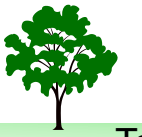
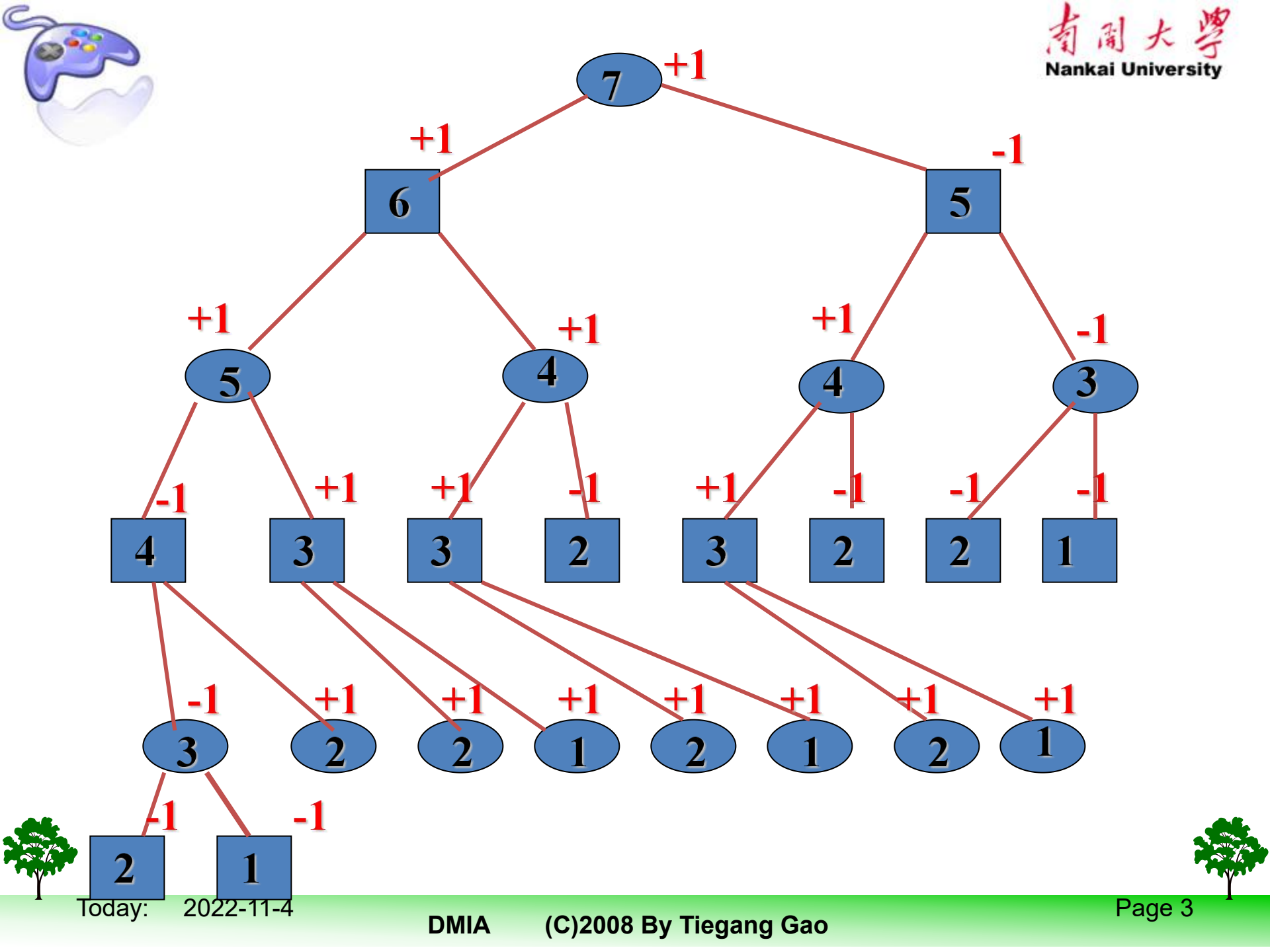
离散数学第三部分之三

最优树、生成树与应 用



搜索树

例：有n根火柴，甲乙两个依次可以从中任意取走1根或2根，但不能不取，取走最后一根火柴者为胜方，试讨论取胜策略。为了便于理解，不妨假设n=7。


7
+1
+1
6
-1
5
+1
5
+1
4
+1
4
-1
3
-1
4
+1
3
+1
3
-1
2
+1
3
-1
2
-1
2
-1
1
-1
3
+1
2
+1
2
+1
1
+1
2
+1
1
+1
2
+1
1
-1
2
-1
1

在树形图中，圆圈中数字7表示轮到甲方取时还有7根，方框中数字6表示表示轮到乙方取时还有6根，依次类推。显然，一旦最后出现方框（1或2）状态时乙方胜。反之，若最后出现圆圈（1或2）状态时甲方胜。

若甲方胜时得分+1，乙方胜时甲方得分-1，无凝轮到甲方取时一定选择能使他进入得分+1状态。同理轮到乙方取时一定选择能使甲方进入得分-1状态。

这样从树叶开始逐层向上，就可以算术出甲方胜负情况。

不难看出，n=1或n=2时，先取者为胜，n=3时，先取者为败。

如果n=3k（k正整数），假定先取者拿走a根（a=1或2），则后者只要取走b 根，（满足a+b=3），使余下火柴为3（k-1）根，那么先取者必败。如果n不等于3k，则先取者取a根，使n-a=3k，那么先取者必胜。

例： 求含n个元素的集合的幂集

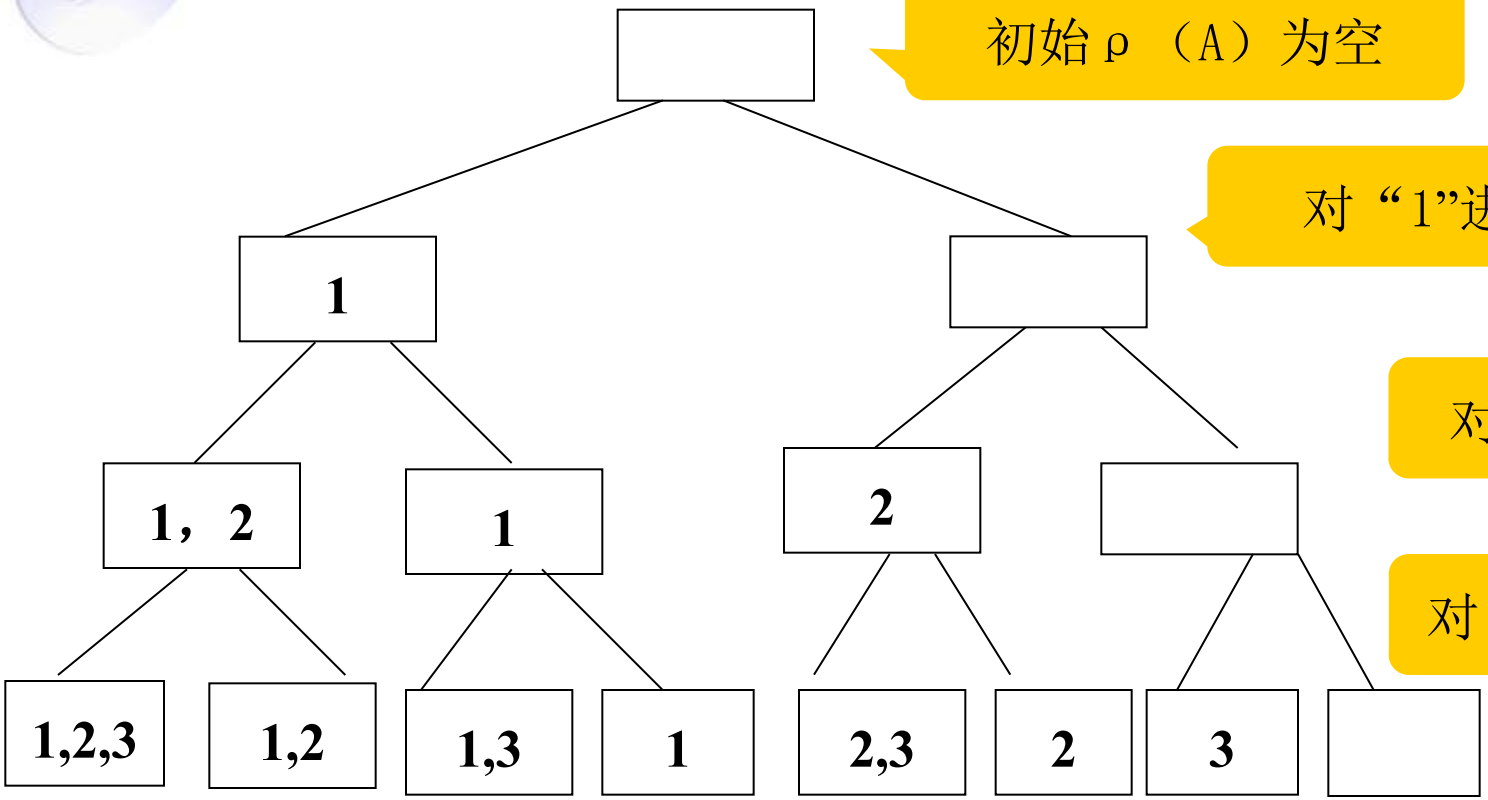
如A={1,2,3}，则A的幂集

ρ（A）={{1,2,3},{1,2},{1,3},{1},{2,3},{2},{3},Φ}

从集合A中每个元素的角度看：集合中的每个元素只有两种状态

- 属于幂集中的元素集
- 不属于幂集中的元素集

求ρ(A)的元素的过程即为依次为集合A中元素进行“取”或“舍”的过程，可构造如下的一棵二叉树：



幂集元素在生成过程中的状态树

约定： 左分支“取”

右分支“舍”

