第三章 树
3.1 树的有关定义

- 给定一个图G=(V,E), 如果它不含任何回路, 我们就叫它是林, 如果G又是连通的, 即这个林只有一个连通支, 就称它是树.
- 定义3.1.1

  一个不含任何回路的连通图称为树, 用T表示. T中的边称为树枝, 度为1的节点称为树叶.

有关度的若干术语

- 孤立点:度为0的顶点
- 悬点:度为1的顶点
  - 悬边:与悬点关联的边
- 奇点:度为奇数的顶点
- 偶点:度为偶数的顶点
- 正则图:各顶点度相同
  - 若度为k,称为k-正则图.
  - 例如:K_n是$(n-1)$-正则图

树的有关定义

- 树的每条边, 都不会属于任何回路. 这样的边叫割边.
- 定义**3.1.2**

  设**e**是**G**的一条边, 若**G'=G-e**比**G**的连通支数增加, 则称**e**是**G**的一条割边.
- 显然, 图**G**删去割边**e=(u,v)**之后, 结点**u, v**分属于不同的分支.

树的有关定义

- 定理3.1.1
  e=(u, v)是割边,当且仅当e不属于G的任何回路.

证明:

必要性。设e=(u, v)是割边, 此时若e=(u, v)属于G的某个回路, 则G’=G-e中仍存在u到v的道路, 故结点u和v属于同一连通支, e不是割边.矛盾.

充分性。设e不属于G的任何回路, 此时若e不是割边, 则G’=G-e与G的连通支数一样. 于是u和v仍属于同一连通支. 故G’中存在道路P(u,v), P(u,v)+e就是G的一个回路.矛盾.

树的有关定义

- 定理3.1.2

  设T是结点数为$n \geq 2$的树，则下列性质等价:

1. T连通且无回路
2. T连通且每条都是割边
3. T连通且有n-1条边
4. T有n-1条边且无回路
5. T的任意两结点间有唯一道路
6. T无回路，但在任两结点间加上一条边后恰有一个回路

T连通且无回路⟶T连通且每条都是割边⟶ T连通且有n-1条边

- $1\rightarrow2$：T无回路，即T的任意边e都不属于回路，由定理3.1.1，e是割边。
- $2\rightarrow3$：对结点数n进行归纳。令$n(T)$, $m(T)$分别表示树T的结点数和边数。当$n=2$时命题成立。设$n\le k$时，$m(T)=n(T)-1$成立。则$n=k+1$时，由于任一边e都是割边，故$G'=G-e$有两个连通支T_1, T_2。由于$n(T_i)\le k$，$i=1,2$，故$m(T_i)=n(T_i)-1$。所以$m(T)=n(T)-1$也成立。

3. T连通且有n-1条边
4. T有n-1条边且无回路

- **3→4**：假定**T**有回路，设**C**是其中一条含有**k(<n)**个结点的初级回路。因为**T**连通，所以**V(T)-V(C)**中一定有结点**u**与**C**上某点**v**相邻，即存在边**(u,v)**∈**E(T)**，依此类推，最终**V(T)-V(C)**中的**n-k**个结点需要**n-k**条边才可能保持**T**连通，但**|E(T)-E(C)|=(n-1)-k<n-k.**矛盾**.**

4. T有n-1条边且无回路
5. T的任意两结点间有唯一道路

- **4→5**：设**u,v**是**T**的任意两结点，先证道路**P(u,v)**的存在性，即证明**T**是连通的。反证法。

  如果**T**不是连通的，则至少有两个连通分支T_1, T_2.由已知**T**中无回路可知，T_1, T_2也没有回路。根据$1 \rightarrow 2 \rightarrow 3$的证明，再由$T_1$和$T_2$是连通的且无回路可得，$m(T_1)=n(T_1)-1$, $m(T_2)=n(T_2)-1$, 则有：

  $m(T)=m(T_1)+m(T_2)=(n(T_1)+n(T2))-2=n-2<n-1$

  与已知$m(T)=n-1$矛盾.

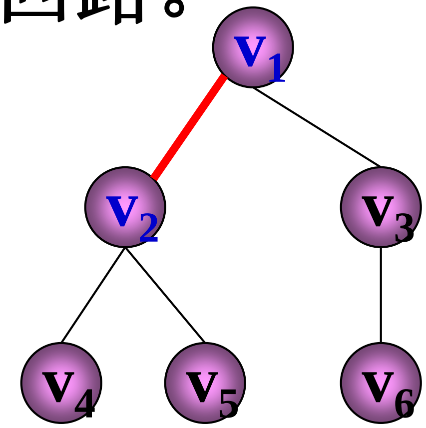
  再证唯一性。若存在两条不同的道路**P(u,v)**, **P'(u,v)**，则其对称差$P(u,v)\oplus P'(u,v)$至少含有一个回路。

注：$G_1\oplus G_2=(V,E)$,其中$V=V_1\cup V_2$,$E=E_1\oplus E_2$;

5. T的任意两结点间有唯一道路
6. T无回路，但在任两结点间加上一条边后恰有一个回路
1. T连通且无回路

- $5 \rightarrow 6$：显然成立
- $6 \rightarrow 1$：只要证明T是连通的。反证法。

假设T不连通，设T_1, T_2为T中的两个连通分支。v_1为T_1中的一个顶点，v_2为T_2中的一个顶点。在T中加边(v_1, v_2)不形成回路。矛盾。



总结

- 树是极小的连通图，减少一条边就不连通
- 树是极大的不含回路的连通图，增加一条边就有回路

树的有关定义

- **定理3.1.3**

  树**T**中一定存在树叶结点**.**

证明**:** 由于**T**是连通图**,**所以任一结点$v_i \in V(T)$, 都有 $d(v_i) \geq 1$. 若无树叶, 则$d(v_i) \geq 2$. 这样

$$n-1=m=\frac{1}{2}\sum d(v_i)\geq n$$

矛盾**.**

树的有关定义

- 定义**3.1.3**

  如果**T**是图**G**的支撑子图**,** 而且又是一棵树**,** 则称**T**是**G**的一棵支撑树**,** 或称生成树**,** 又简称**G**的树

