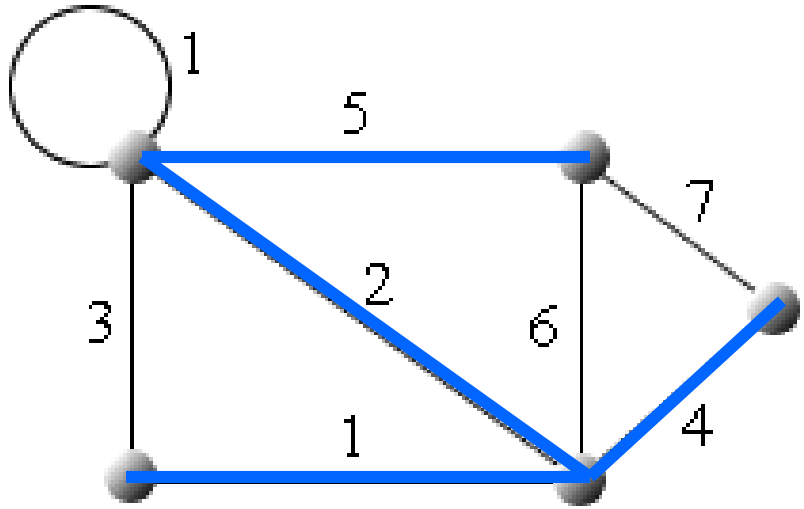
- 该方法由Rosenstiehl 和管梅谷各自独立提出。
- 基本思想： 设G是连通的赋权图。 若G不是树， 则G一定含有圈。 从赋权图G中任取一个圈， 去掉该圈中权值最大的一条边，称为破圈， 得图G1,它是G的连通生成子图； 下一步，若G1不是树，又从G1的某个圈中删去权值最大的一条边；如此下去， 最后不能按上述方式删边时得到的图不含圈是连通生成子图， 此时的图就是最小生成树。
- 破圈法+greedy

- 如何证明它的正确性呢？
- 可参看《数学的认识与实践》4,(1975),38-41。

例 求下图所示两个图中的最小生成树。



(1)



(2)

$W(T_1)=6$

$W(T_2)=12$

Prim算法

- Prim算法由Prim在1957年提出。
- 设S记录子树
- 初始化， S包含取定的任意一个点v_0
- 从S到V-S中的边中选取权最小的一条边另外一个顶点加入到S中
- 直到所有的点都被包含在S中

思考： Prim算法是不是也是避圈法+greedy？ 避圈方式不同而已？

Prim算法的正确性

- 定理： 设$e_1, e_2, \cdots, e_{n-1}$是prim算法获得的边， 则边导出子图G[{$e_1, e_2, \cdots, e_{n-1}$ }]是G的最小生成树。
- 证明： 用T_k表示Prim算法第k次循环后得到的边集所构成的子图，即

  $T_k=G[E_k]=G[\{e_1, e_2, \cdots, e_k\}]$， $k=1,2,..,n-1$

  我们用归纳法证明，对每个k(k=1,2,..,n-1)， T_k必含于G的某个最小生成树中。

  当k=1时，T_1是由算法中从初始点v_0发出的边e_1生成。对G的任意一棵最小生成树T*，若T*不含有e_1,则T*+e_1有一个圈C。设C上与v_0关联且异于e_1的边为e,则T’ =T*+ e_1 -e也是G的一颗生成树。 按照算法， e_1是G中与v_0关联的边权最小的边，因此权w(e)≥w(e_1),从而w(T’)≤w(T*).可见T’ 也是G的一棵最小生成树，包含边e_1,即T_1含于最小生成树T’ 中。

- 假设k=l时(1≤l<n-1)结论成立， 即T_l含于G的某个最小生成树T*中。考虑最后一条进入$T_{l+1}=G[\{e_1, e_2, \cdots, e_l, e_{l+1}\}]$的边$e_{l+1}$. 设$e_{l+1}=v_lv_{l+1}$