■ 赫夫曼（Huffman）树及其应用

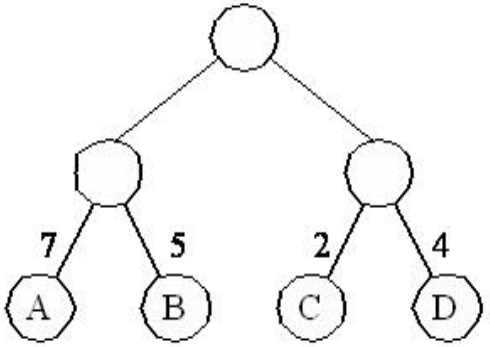
路径和路径长度

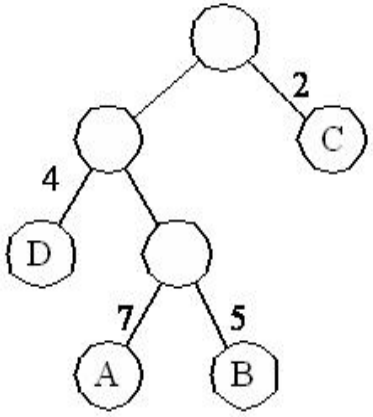
路径：是指从一个结点到另一个结点之间的分支序列

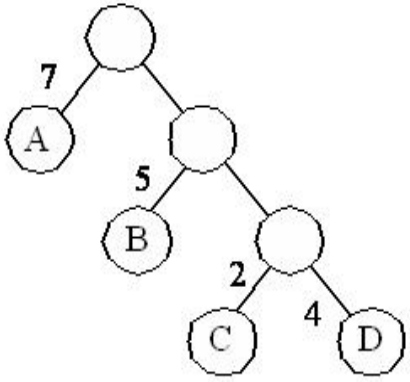
路径长度：是指从一个结点到另一个结点所经过的分支数目。

2. 结点的权和带权路径长度

在实际的应用中，人们常常给树的每个结点赋予一个具有某种实际意义的实数，我们称该实数为这个结点的权。在树形结构中，我们把从树根到某一结点的路径长度与该结点的权的乘积，叫做该**结点的带权路径长度。**





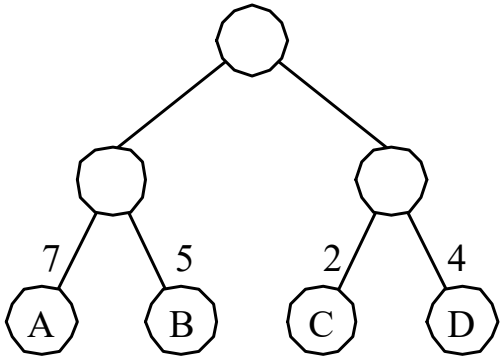


3. 树的带权路径长度（Weighted Path Length of Tree）

树的带权路径长度为树中所有叶子结点的带权路径长度之和，通常记为：

$$WPL = \sum_{i=1}^{n} W_i \times 1_i$$

其中n为叶子结点的个数，w_i为第i个叶子结点的权值，l_i为第i个叶子结点的路径长度。



图a

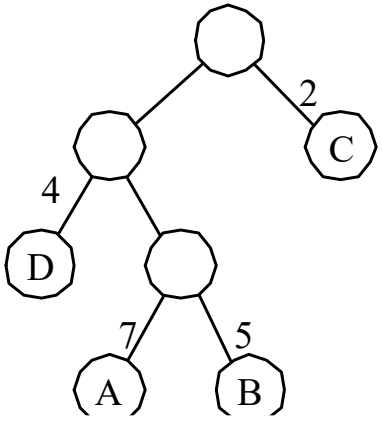
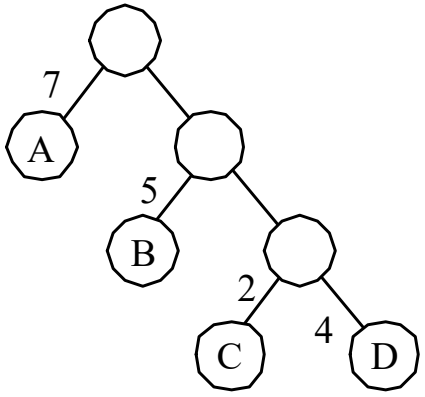


图 b



图c

WPL(a)= 7×2＋5×2＋2×2＋4×2＝36

WPL(b)= 4×2＋7×3＋5×3＋2×1＝46

WPL(c)= 7×1＋5×2＋2×3＋4×3＝35

4. 哈夫曼树

又称**最优二叉树**，它是n个带权叶子结点构成的所有二叉树中，带权路径长度WPL最小的二叉树。

■ 构造最优二叉树

具体构造哈夫曼算法的步骤如下：

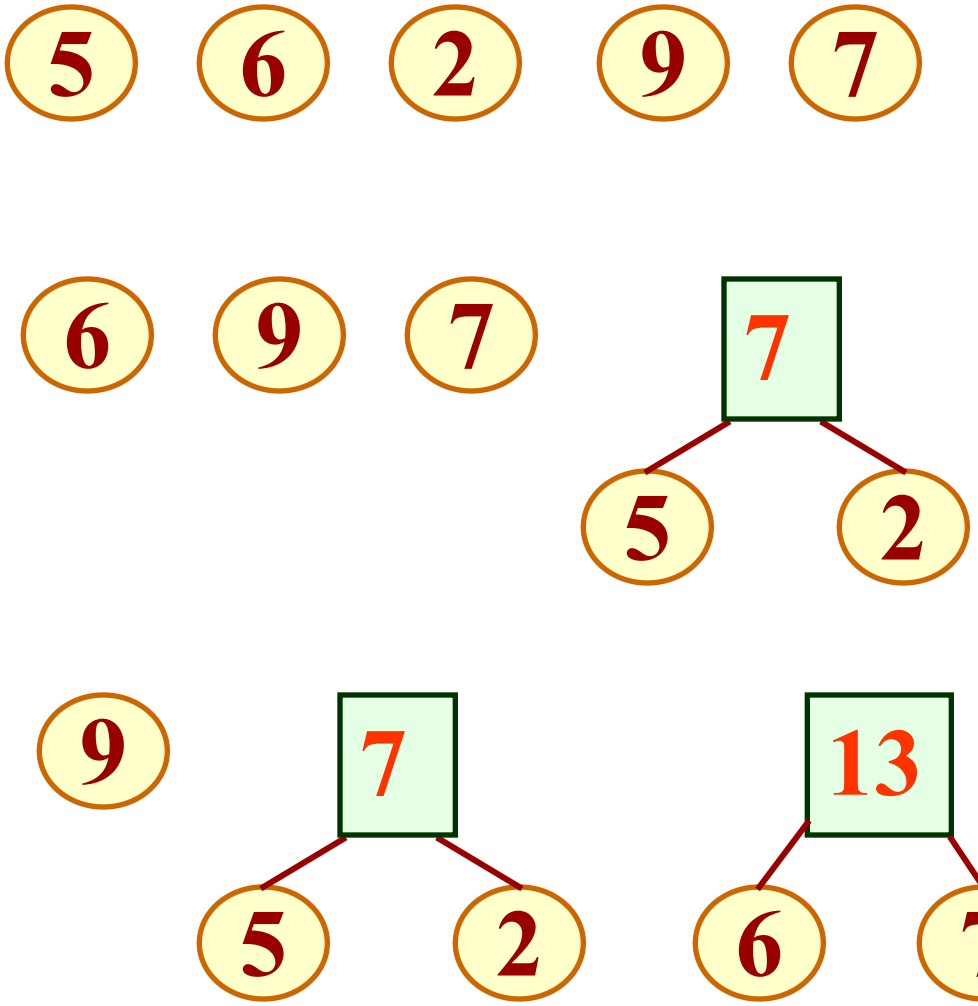
（1） 根据给定的 n 个权值$\{w_1, w_2, \ldots, w_n\}$构成 n 棵二叉树的集合 F =$\{T1, T2, \ldots, Tn\}$，其中每棵二叉树$Ti(1 \le i \le n)$ 中只有一个带权为w_i的根结点，其左右子树均空；

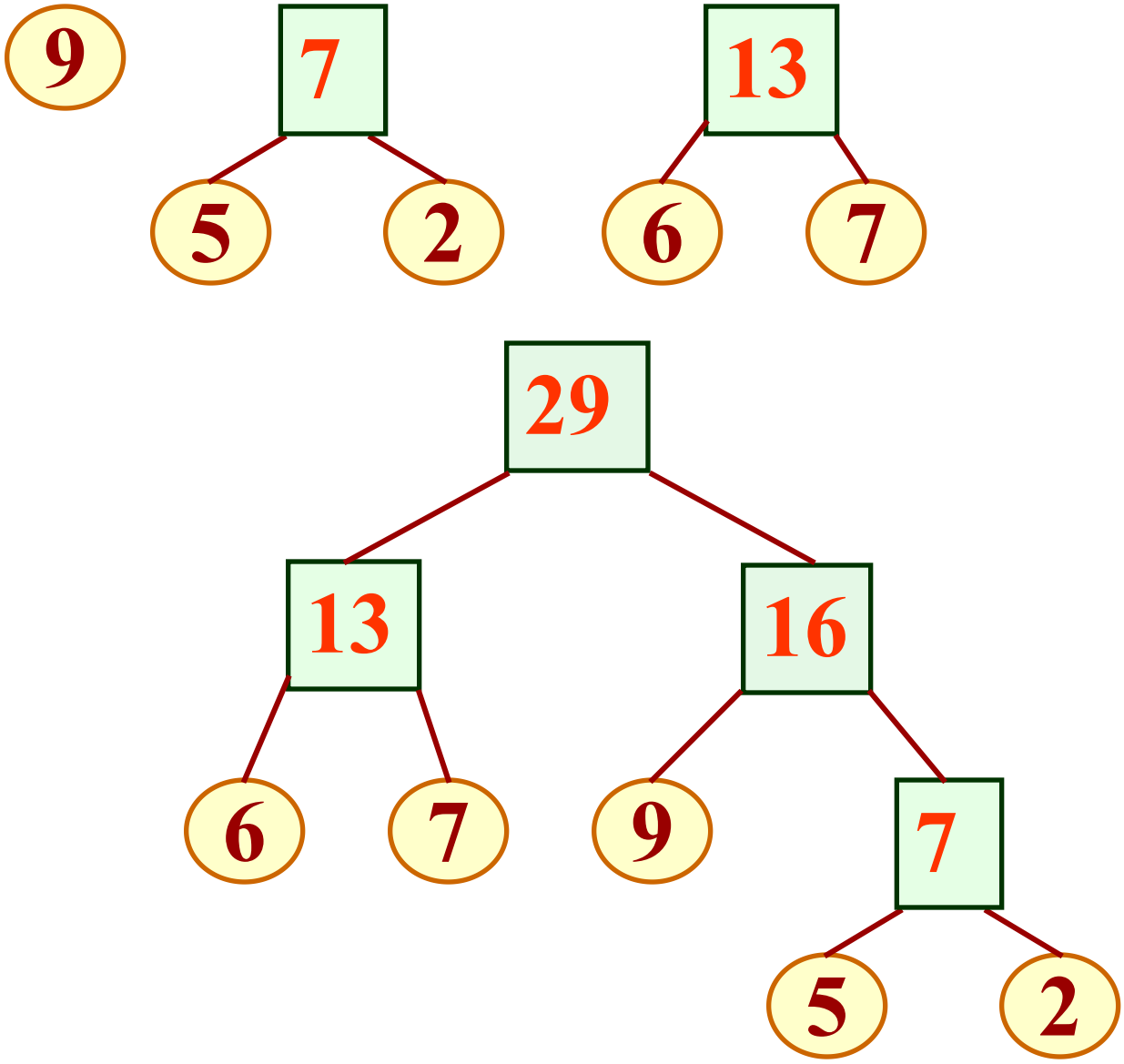
（2） 在 F 中选取两棵根结点的权值最小的树作为左右子树，构造一棵新的二叉树，且置新的二叉树的根结点的权值为其左、右子树上根结点的权值之和。