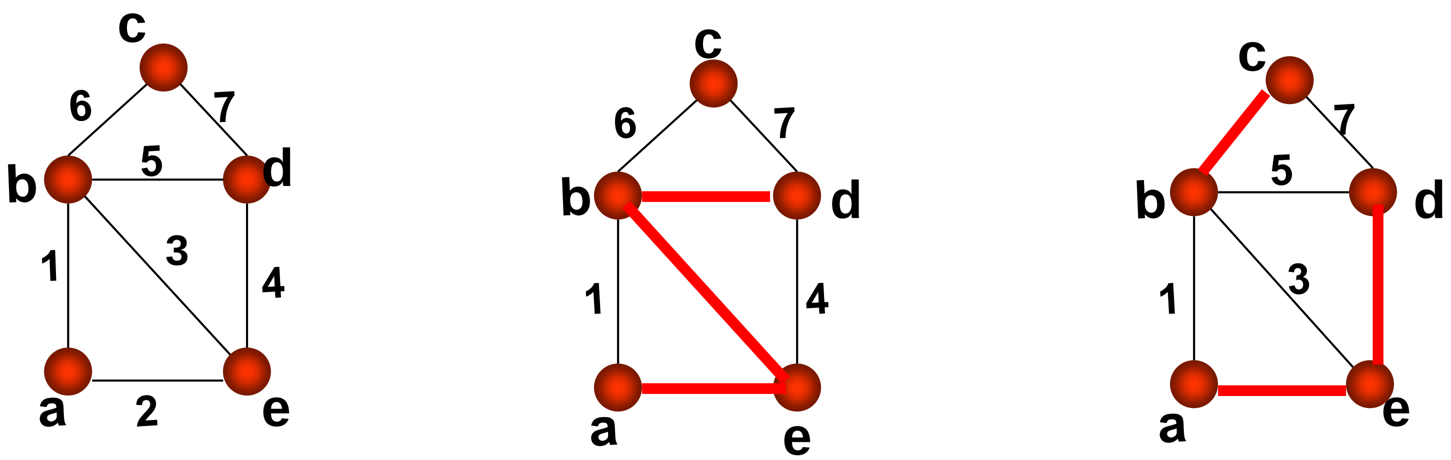
生成树

定理：任何无向连通图G都存在生成树。

证明：如果**G**无回路，则**G**本身就是它的生成树。如**G**有回路，则在回路上任取一条边去掉仍是连通的，如**G**仍有回路，则继续在该回路上去掉一条边，直到图中无回路为止，此时，该图就是**G**的一棵生成树。

一个连通图的生成树可能不唯一。

因为在取定一个回路后，就可以从中去掉任一条边，去掉的边不一样，故可能得到不同的生成树。

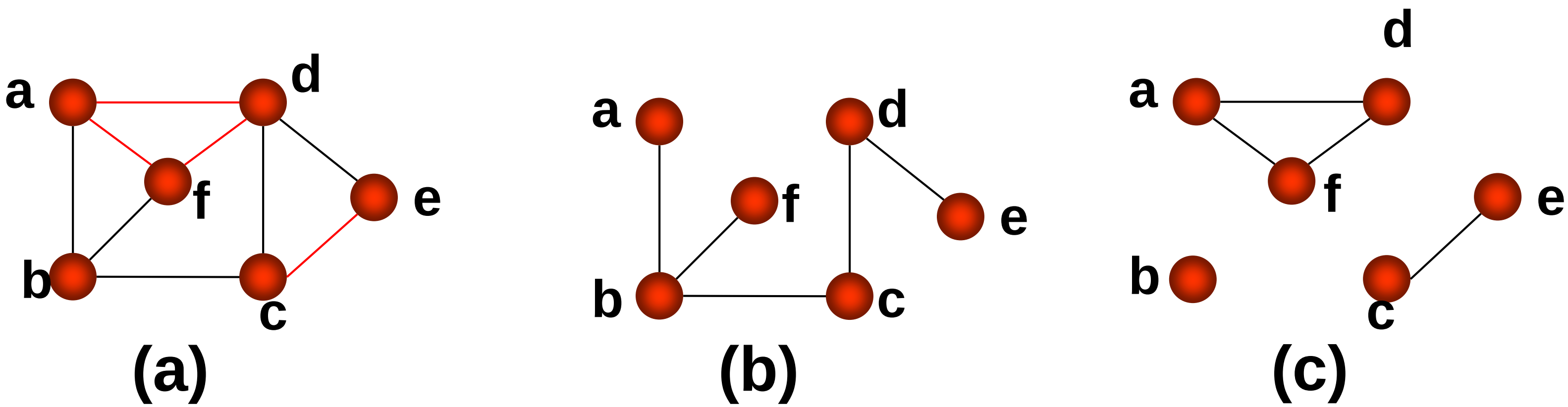


(a) (b) (c)