则按照算法必定一个端点v_l在T_l中， 另外一个端点v_{l+1}不在T_l中。

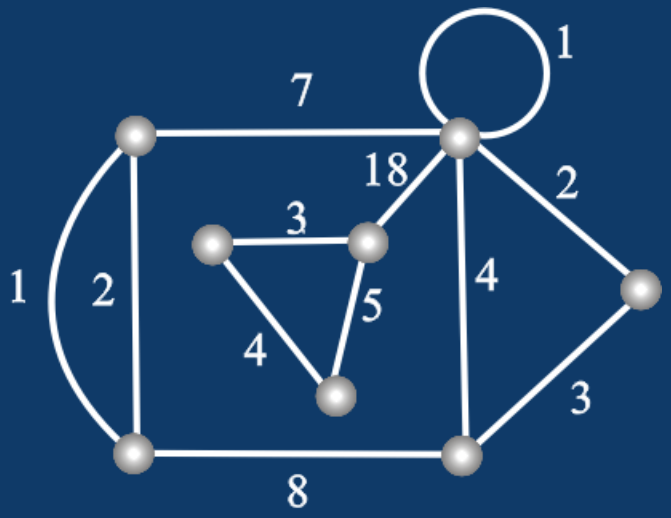
若e_{l+1}在T*中，则$T_{l+1}=T_l+e_{l+1}$含于最小生成树T*中。

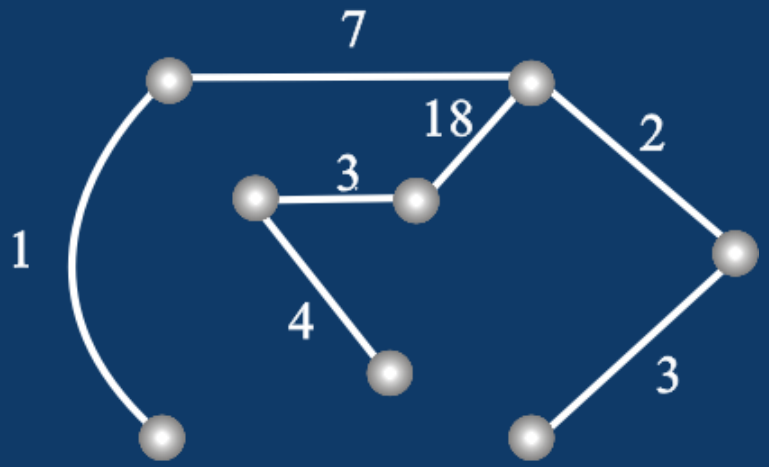
若e_{l+1}不在T*中， 则T*+ e_{l+1}中有一个圈C， e_{l+1}在圈C上。C上以v_{l+1}为端点且异于e_{l+1}的边不属于T_l(因v_{l+1}不在T_l中)。因此在C上必可找到一条异于e_{l+1}的边e=uv,它不属于T_l且满足$u\in T_l$，但v不属于T_l(因e_{l+1}的端点v_l在C上且$v_l\in T_l$). 由算法步骤知， 权$w(e)\geqslant w(e_{l+1})$. 于是T’ =T* + e_{l+1}-e也是G的一颗最小生成树，并且含边e_{l+1}。根据归纳假设T_l的边全在T*中，因此也全在T’中， 从而$T_{l+1}=T_l+e_{l+1}$的边全在最小生成树T’中。 归纳法完成。