（3）在F中删除这两棵树, 同时将新得到的二叉树加入F 中。

重复(2)和(3)，直到 F 只含一棵树为止。这棵树便是所求的赫夫曼树。

例如：已知权值 W={ 5, 6, 2, 9, 7 }




9
7
5
2
13
6
7
29
13
16
6
7
9
7
5
2

■ 赫夫曼编码

一般编码方式：

1）等长编码

例：假设传送的电文为英文序列‘ABACCDA’

A	B	C	D
00	01	10	11

则电文编码为“00010010101100”，总长14，

译码时：两位1分

2）不等长编码

一般地，在英文字母a—z中，e的使用频率比q,z要大得多,使用频率高的用短码,使用频率低的用长码　编码不等长，但大部分情况下电文长度会减少

使用频率：　3　1　2　1

A　B　C　D

0　00　1　10

译码：AAAA 或 BB 或 ABA

电文编码“000011010”，总长9

原因：A(0)为B(00)的前缀

■ 前缀编码

指的是，**任何一个字符的编码都不是同一字符集中另一个字符的编码的前缀。**

如A B C D 四个字符的使用率由高到低，编码为

A --- 0　B --- 10　C --- 110　D --- 111

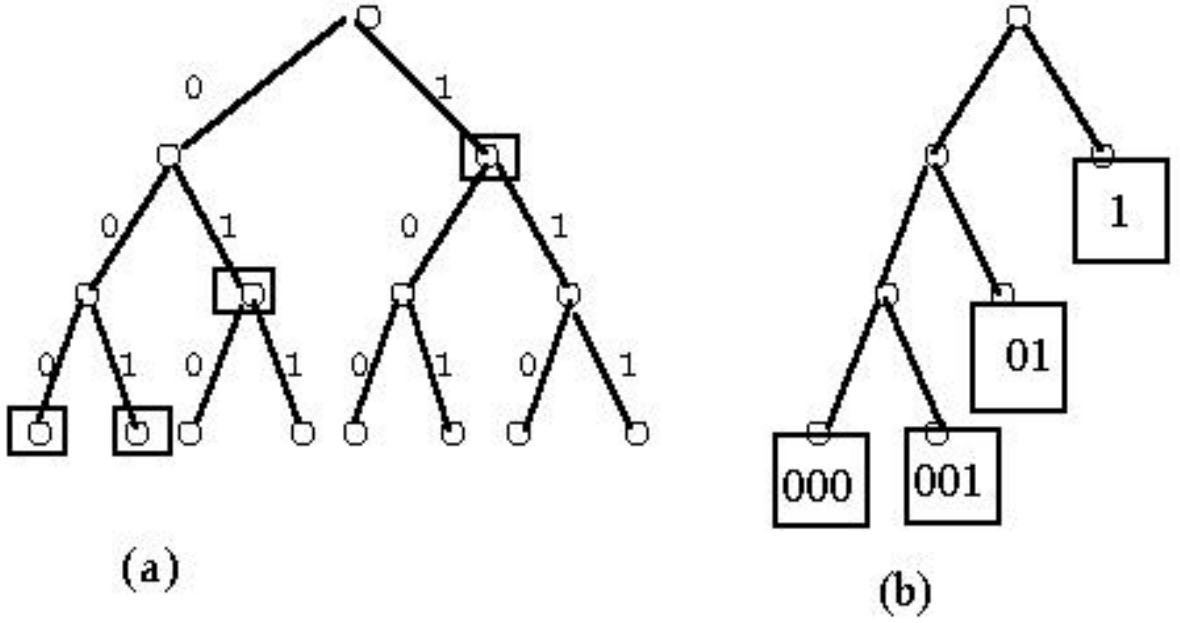
利用赫夫曼树可以构造一种不等长的二进制编码，并且构造所得的赫夫曼编码是一种最优前缀编码，即使所传电文的总长度最短。

定理 任何一棵二叉树的树叶可对应一个前缀码。

证明 给定一棵二叉树，从每一个分枝点引出两条边，对左侧边标以0，对右侧边标以1，则每片树叶可以标定一个0和1的序列，它是由树根到这片树叶的通路上各边标号所组成的序列，显然，没有一片树叶的标定序列是另一片树叶的标定序列的前缀，因此，任何一棵二叉树的树叶可对应一个前缀码。

定理 任何一个前缀码都对应一棵二叉树

证明 设给定一个前缀码，h表示前缀码中最长序列的长度。我们画出一棵高度为h的正则二叉树，并给每一分枝点射出的两条边标以0和1，这样，每个结点可以标定一个二进制序列，它是从树根到该结点通路上各边的标号所确定，因此，对长度不超过h的每一二进制序列必对应一个结点。对应于前缀码中的每一序列的结点，给予一个标记，并将标记结点的所有后裔和射出的边全部删去，这样得到一棵二叉树，再删去其中未标记的树叶，得到一棵新的二叉树，它的树叶就对应给定的前缀码。