在上图(a)中，删去边2,3,5，就得到生成树 (b)，

若删去边2，4，6,可得到生成树 (c)。

生成树



给定图G的一棵树T，我们称G-T，即G删去T中各边后的子图为T的余树。

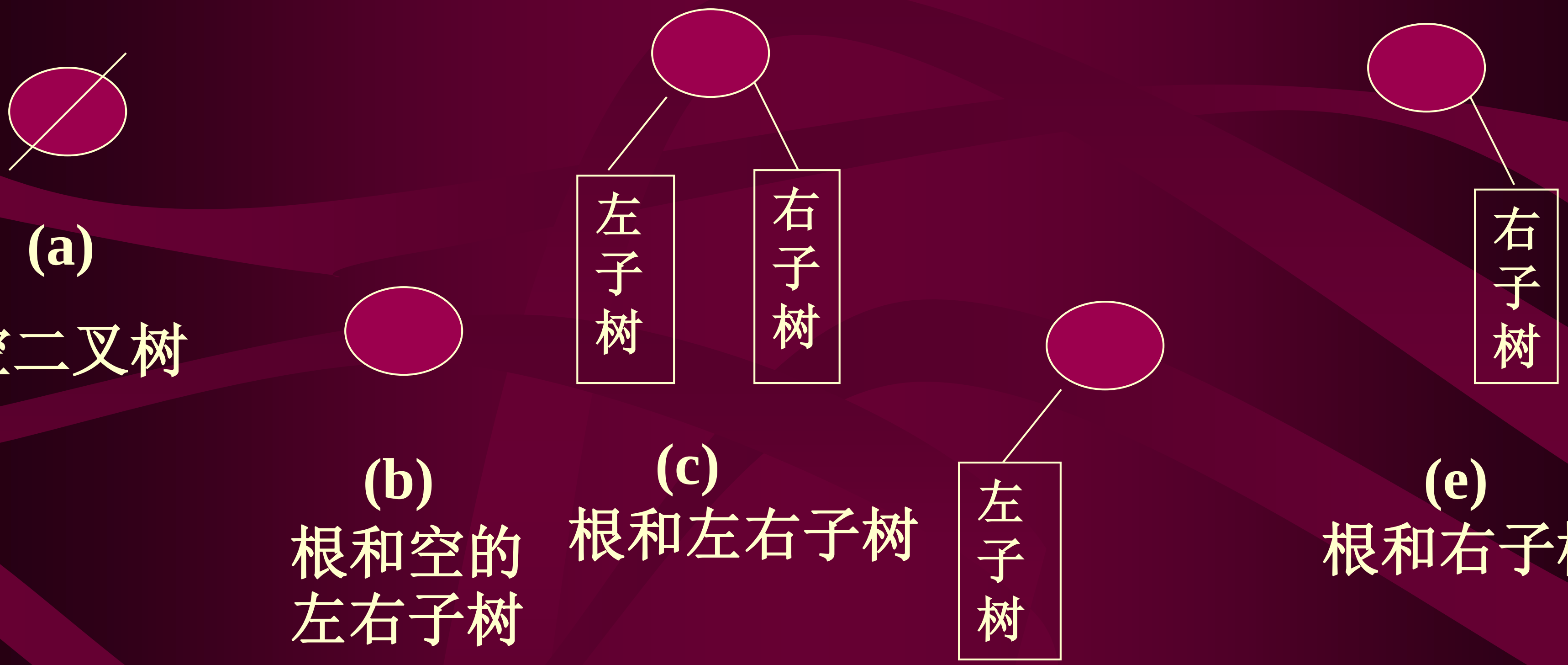
注：余树不一定连通，也不一定无回路，因而余树不一定是树，更不一定是生成树。

二叉树的定义

二叉树（Binary Tree）是结点的有限集合，它或者为空，或者由一个根结点及两棵互不相交的左、右子树构成，而其左、右子树又都是二叉树。

注意：二叉树必须严格区分左右子树。即使只有一棵子树，也要说明它是左子树还是右子树。交换一棵二叉树的左右子树后得到的是另一棵二叉树。

二叉树的基本形态



(a) 空二叉树

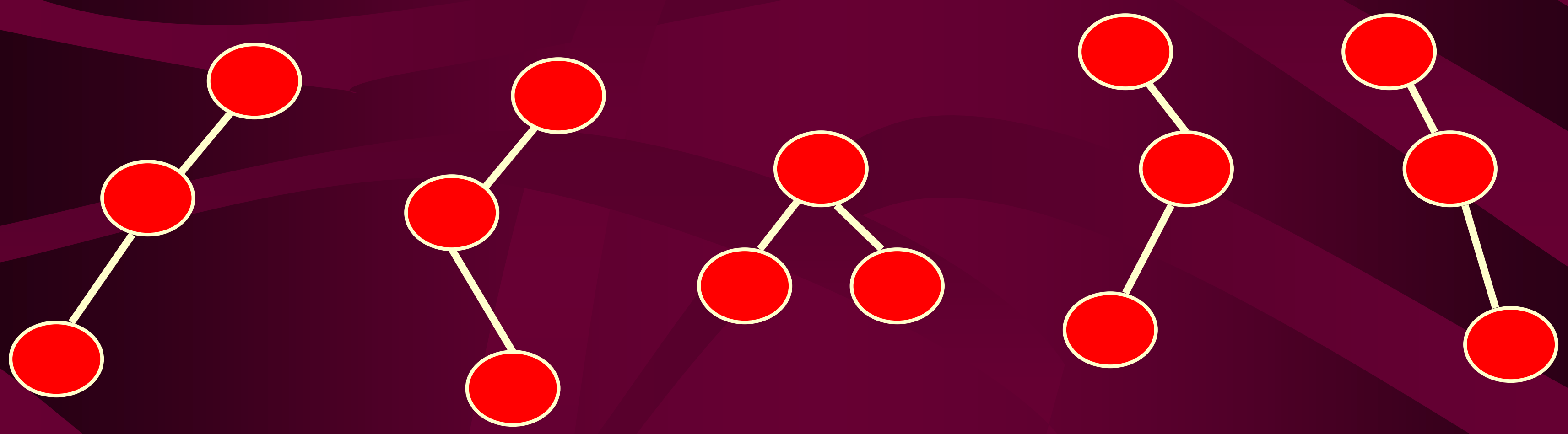
(b) 根和空的左右子树

(c) 根和左右子树

(d) 根和左子树

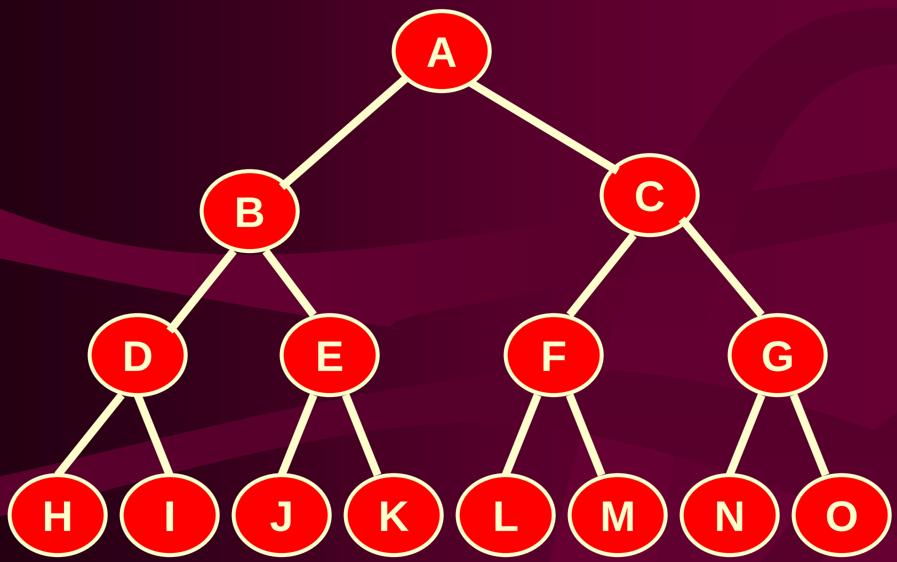
(e) 根和右子树

结点总数为3 的所有二叉树的不同形状

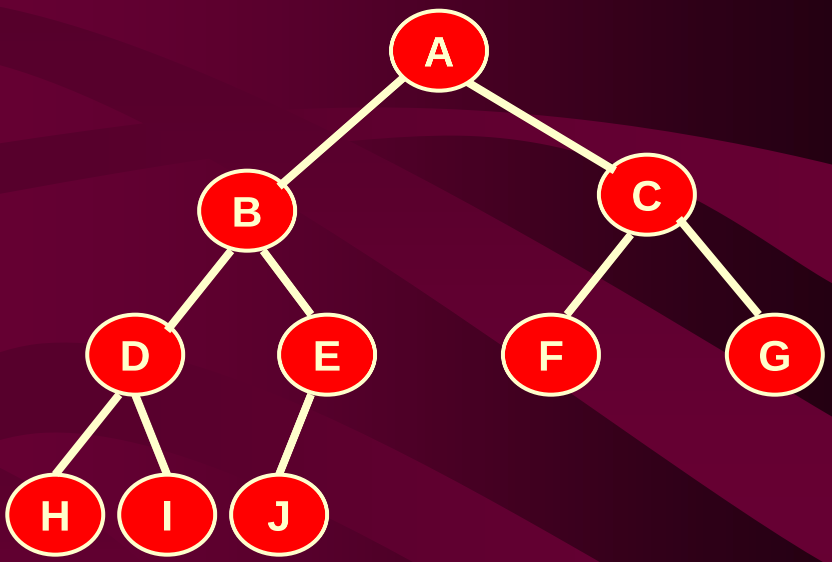


满二叉树

- 一棵高度为k并具有2^k－1个结点的二叉树称为满二叉树。
- 一棵二叉树中任意一层的结点个数都达到了最大值



满二叉树实例



不是满二叉树

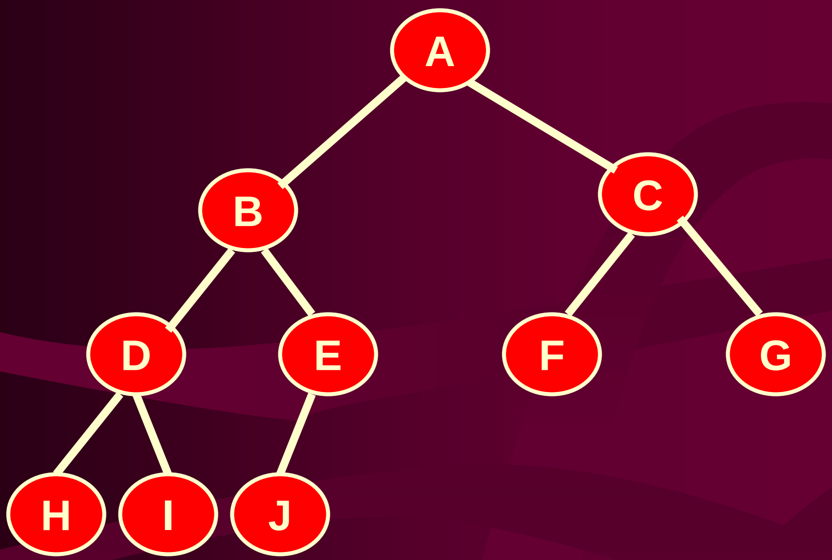
完全二叉树

在满二叉树的最底层自右至左依次(注意：不能跳过任何一个结点)去掉若干个结点得到的二叉树也被称之为完全二叉树。满二叉树一定是完全二叉树，但完全二叉树不一定是满二叉树。

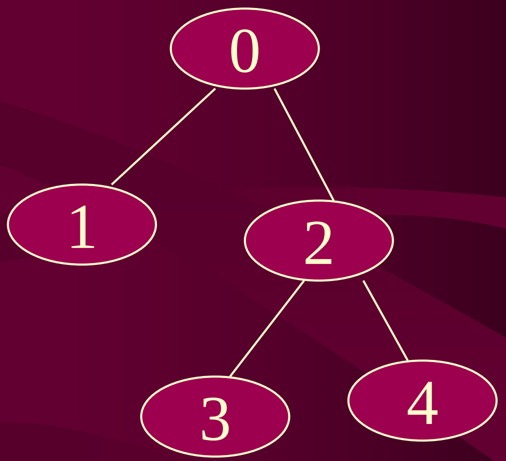