例题

例如　求所示图的一棵最小生成树。

解答





最小生成树 $W(T)=38$

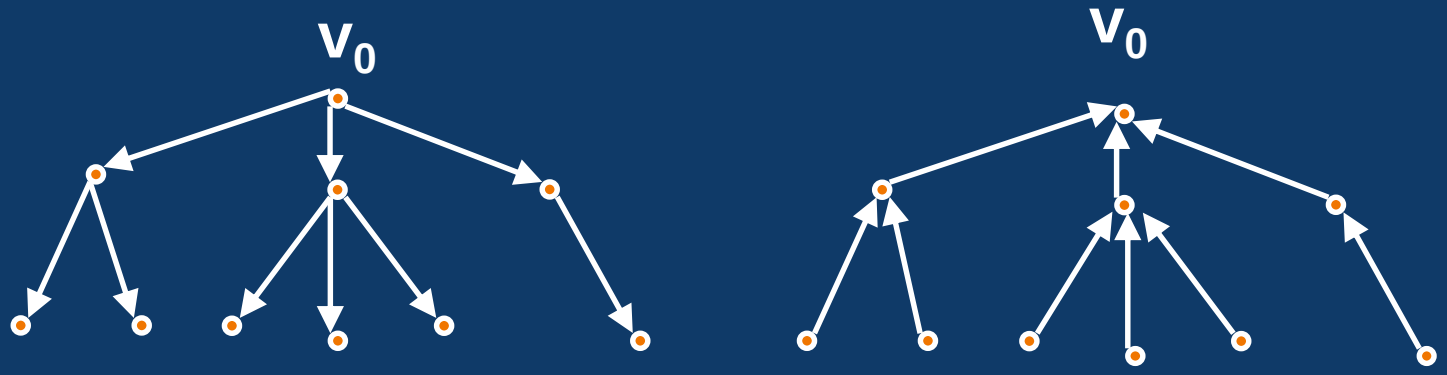
延展---有兴趣的可以去查阅资料

（1）prim算法的矩阵实现---求最小生成树的权矩阵法

（2）Prim算法的标号形式---标号Prim算法

（四）有序二元树

- **定义1：** 图T是一棵树，把每边规定一个方向且使得任意的$v_i \in V(T)$，存在有向道路$P(v_0, v_i)$，则称T是*外向树*，v_0叫做*根*，把外向树之定向反过来，得到的有向树叫*内向树*。



- 定义2：T为外向树，对任意的顶点v∈V(T)， 都有 $d^+(v) \leqslant \sigma$，则称T为 *σ元树*；
- 当e=(u, v)时，u称为v之*父*，v称为u之*子*；同父之子称为*兄弟*。
- 除叶子外，每顶点皆有σ子时，称为*典型σ元树*；
- 兄弟间有序时，叫*有序树*，有序树之序列叫做*有序林*。
- 有序树当σ=2时，就叫*有序二元树*。