图(a)是高度为3的正则二叉树，对应前缀码中序列的结点用方框标记

图(b)中所对应的前缀码{000,001，01，1}，可对任意二进制序列进行译码。设有二进制序列

0001001101110100 1

可译为**000,1,001,1,01,1，1,01,001。**

例. 假设字符A, B, C, D, E的出现概率为42, 28, 15, 10, 5　求电文的总长度最短的一种编码？

首先构建赫夫曼树（根据概率）

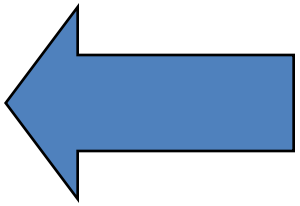
A --- 1

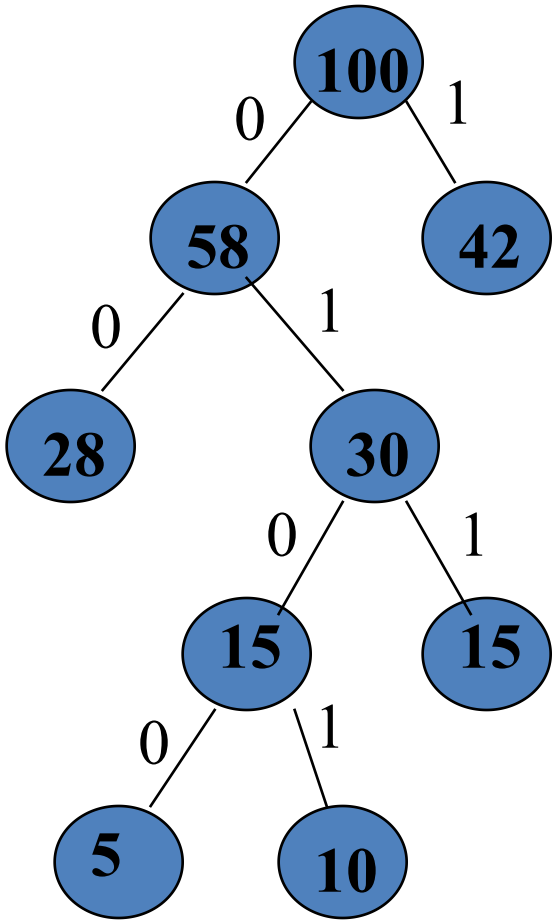
B --- 00

C --- 011

D --- 0101

E --- 0100





Huffman树的应用：寻求最佳判断过程

根据出现的频率
决定比较的次数

例：编制一个将百分制转换成五级分制的程序

0～59 ——————————— bad

60～69 ——————————— pass

70～79 ——————————— general

80～89 ——————————— good

90～100 ——————————— excellent

.....5%

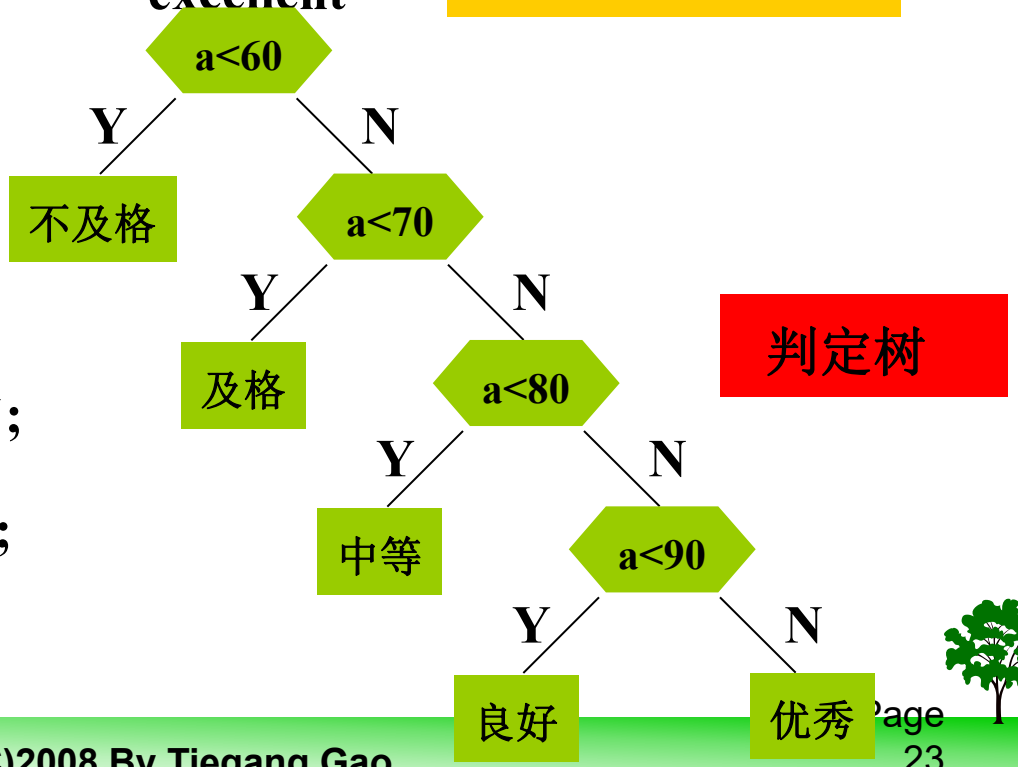
.....15%

.....40%

.....30%

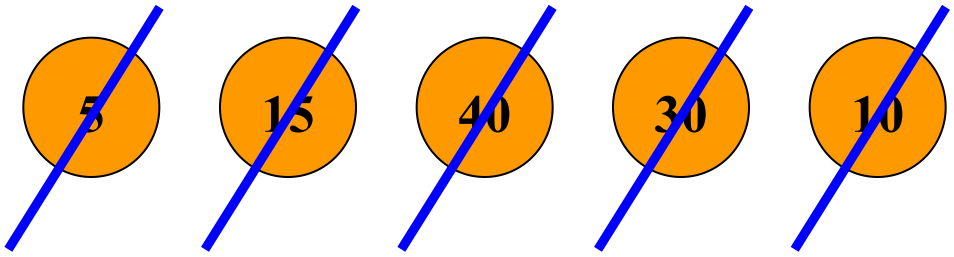
.....10%

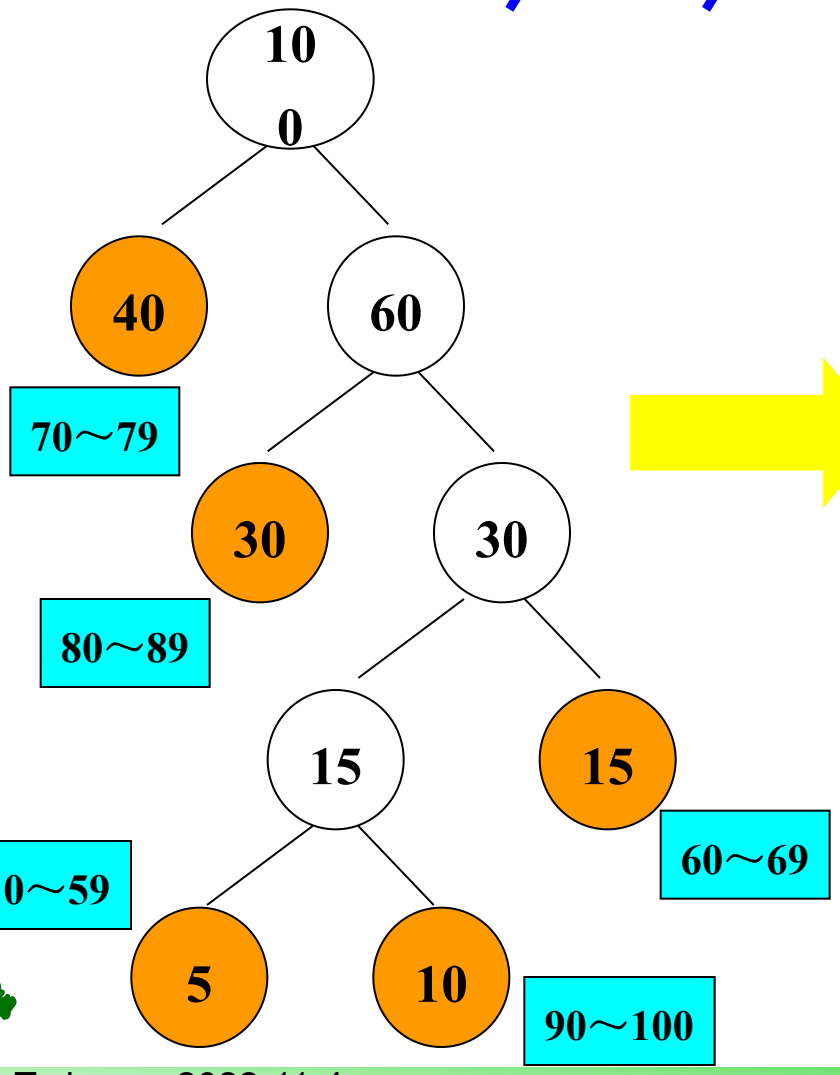
```
if (a<60) b="bad";
else if (a<70) b="pass";
    else if (a<80) b= "general";
        else if (a<90) b="good";
            else b="excellent";
```

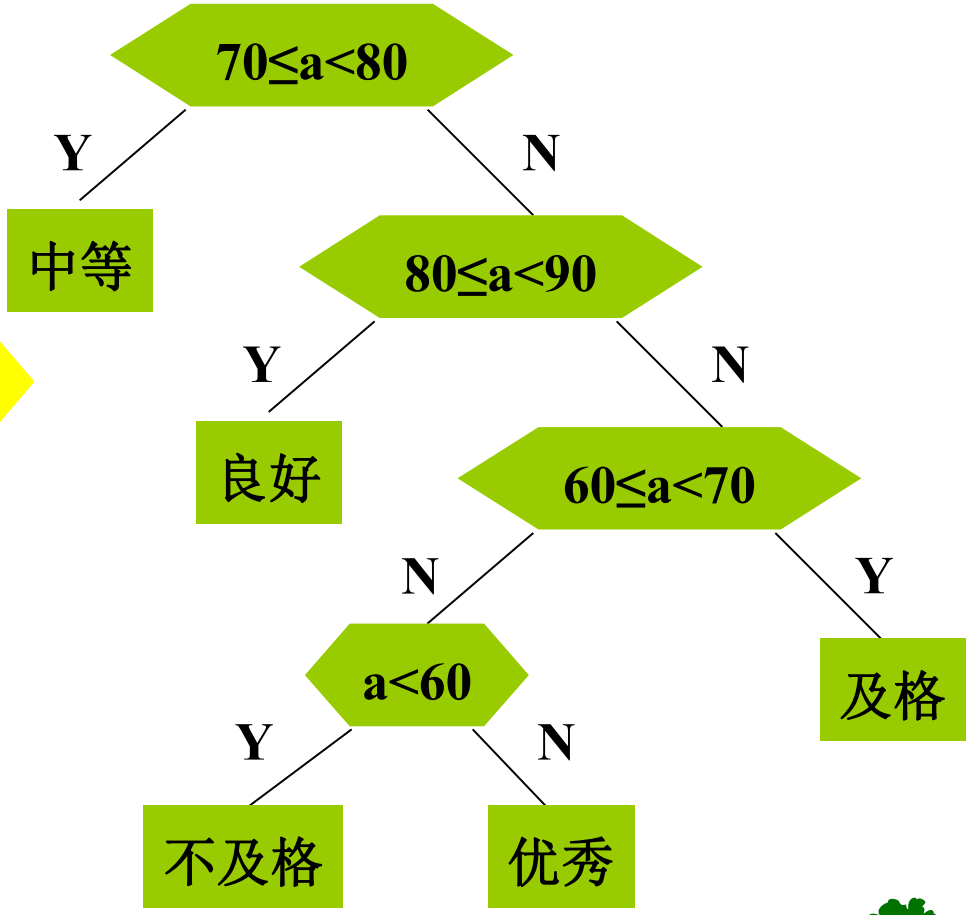


判定树

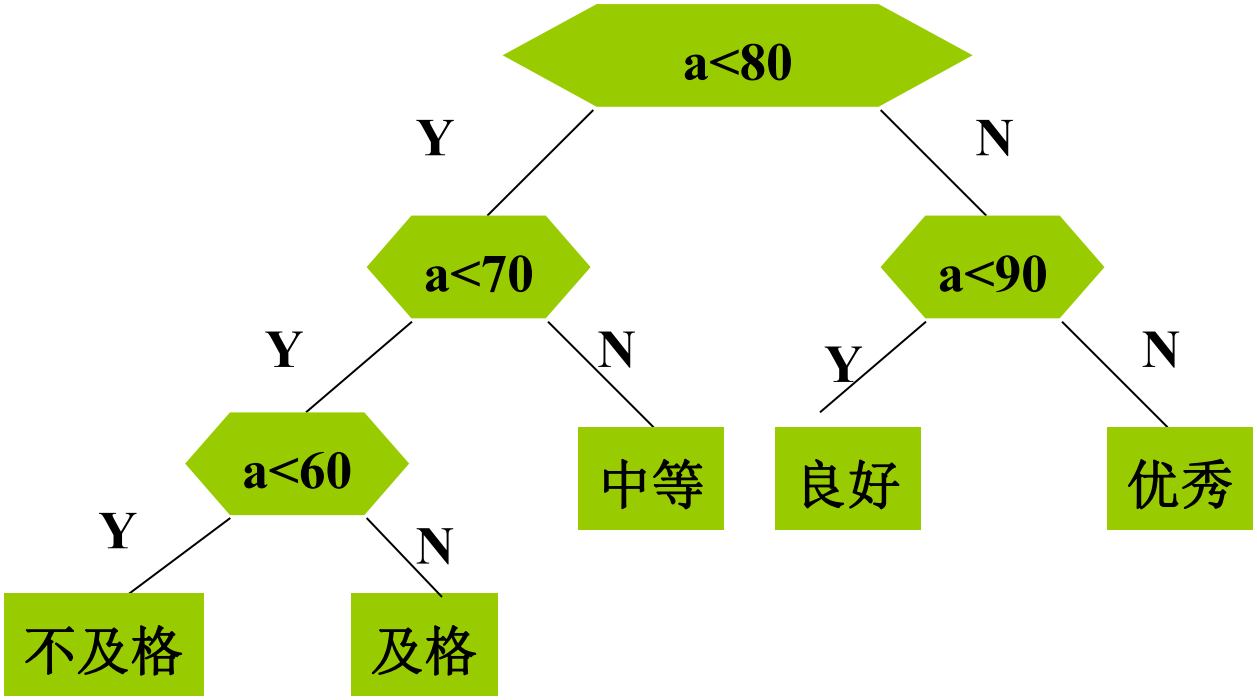
出现的频率：


5
15
40
30
10


100
40
60
70～79
30
30
80～89
15
15
60～69
0～59
5
10
90～100


70≤a<80
Y
N
中等
80≤a<90
Y
N
良好
60≤a<70
N
Y
a<60
及格
Y
N
不及格
优秀

改造成每个判定框只有一次比较的判定树：



实践证明：按照此棵判定树，输入10000个输入数据，原判定树需进行31500次比较，此判定树需进行22000次比较

- 问题

在有n个叶子结点的哈夫曼树中，其结点总数为（　　）

A、不确定　　B、2n-1

C、2n+1　　D、2n

分析在二叉树中，如果是一棵满二叉树，则树的总结点树与叶结点的关系是：

设叶子结点为n，则总结点数s：s=2n-1个。

哈夫曼树是一种特殊的满二叉树，因此若有n个叶子结点，则其总结点数也是2n-1。

哈夫曼树又是最优二叉树，它是一类带权路径长度（WPL）最短的树。

完全二叉树的叶子结点树与总结点数之间就不一定满足这种关系.