完全二叉树的特点是：

（1）所有的叶结点都出现在最低的两层上。

（2）对任一结点，如果其右子树的高度为k，则其左子树的高度为k或k＋1。



完全二叉树



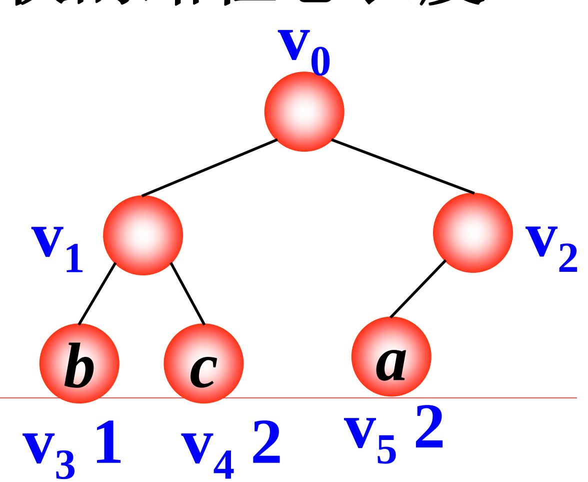
非完全二叉树

树和二叉树

- 二叉树的每个结点至多只有二棵子树
  二叉树的子树有左右之分，次序不能颠倒
  二叉树的第i层至多有$2^{(i-1)}$个结点
  深度为k的二叉树至多有2^k-1个结点(根结点的深度为1)
- 树和二叉树的两个主要差别：
  - 树中结点的最大度数没有限制，而二叉树结点的最大出度数为2
  - 树的结点无左、右之分，而二叉树的结点有左、右之分

- 如果二叉树**T**的每个树叶结点**v_i**都分别赋以一个正实数**w_i**, 则称**T**是赋权二叉树。
- 路径：从树中一个结点到另一个结点之间的边构成这两个结点间的路径
- 从根到树叶**v_i**的路径**$P(v_0, v_i)$**所包含的边数计为该路径的长度**l**，这样二叉树**T**带权的路径总长度为：

$$WPL = \sum_i l_i w_i, \quad v_i\text{是树叶}$$

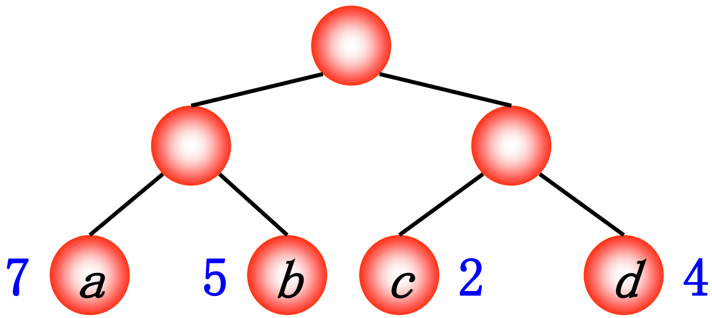


最优二叉树

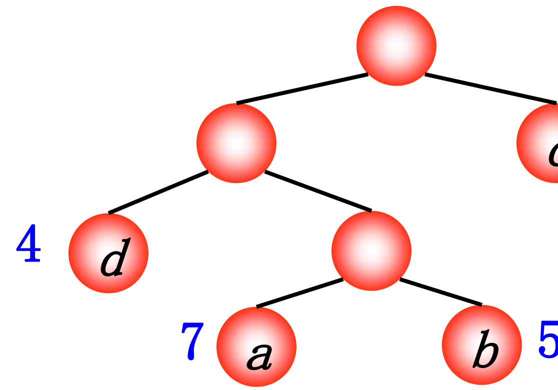
- 如果给定了树叶数目以及它们的权值，可以构造许多不同的赋权二叉树

  在这些赋权二叉树中，必定存在带权路径总长最小的二叉树，这样的树称为最优二叉树

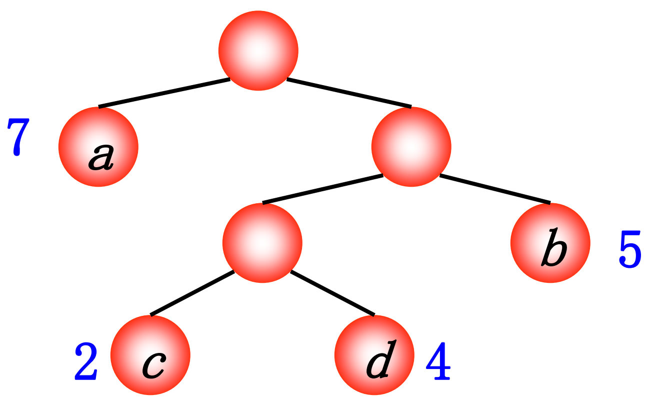
例：有4个结点 a, b, c, d, 权值分别为 7，5，2，4，试构造以此4个结点为叶子结点的二叉树。