- 对有序二元树的顶点编码如下：

(1) 根不标码；

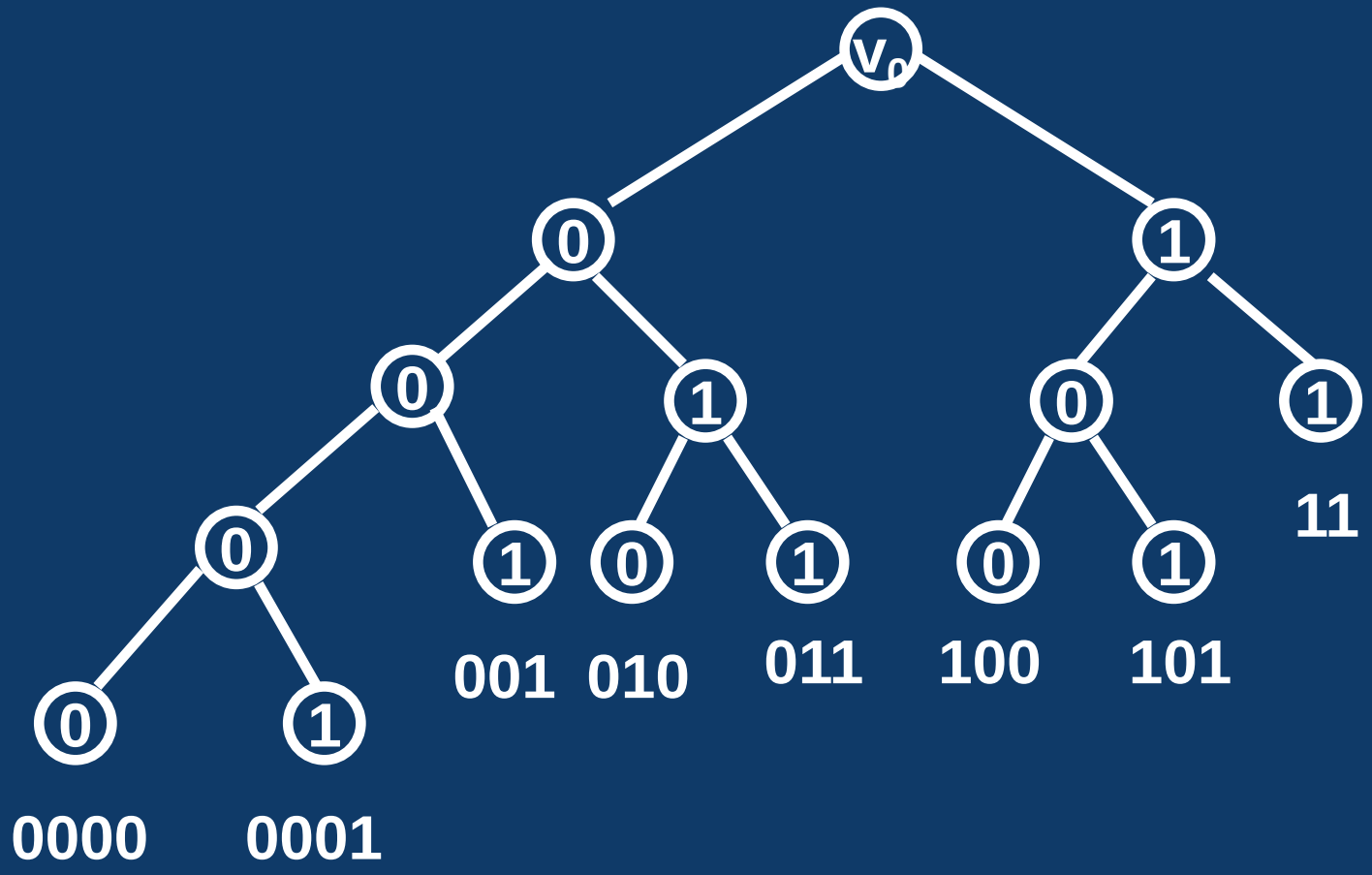
(2) 兄弟有序，左为兄，标0，右为弟，标1；

(3) 从根到叶的道路依次抄出各点之码，写在叶下方，称*该叶的前缀*；

(4) 全树的叶从左到右把它们的前缀依次抄出，叫做*该树的前缀码*，每个叶子的前缀后加逗号，最后一个叶子前缀后加句号。

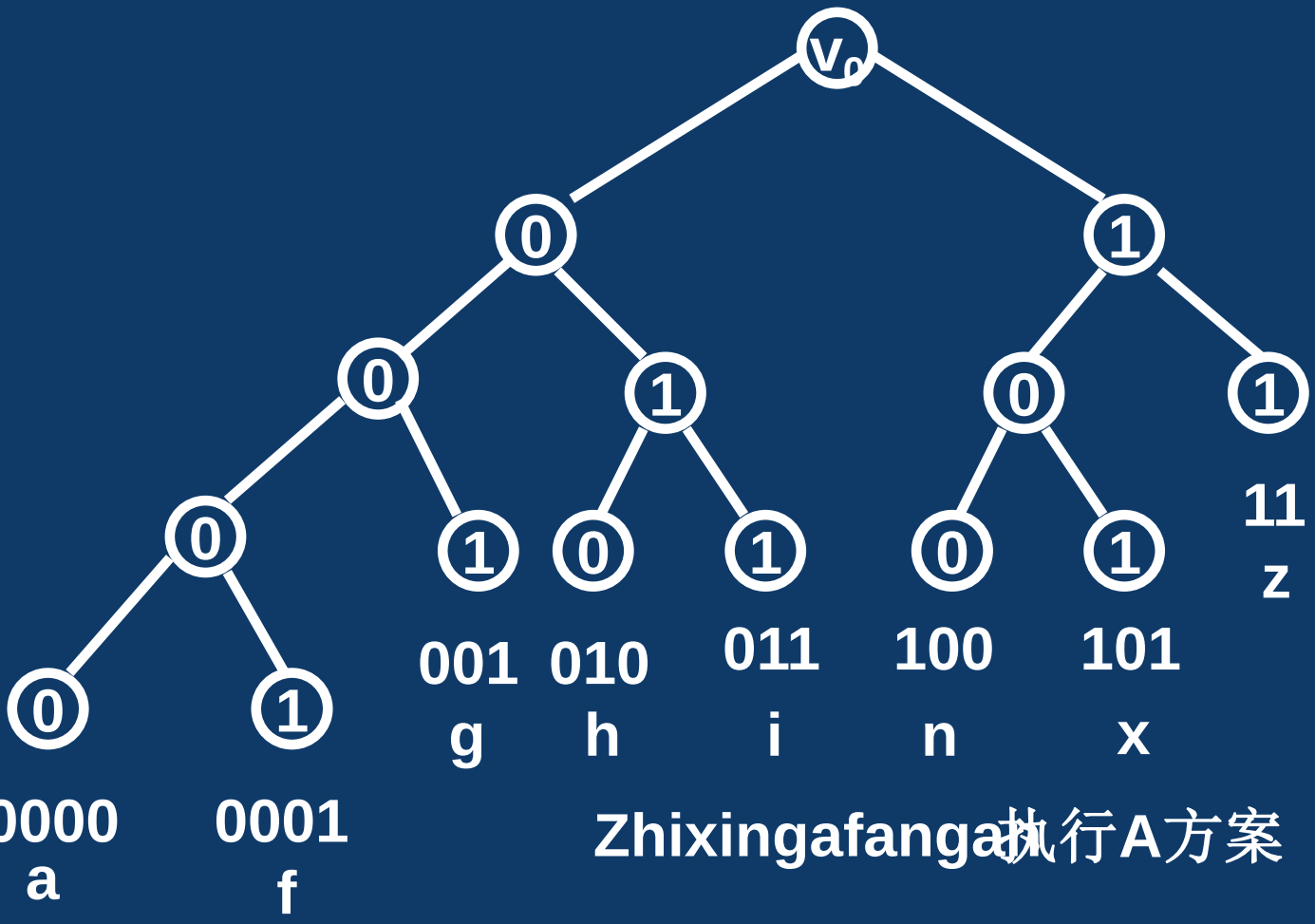
- 显然有序二元树与前缀码一一对应。

- 例如，0000，0001，001，010，011，100，101，11这一前缀码对应的有序二元树如下图所示：



取定一个有26个叶子的有序二元树，把它标出前缀码。再把每个叶子分别写上26字母，则可以用前缀码表示字母。

例如，刚才的有序二元树的叶子写上字母如下：



Zhixingafanga执行A方案

如果收到一串数字：11，010，011，101，011，100，001，0000，0001，0000，100，001，0000，100。

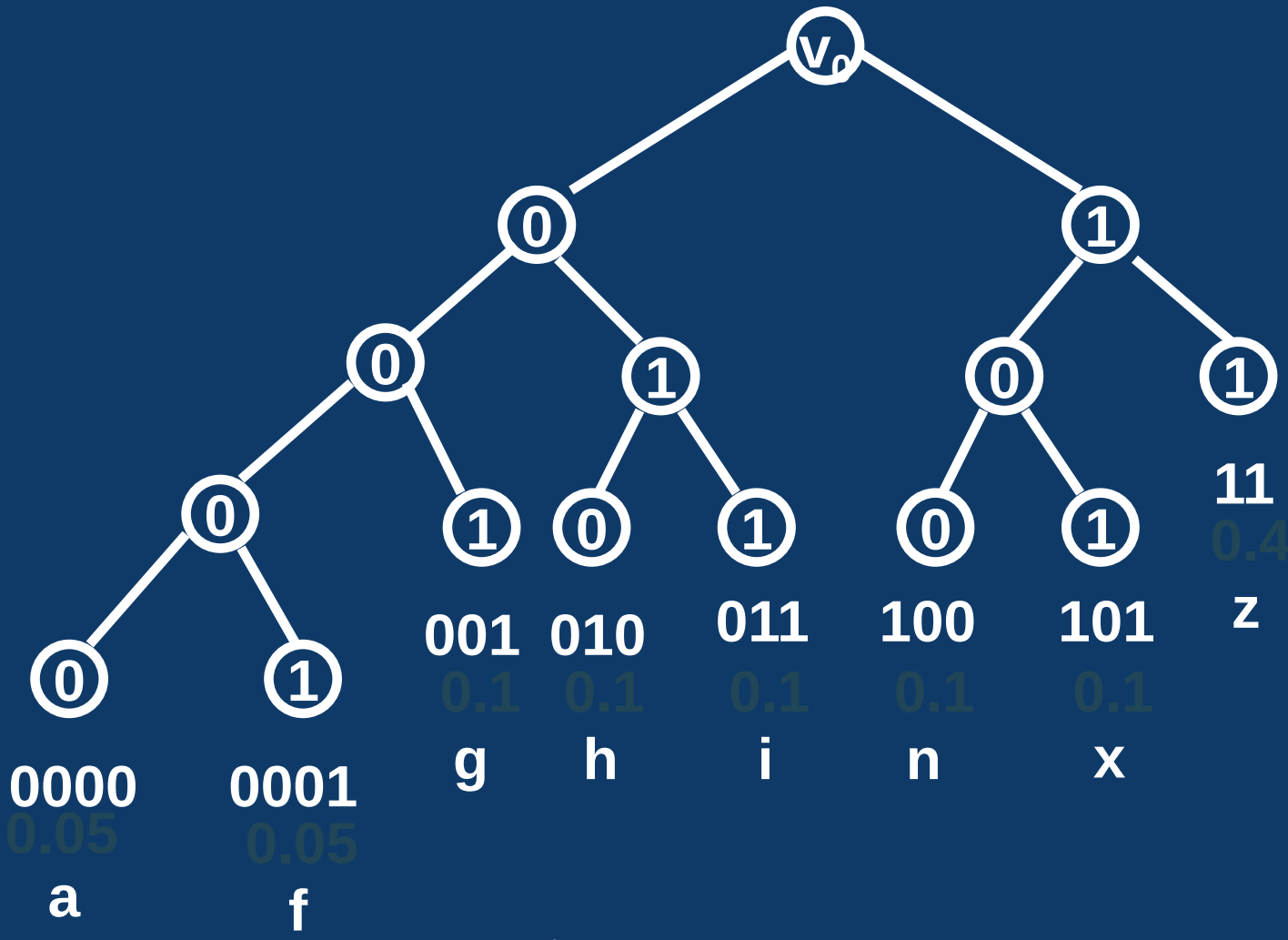
（五）Huffman树