■ 生成树

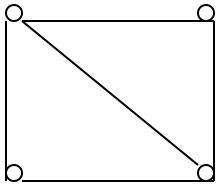
定义3.2.1 给定一个图 G，若图 G中存在一个生成子图是树 T，则称 T 是 G 的一个生成树。

由定义可见下面性质成立：

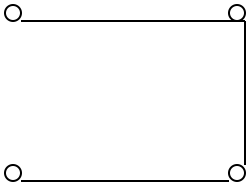
1.若 T 是 G 的一个生成树，则 $V(T)=V(G)$

2.若 G 有生成树 T，由于 T 连通，则 G 本身一定连通。

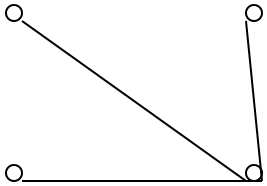
例1



G



λ



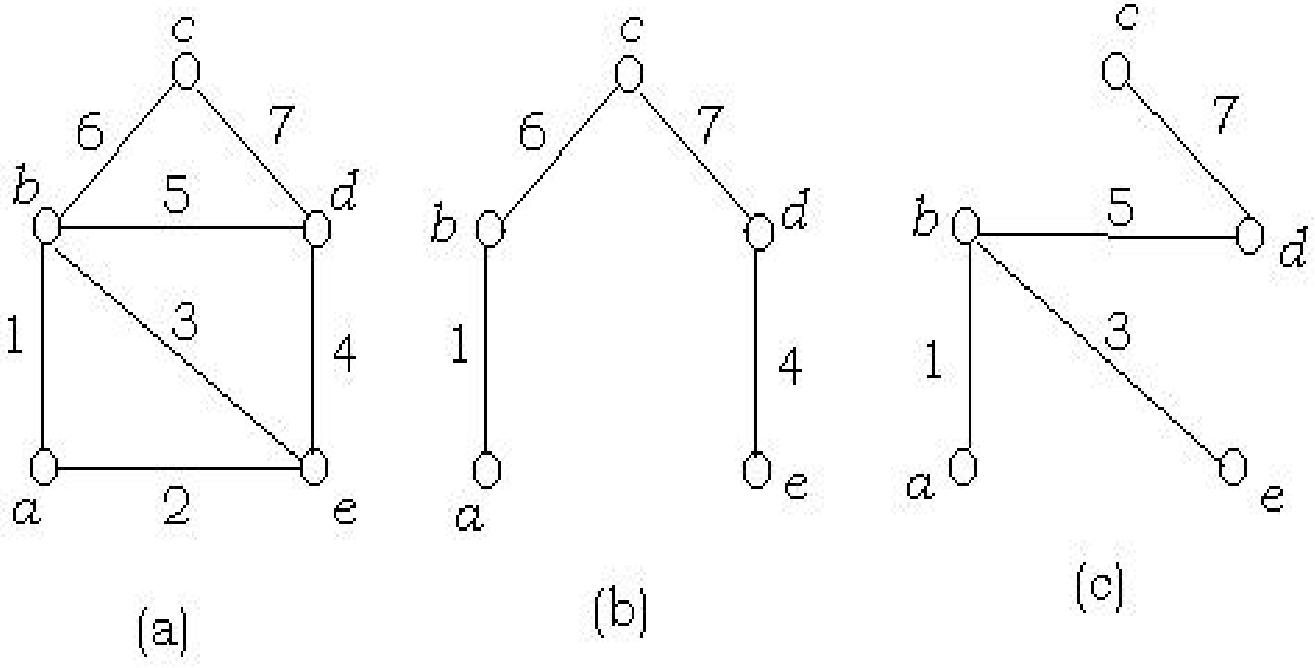
T'

定理　连通图至少有一棵生成树

证明 设连通图G没有回路，则它本身就是一棵生成树。若G至少有一个回路，我们删去回路上的一条边，得到$G1$，它仍然是连通的，并与G有相同的结点集。若$G1$没有回路，则$G1$就是G的生成树。若$G1$仍然有回路，再删去$G1$回路上的一条边，重复上面的步骤，直到得到一个连通图H，它没有回路，但与G有相同的结点集，因此H为G的生成树。

因此，一个连通图可能有许多生成树。因为取定一个回路后，就可以从中去掉任何一条边，去掉的边不一样，故可以得到不同的生成树。

例如，图 (a)中，相继删去边2、3和5，就得到生成树$T1$，如图(b)，若相继删去2、4和6，可得生成树$T2$，如图 (c)。



■ 最小生成树

在一个连通网的所有生成树中， 各边的代价之和最小的那棵生成树称为该连通网的最小代价生成树（Minimum Cost Spanning Tree），简称为最小生成树(MST)。

最小生成树有如下重要性质(MST性质)：

设 N=(V, {E}) 是一连通网，U 是顶点集V的一个非空子集。 若（u， v）是一条具有最小权值的边， 其中u∈U， v∈V-U， 则存在一棵包含边（u， v）的最小生成树。

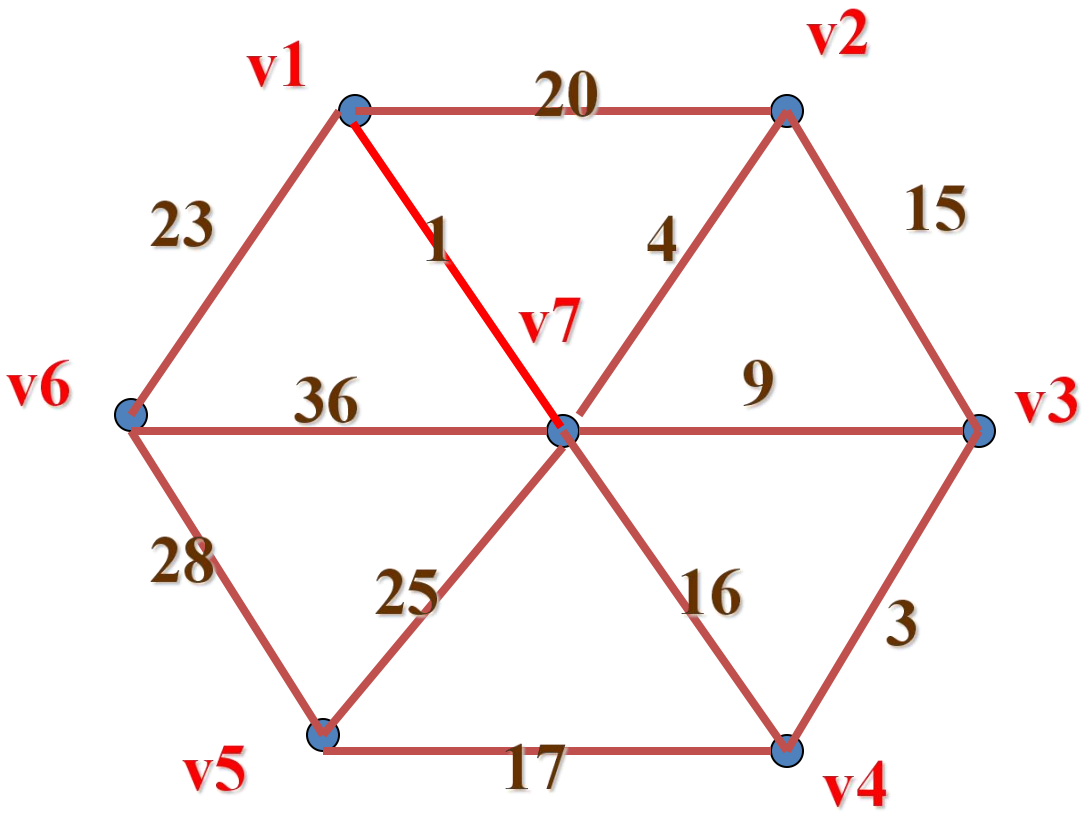
Kruskal算法 —避圈法

设n阶无向连通带权图G=<V,E,W>有m条边,不妨设G中无环(否则可先删去)，算法为：

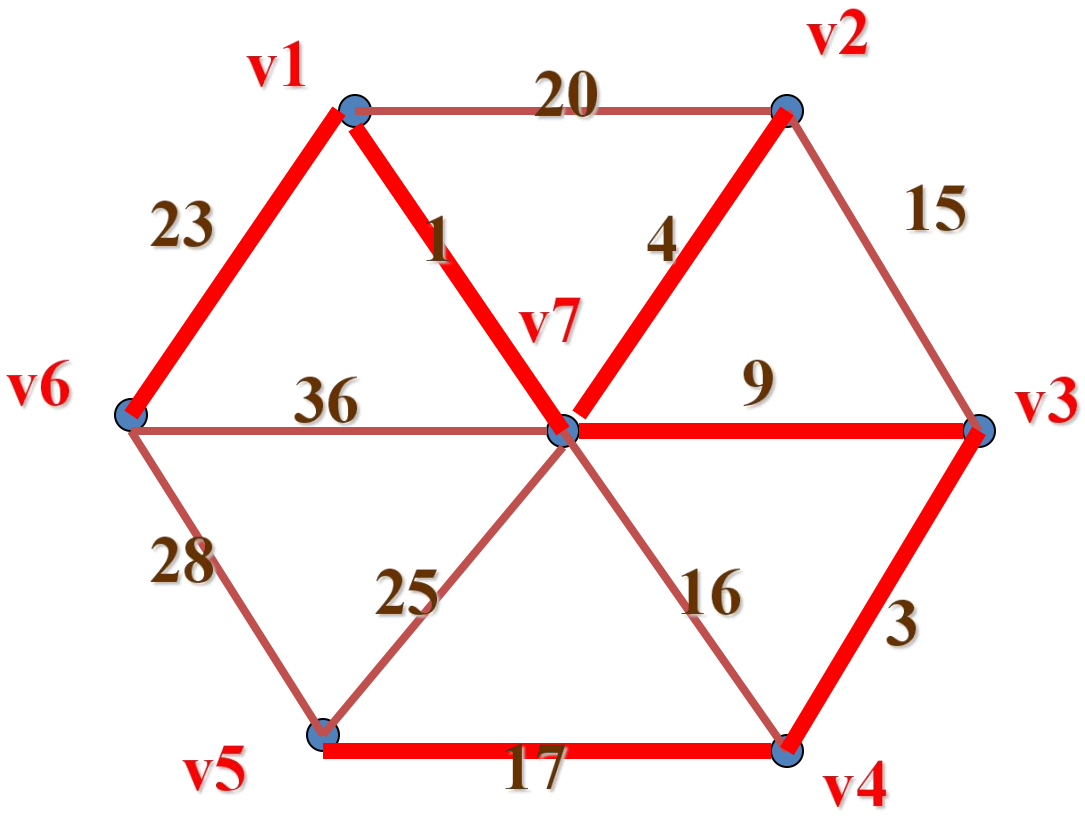
(1) 将m条边按权从小到大顺序排列，设为 $e_1,e_2,\ldots,e_m$。

(2) 取e_1在T中，然后依次检查$e_2,\ldots,e_m$，若e_j (j=2,3, …,m)与T中的边不能构成回路，则取e_j在T中，否则放弃e_j，考虑下一条边，直至j>m。