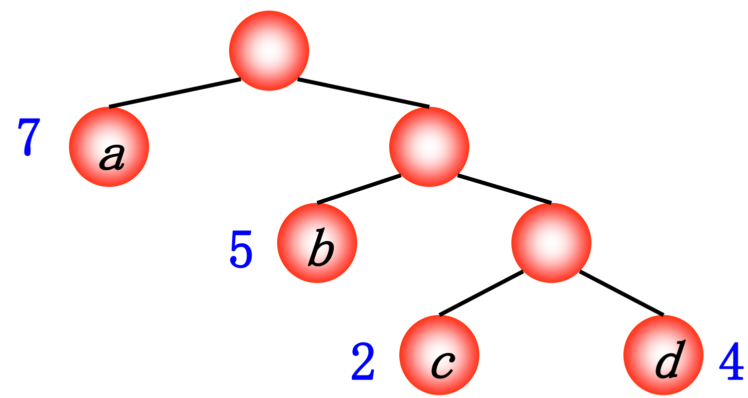
WPL=7×2+5×2+2×2+4×2= 36



WPL=7×3+5×3+2×1+4×2= 46



WPL=7×1+5×2+2×3+4×3= 35



WPL=7×1+5×2+2×3+4×3= 35

最优二叉树

带权路径长(**WPL: Weighted Path Length**)

最小的二叉树

（权值大的结点离根最近）

因为构造这种树的算法是由哈夫曼于 1952 年提出的，

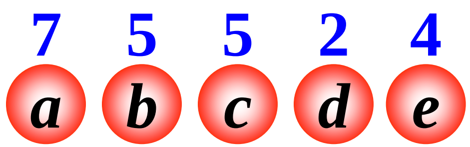
所以被称为哈夫曼树，相应的算法称为哈夫曼算法。

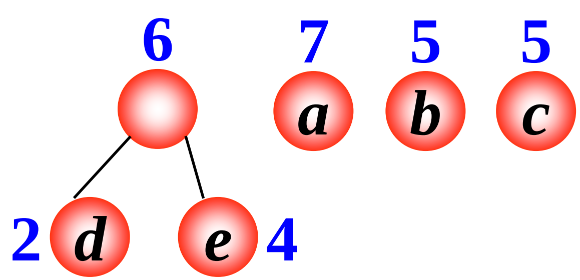
Huffman树

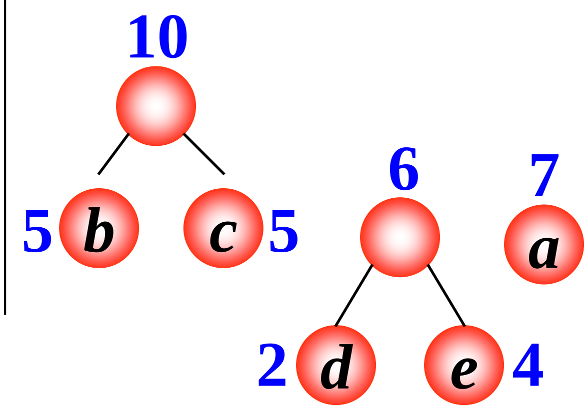
- 哈夫曼算法:

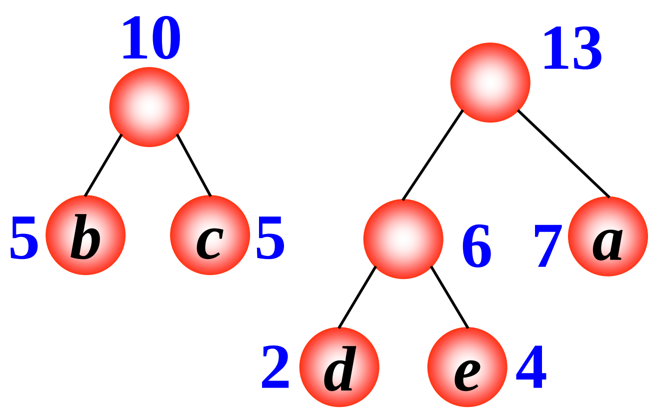
1. 对n(≥2)个权值进行排序,满足$w_{i1} \le w_{i2} \le \ldots \le w_{in}$
2. 计算$w_i = w_{i1} + w_{i2}$作为中间结点v_i的权, v_i的左儿子是v_{i1}, 右儿子是v_{i2}. 在权序列中删去w_{i1}和w_{i2}, 加入w_i, $n \leftarrow n-1$. 若$n=1$,结束. 否则转1.

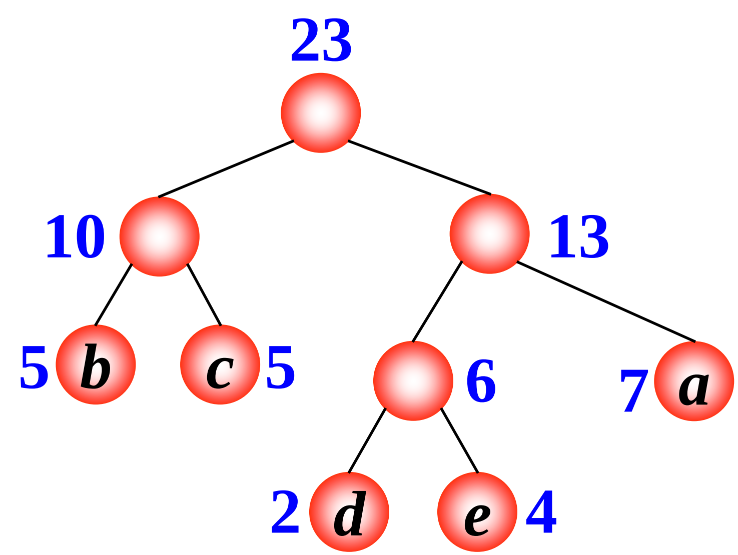
例：有5 个结点 *a*, *b*, *c*, *d*, *e*，权值分别为 7, 5, 5, 2, 4，构造哈夫曼树。











Huffman树

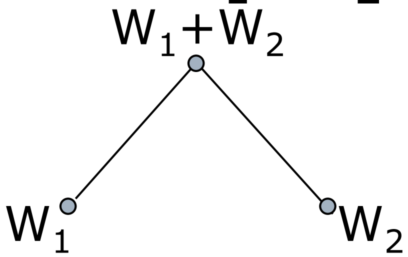
- 定理3.6.1

  由Huffman算法得到的二叉树是最优二叉树。

证明：假定$n \geq 3$, $w_1 \leq w_2 \leq \ldots \leq w_n$，并设T是最优树。

则一定有w_1离根最远，即$l_1 = \max\{l_1, l_2, \ldots, l_n\}$。否则，假设$w_k > w_1$而$l_k > l_1$, 则有$w_k l_1 + w_1 l_k < w_k l_k + w_1 l_1$. 所以将$w_k$和$w_1$对调可得到T’，其满足WPL(T’)<WPL(T), 与T最优矛盾。