- 定义：在v_0为根，$v_1, v_2, \ldots, v_n$为叶的有序二元树中，路径$P(v_0, v_i)$ $(i=1, 2, \ldots, n)$的长度l_i叫做叶v_i的*码长*。若v_i $(i=1, 2, \ldots, n)$代表的事物出现的概率为p_i，$\sum_{i=1}^{n} p_i = 1$，使得

$$m(T) = \sum_{i=1}^{n} p_i l_i = \min$$

有序二元树T叫做带权$p_1, p_2, \ldots, p_n$的*Huffman树*，也叫做*最优二元树*（*最优二叉树*）。

Huffman树的前缀码称为*Huffman编码*。



WPL＝4×(0.05+0.05)＋3×(0.1+0.1+0.1+0.1+0.1)+2×0.4＝2.7

Huffman树的用途很广：

- *分支程序的判断流程*：如果出现概率越大的分枝（条件语句）离根越近，那么所需执行的判断语句就越少，这样便可提高程序的执行效率；
- *文件的压缩*：根据文件中字符出现的频率，建成一棵Huffman树，出现次数越多的字符的Huffman编码越短，这样可以达到文件的压缩。

几乎所有应用图论的领域都要在相当大的程序上依靠树来解决某些重要问题。树，尤其是生成树可以说是图的骨骼，从来是被重视的一个问题 。

定理　设T是Huffman树，　$p_1 \leq p_2 \leq \ldots \leq p_t$

$v_1, v_2, \ldots v_t$为叶，（1）若v_i，v_j为兄弟，则$l_i = l_j$.

（2）设T+是带权$p_1+p_2, p_3, \ldots, p_t$Huffman树，将$p_1+p_2$相应的叶子生出两个新叶分别带权$p_1$，$p_2$ 则得到带权p_1，$p_2, p_3, \ldots, p_t$Huffman树

求最优树的算法（Huffman算法）

给定实数$p_1, p_2, \ldots, p_t$，且$p_1 \le p_2 \le \cdots \le p_t$。

①连接权为p_1, p_2的两片树叶，得一个分支点，其权为p_1+p_2。

② 在$p_1+p_2, p_3, \ldots, p_t$中选出两个最小的权，连接它们对应的顶点（不一定是树叶），得新分支点及所带的权。

③ 重复②，直到形成$t-1$个分支点、t片树叶为止。

算法举例

例 2.6 求带权 0.2,0.2,0.3,0.3 的 Huffman 树.

解