(3) 算法停止时得到的T为G的最小生成树。



避圈法



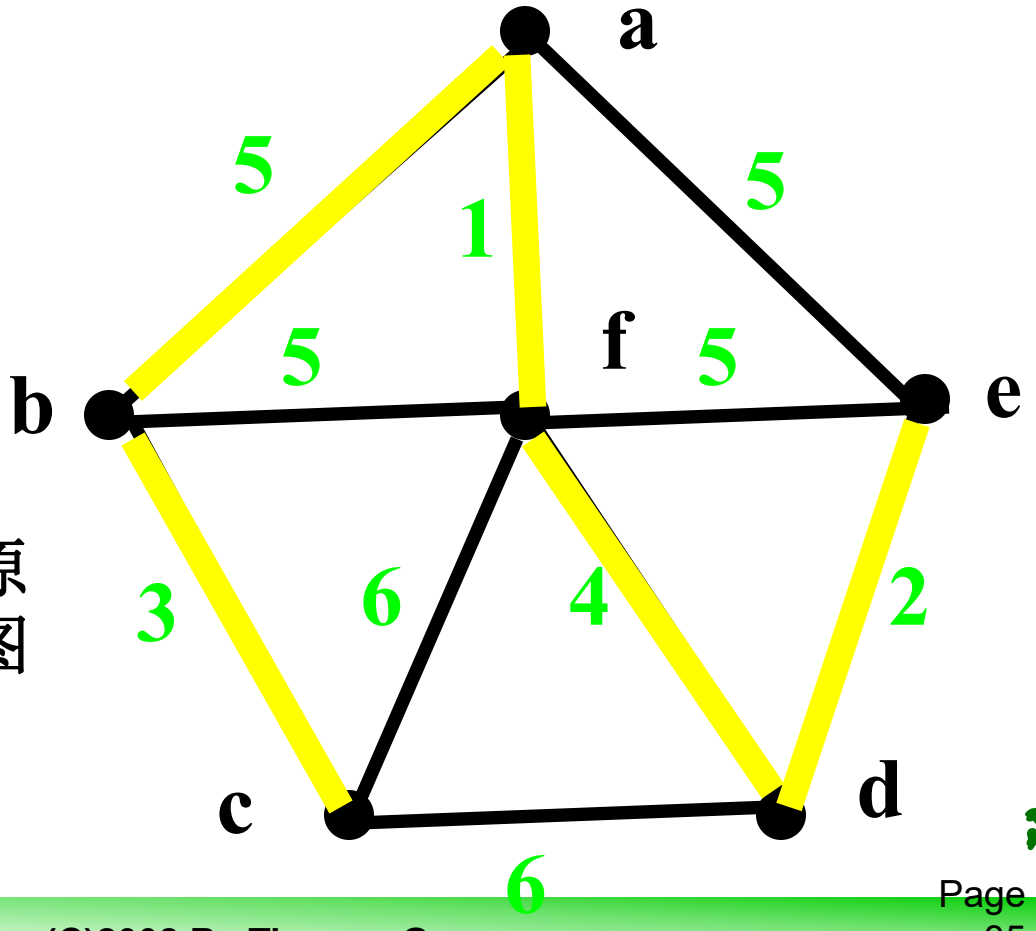
总造价=1+4+9+3+17+23=57

用避圈法求下图所示的最小生成树

解:

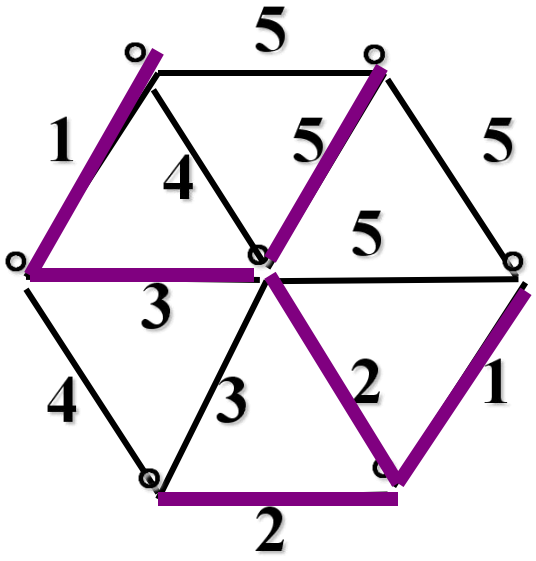
W(T)=1+2+3+4+5

=15



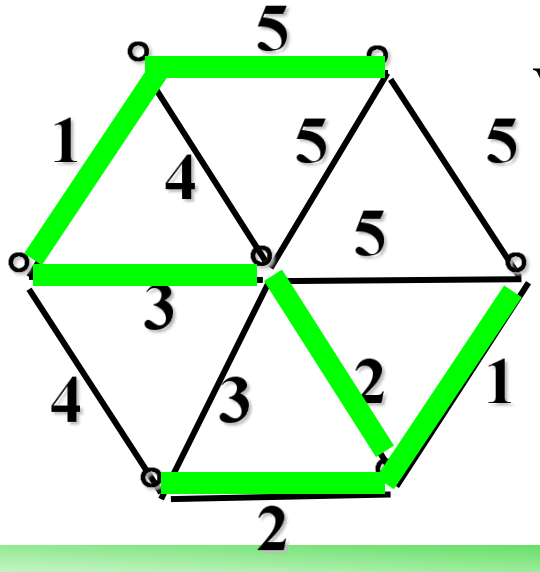
注意:
最小生成树的结点数与原图相等，边的数目比原图少。

用Kruskal算法求下图的最小生成树。



W(T)=1+1+2+3+2+5

=14



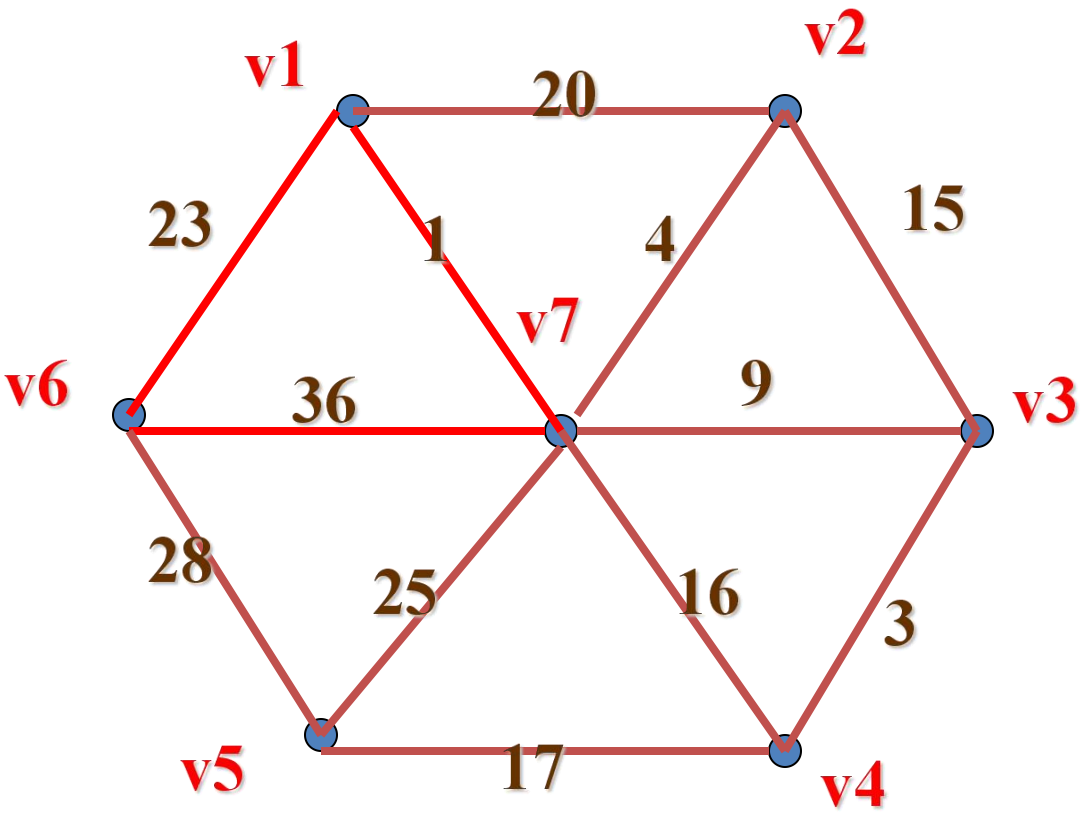
W(T)=1+1+2+3+2+5

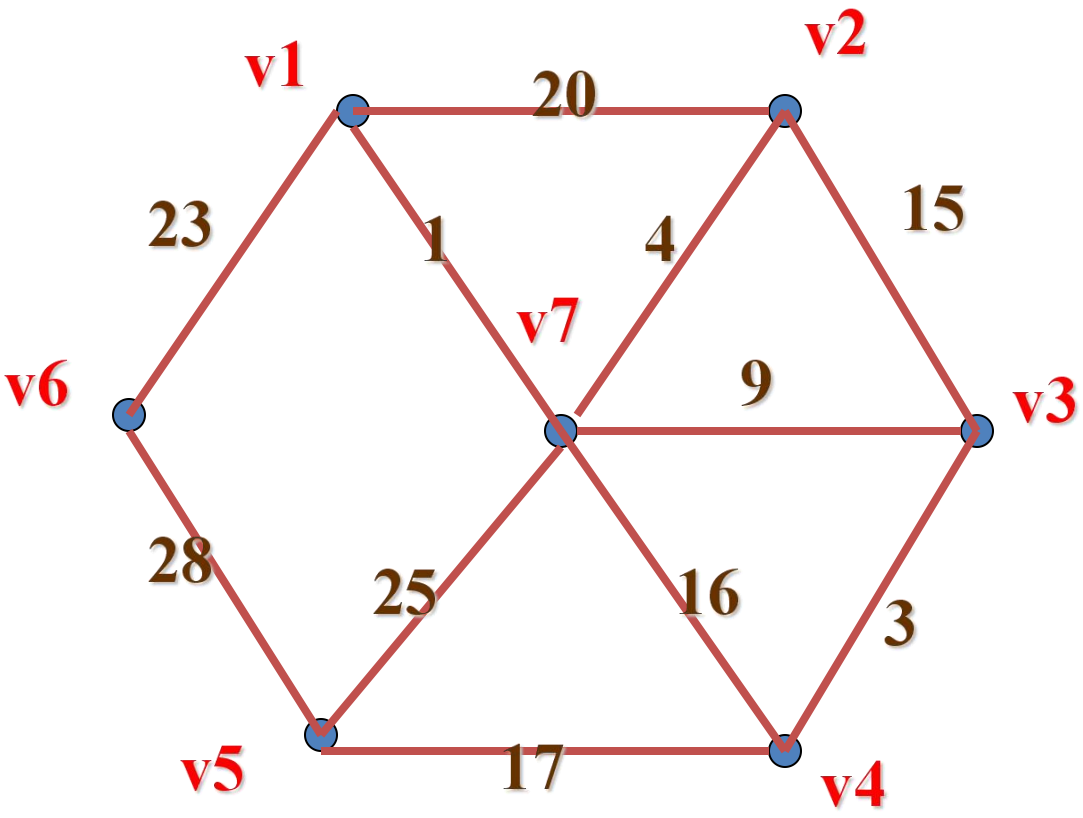
=14

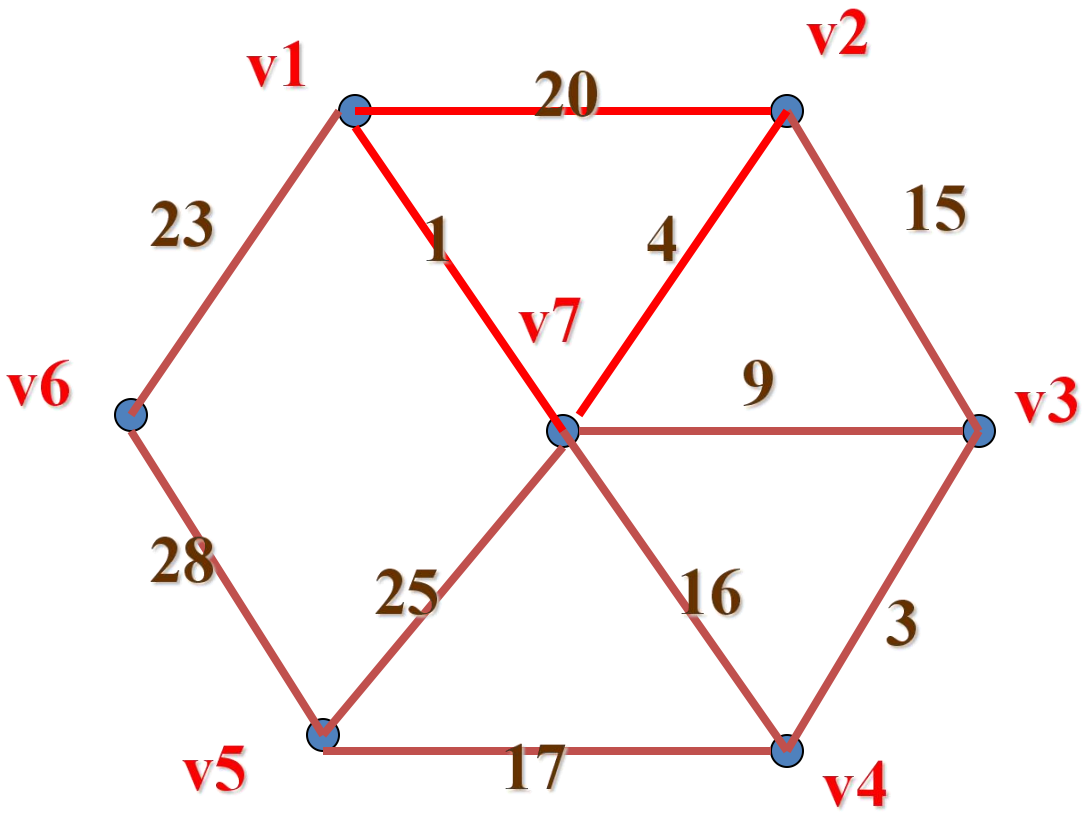
破圈法

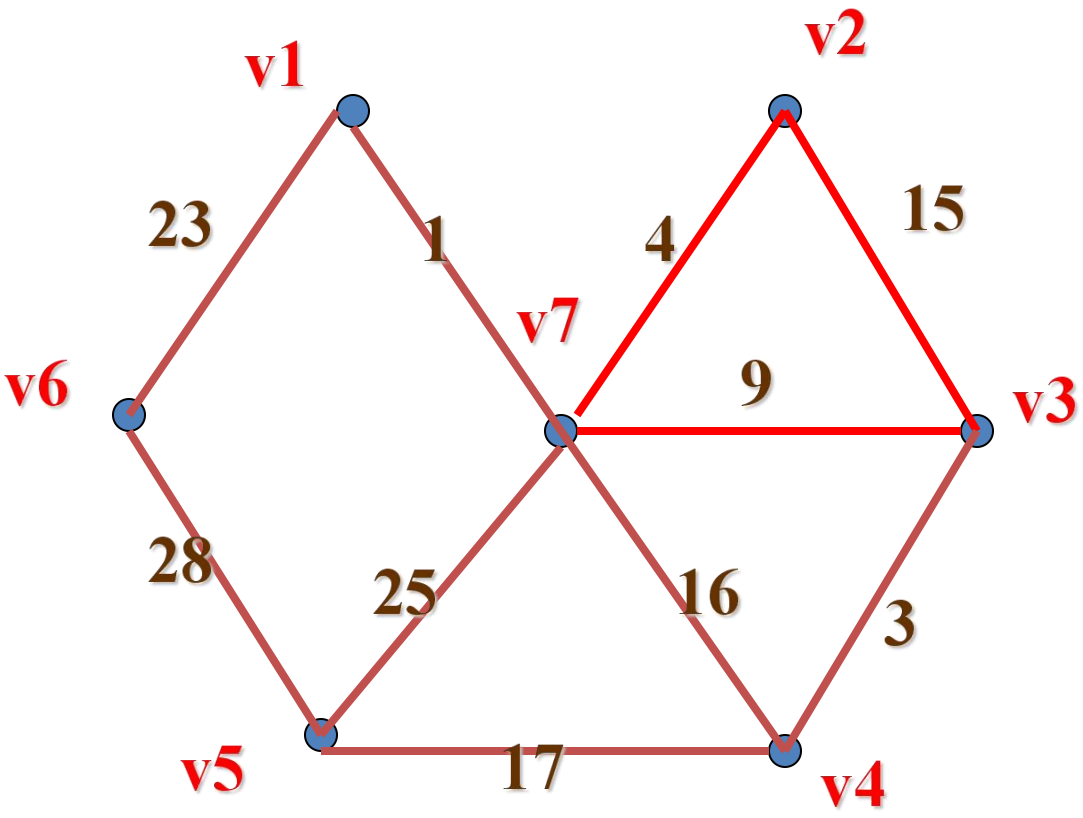
上述算法是贪婪地增加不构成回路的边，以求得最小生成树(最优树)，所以通常称为“避圈法”；

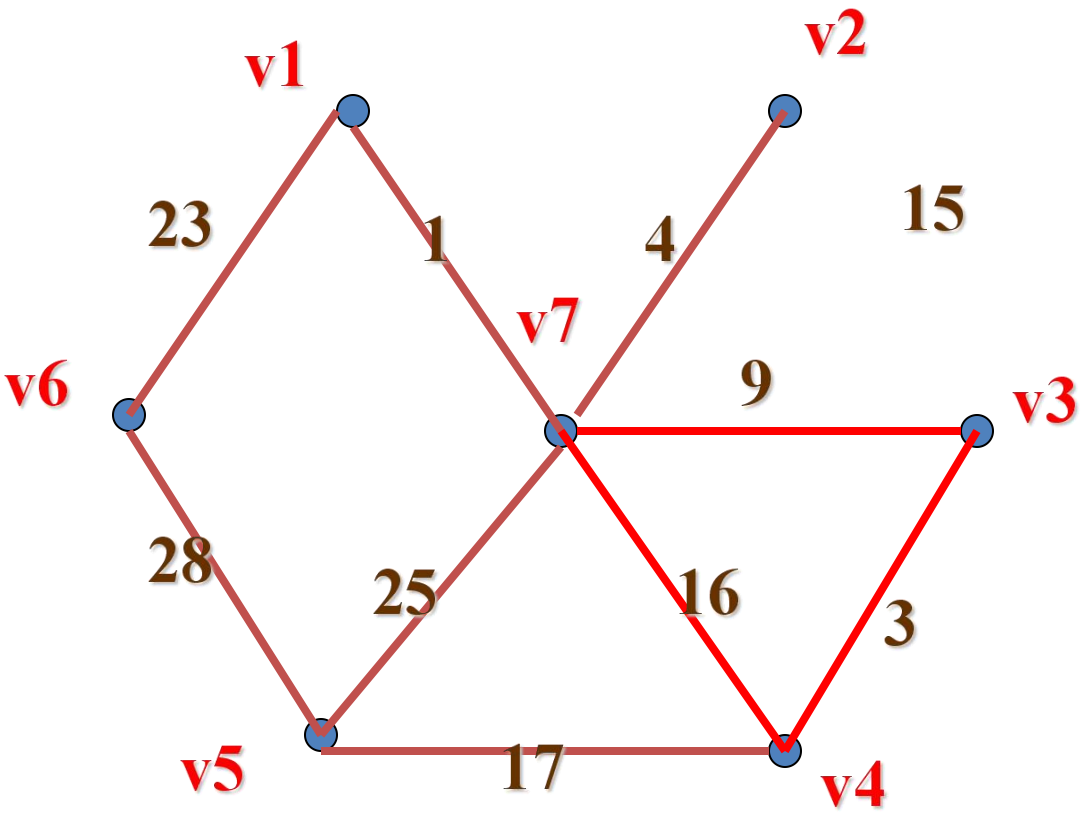
我们还可以从另一个角度来考虑最小成树(最优树)问题，由G的圈（回路）最优条件，我们也可以在原连通权图G中逐步删除构成回路中权最大的边，最后剩下的无回路的生成子图为最小成树(最优树)。我们把这种方法称为“破圈法”。