同时立即可知，w_1必有兄弟。否则让w_1赋值给该树叶的父亲结点，就可得到路径总长更小的树。由于w_2是序列中次最小的权，故可令w_1的兄弟是w_2。因此分支w_1+w_2可以是最优树T的子图。



最佳前缀码—哈夫曼编码

哈夫曼树的应用很广，哈夫曼编码就是其在电讯通信中的应用之一。在电讯通信业务中，通常用二进制编码来表示字母或其他字符，并用这样的编码来表示字符序列。

例：如果需传送的电文为‘**A B A C C D A**’，它只用到四种字符，用两位二进制编码便可分辨。假设 **A, B, C, D** 的编码分别为 **00**，**01**，**10**，**11**，则上述电文便为‘**00010010101100**’（共 **14** 位），译码员按两位进行分组译码，便可恢复原来的电文。

在编码过程通常要考虑两个问题：

- 数据的最小冗余编码问题
- 译码的惟一性问题

➢ 数据的最小冗余编码问题

在实际应用中，各个字符的出现频度是不尽相同的，有些字符出现的频率较高，有些字符出现的频率较低。我们希望用较短的编码来表示那些出现频率大的字符，用较长的编码来表示出现频率少的字符，从而使得编码序列的总长度最小，使所需总空间量最少。这就是最小冗余编码问题。

在上例中，若假设 A, B, C, D 的编码分别为 0，00，1，01，则电文 ‘A B A C C D A’ 便为 ‘000011010’（共 9 位）。

但此编码存在多义性：可译为 ‘A A A A C C A C A’、‘B B C C D A’、‘A B A C C D A’ 等。

➢ 译码的惟一性问题

要求任一字符的编码都不能是另一字符编码的前缀！

这种编码称为最佳前缀码（其实是非前缀码）。

利用最优二叉树可以很好地解决上述两个问题。

最佳前缀码

定义 设$\alpha=\alpha_1\alpha_2\ldots\alpha_{n-1}\alpha_n$是长度为$n$的符号串，$\alpha_1\alpha_2\ldots\alpha_k$称作$\alpha$的长度为$k$的前缀，$k=1,2,\ldots,n-1$.

若非空字符串$\beta_1, \beta_2, \ldots, \beta_m$中任何两个互不为前缀，则称$\{\beta_1, \beta_2, \ldots, \beta_m\}$为前缀码

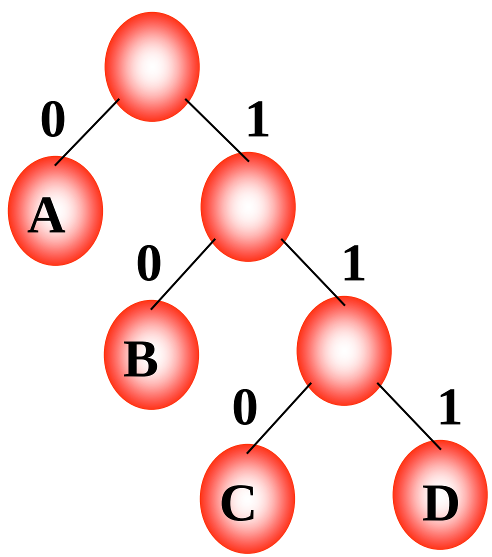
只出现两个符号(如0与1)的前缀码称作二元（二进制）前缀码

例如 { 0, 10, 110, 1111 }, { 10, 01, 001, 110 }是二元前缀码
{ 0, 10, 010, 1010 } 不是前缀码

用二叉树设计二进制前缀码

以电文中的字符作为叶子结点构造二叉树。然后将二叉树中结点引向其左孩子的分支标‘0’，引向其右孩子的分支标‘1’；每个字符的编码即为从根到每个叶子的路径上得到的 0, 1 序列。如此得到的即为二进制前缀码。

例：



编码：A：0

B：10

C：110

D：111

用哈夫曼树设计总长最短的二进制前缀编码

假设各个字符在电文中出现的次数（或频率）为 w_i ，其编码长度为 l_i，电文中只有 n 种字符，则电文编码总长为：

$$\mathbf{WPL} = \sum_{i=1}^{n} w_i l_i$$

从根到叶子的路径长度

叶子结点的权

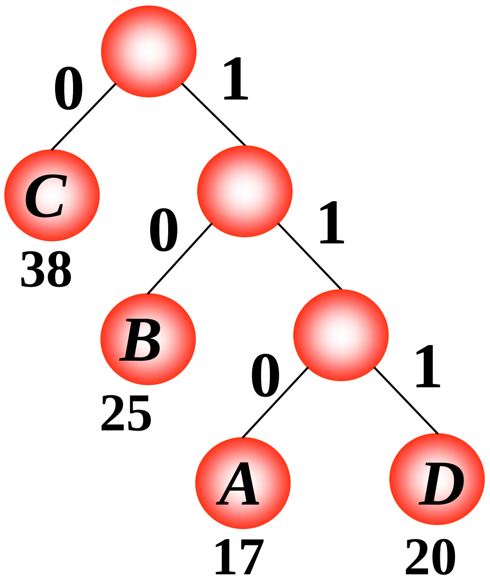
设计电文总长最短的编码 ⟺ 设计哈夫曼树（以 n 种字符出现的频率作权）

由哈夫曼树得到的二进制前缀码称为哈夫曼编码。

例：设 *A*, *B*, *C*, *D* 的频率（即权值）分别为 17%, 25%, 38%, 20%，

试设计哈夫曼编码（最佳前缀码）。

解：



编码： *C*：0

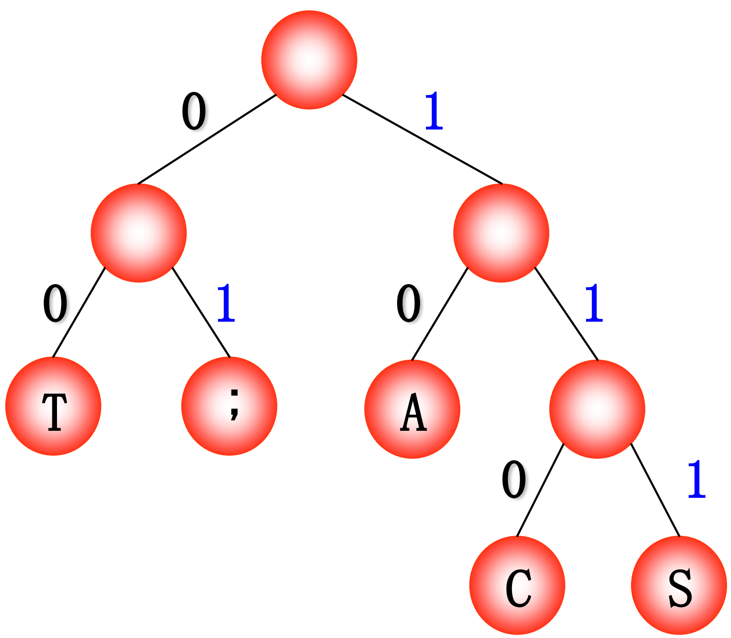
B：10

A：110

D：111

译码

从哈夫曼树根开始，对待译码电文逐位取码。若编码是“0”，则向左走；若编码是“1”，则向右走，一旦到达叶子结点，则译出一个字符；再重新从根出发，直到电文结束。



电文为 “1101000”

译文只能是“CAT”

3.7 最短树(最小生成树)

- 在赋权连通图中，计算该图总长最小的支撑树，即求最短树
- 两种算法
  - **Kruskal**算法
  - **Prim**算法

3.7.1 Kruskal算法

- 基本思想：

  不断往T中加入当前的最短边e,如果此时会构成回路，那么它一定是这个回路中的最长边，删之。直至最后达到n-1条边为止。这时T中不包含任何回路，因此是树。

依据树的性质：

若T有n-1条边且无回路，则T是树

3.7.1 Kruskal算法

- **Kruskal算法的描述如下:**

T←Φ。

当|T| < n – 1且E≠Φ时,

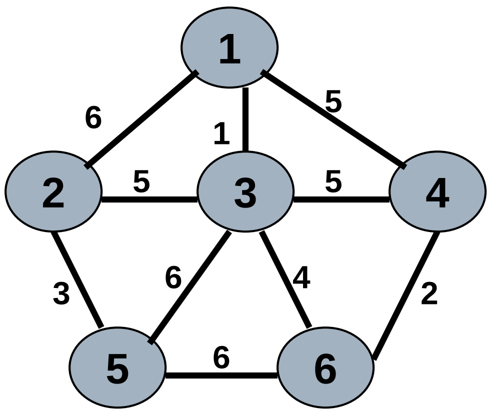
begin

1. **e←E中最短边.**
2. **E←E-e.**
3. **若T+e无回路,则T←T+e.**

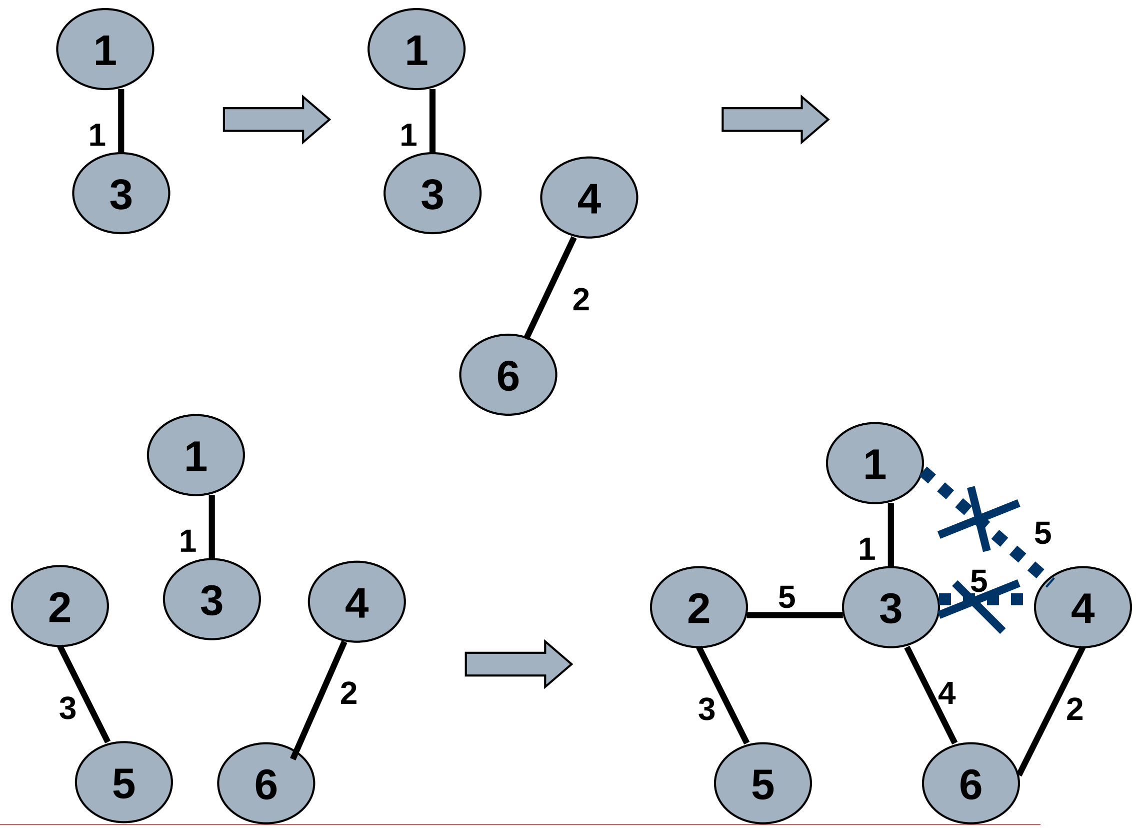
end

若|T|<n-1打印”非连通”,否则输出最短树

Kruskal算法实例



图G



最短树

- 定理 **3.7.1**

  T=(V, E’)是赋权连通图**G=(V, E)**的最短树**,** 当且仅当对任意的余树边**e**∈**E-E’, ,** 回路C^e($C^e \subseteq$ E’+e)**,** 满足其边权**w(e)≥w(a), a**∈C^e**(a≠e)**.

最短树

- 定理 **3.7.2**
  Kruskal算法的计算复杂性是
  $$O(m+p*\log m)$$
  其中**m**为边数，**p**是迭代次数.

3.7.2 Prim算法

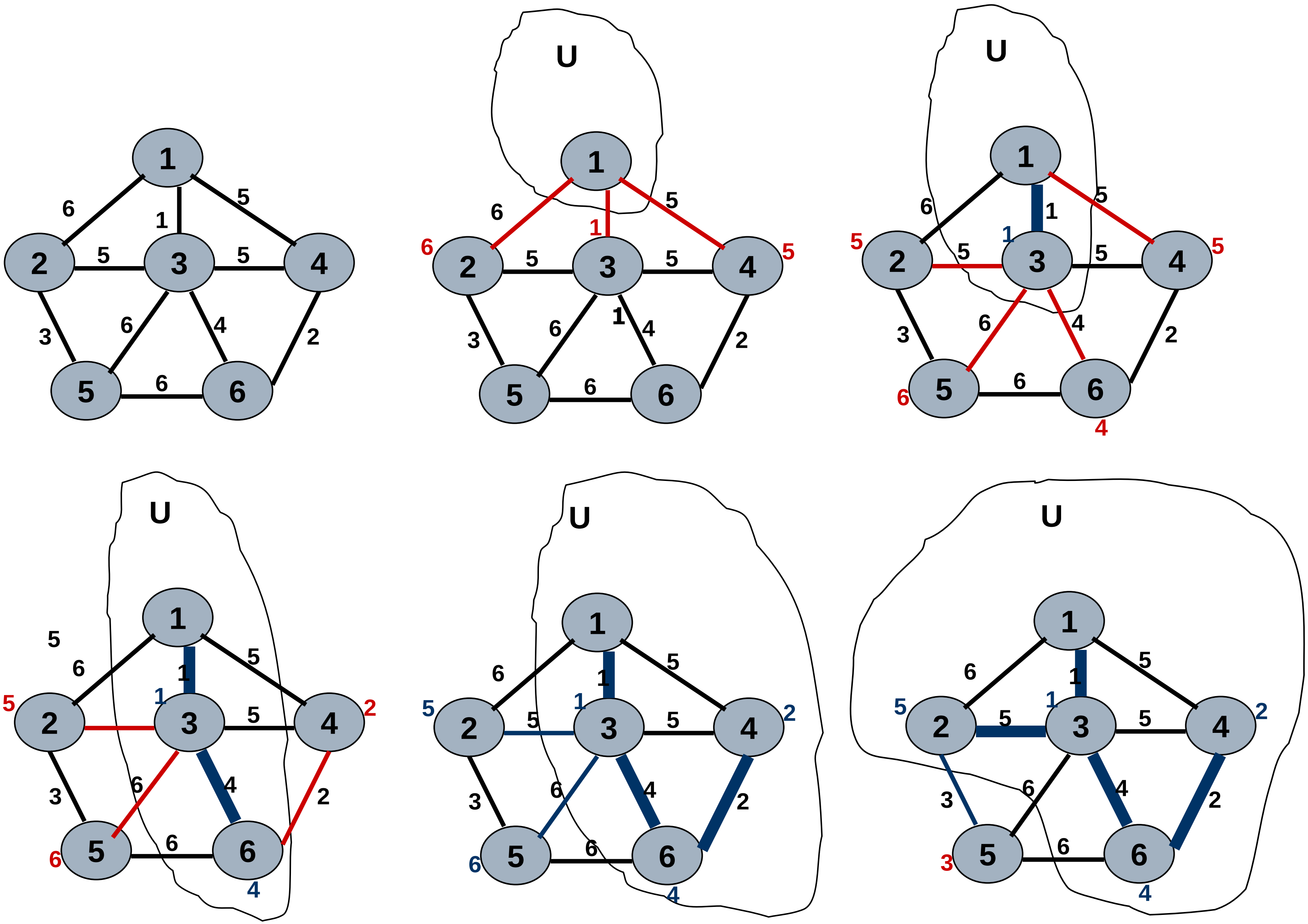
- **Prim算法的基本思想是：**

  首先任选一结点v_0构成集合**U**, 然然后不断在**V-U**中选一条到**U**中最短的边**(u,v)**（其中$u \in V-U$, $v \in U$）进入树**T**, 并且$U \leftarrow U+u$, 直到**U=V**

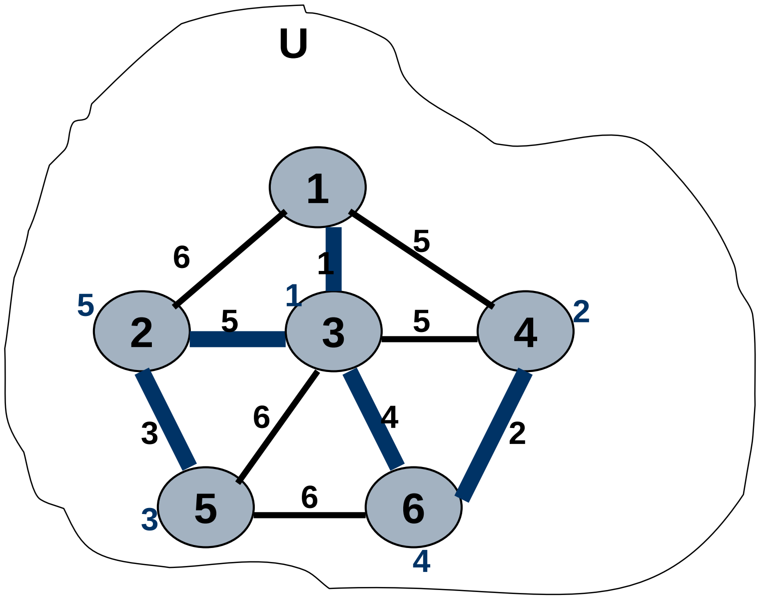
最短树

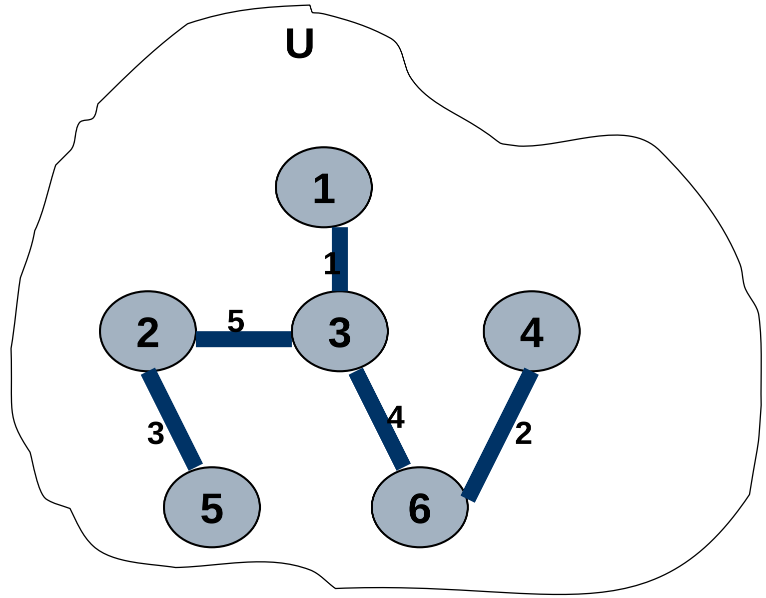
- Prim算法的描述如下:

1. $t \leftarrow v_1$, $T \leftarrow \varnothing$, $U \leftarrow \{t\}$
2. While $U \neq V$ do
   begin
3. $w(t, u)=\min_{v \in V-U}\{w(t, v)\}$
4. $T \leftarrow T+e(t, u)$
5. $U \leftarrow U+u$
6. For $v \in V-U$ do
   $w(t, v) \leftarrow \min\{w(t, v) | t \in U\}$
   end


U
U
U
U
U

Prim算法实例





即：选择 **V - U** 中结点直接至 **U** 中结点的最短边（记为（**u, v**））进入**T**，并将结点**u**加入**U**，直至**U=V**

最短树

- 定理 **3.7.3**

  设**V’**是赋权连通图**G=(V,E)**的结点真子集，**e**是二端点分跨在**V’**和**V-V’**的最短边，则**G**中一定存在包含**e**的最短树**T**.

最短树

- 定理 **3.7.4**

  Prim算法的结果是**:** 得到赋权连通图**G**的一棵最短树.

作业

习题三 P66

- 1，2
- 14，16

考核范围

- 《数理逻辑与集合论》

第 1 章；第 2 章；第 4 章；

第 5 章 (存在型前束范式不讲)

归结法不考核

- 《图论与代数结构》

第 1 章：(**1.2.4**，**1.2.5**，**1.2.6**不讲)

第 2 章： **2.1, 2.3, 2.4**

第 3 章： **3.1, 3.6, 3.7**