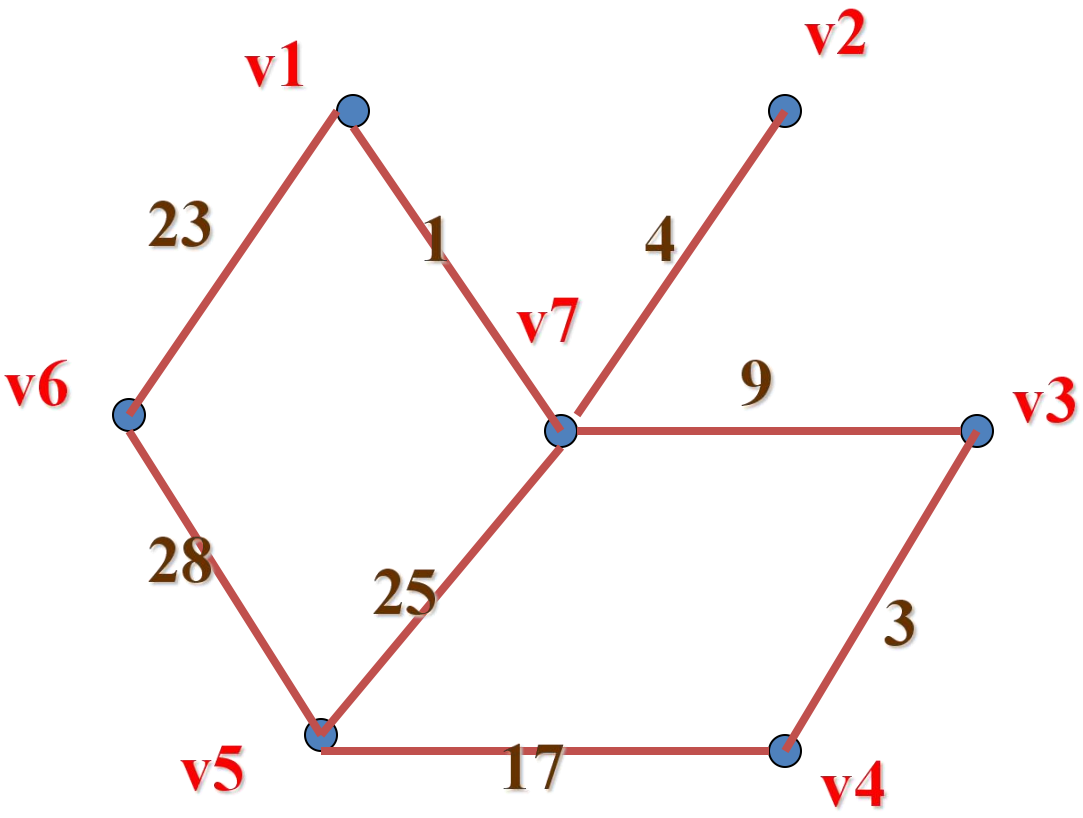

v1
v2
20
23
1
4
15
v7
v6
36
9
v3
28
25
16
3
v5
17
v4

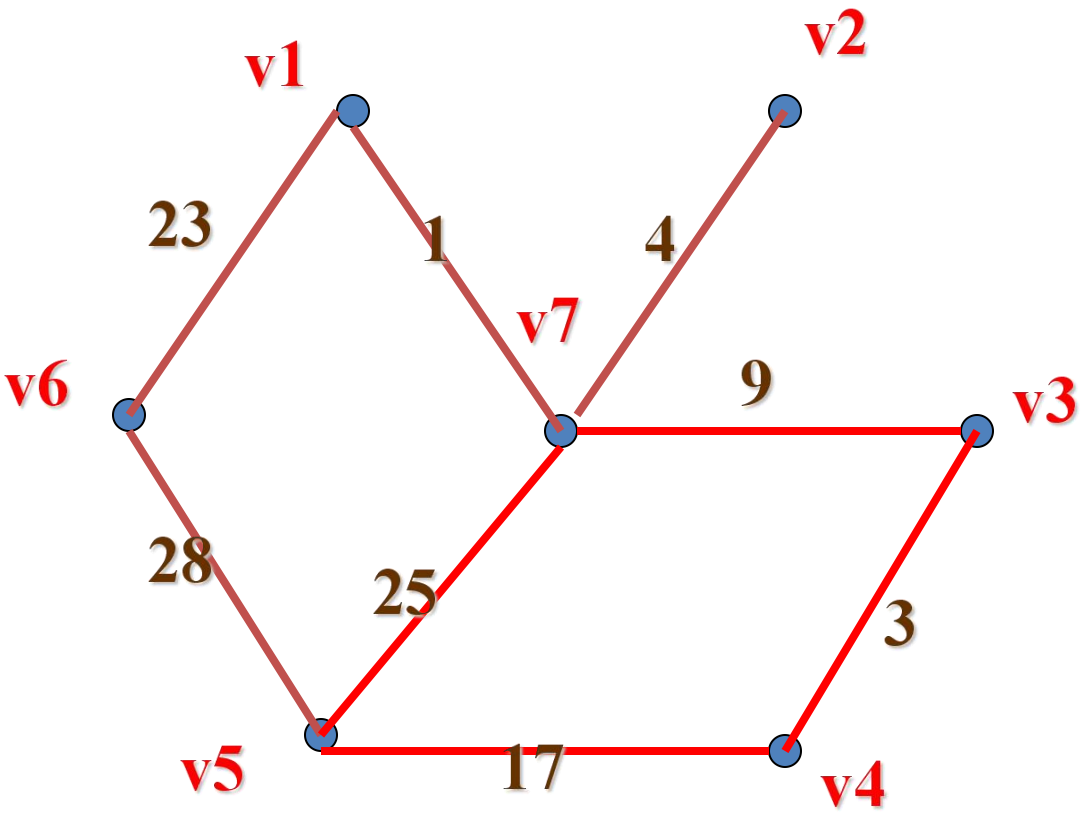

v1
v2
20
23
1
4
15
v7
v6
9
v3
28
25
16
3
v5
17
v4

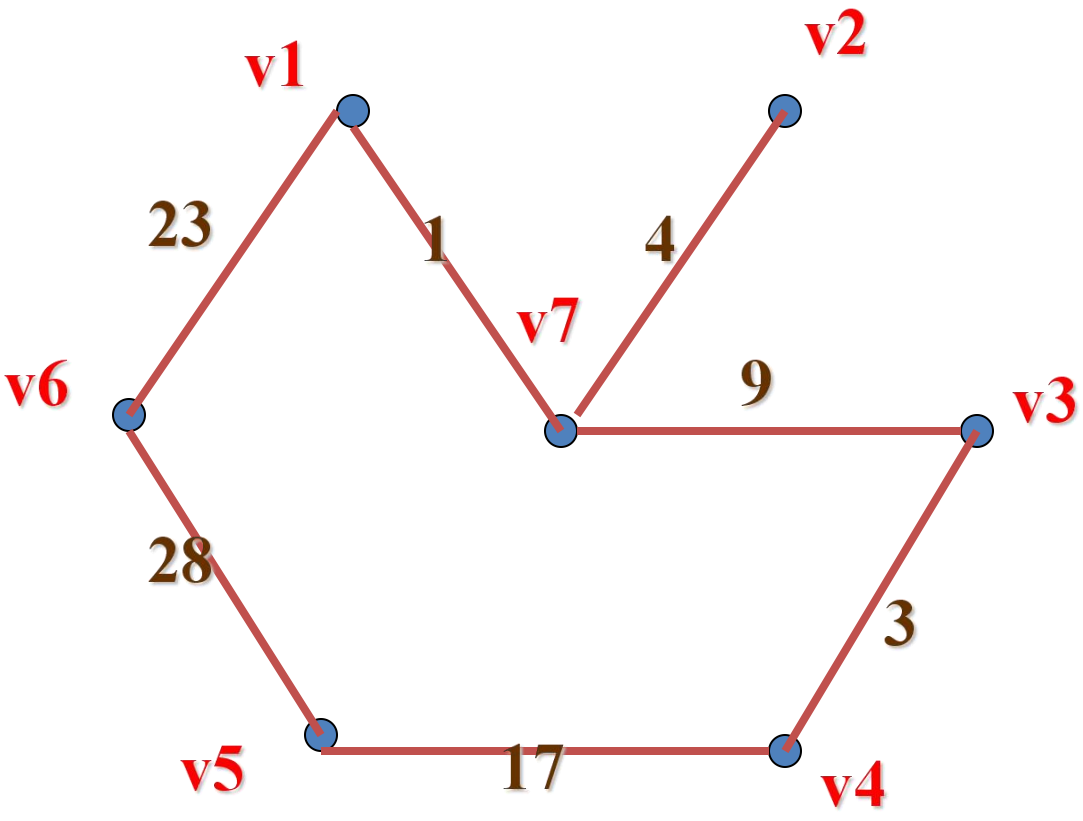

v1
v2
20
23
1
4
15
v7
v6
9
v3
28
25
16
3
v5
17
v4

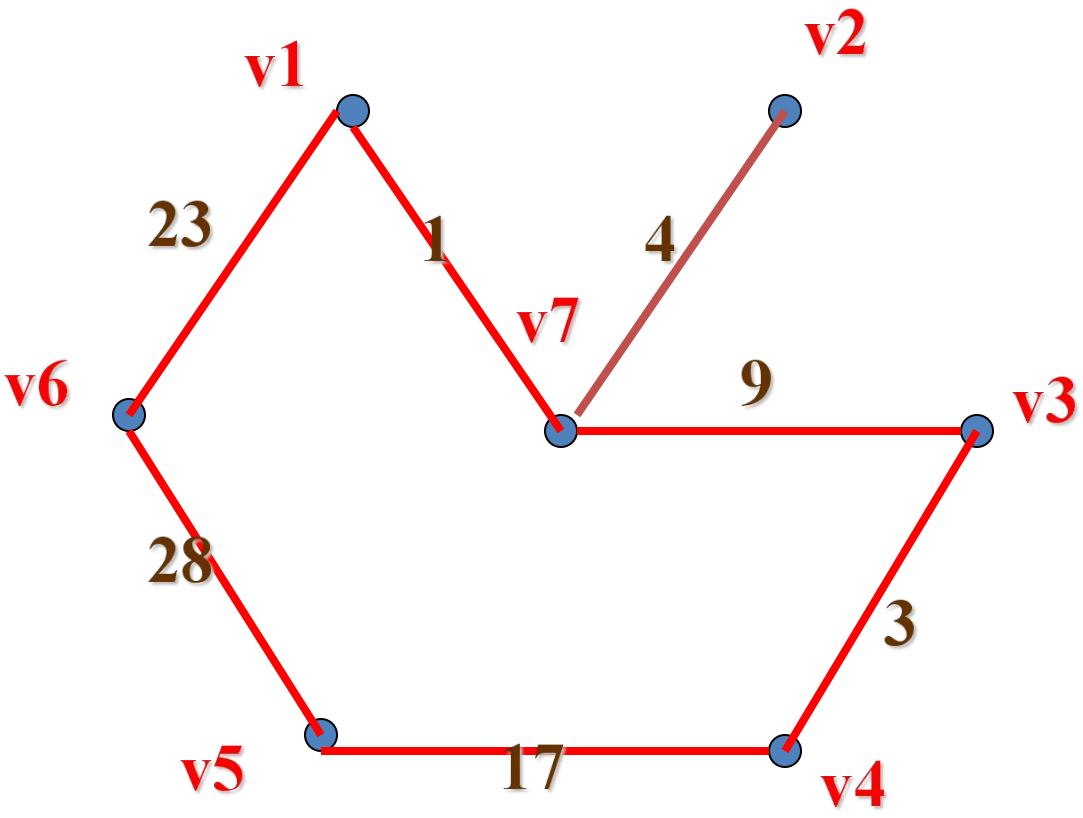

v1
v2
23
1
4
15
v7
v6
9
v3
28
25
16
3
v5
17
v4

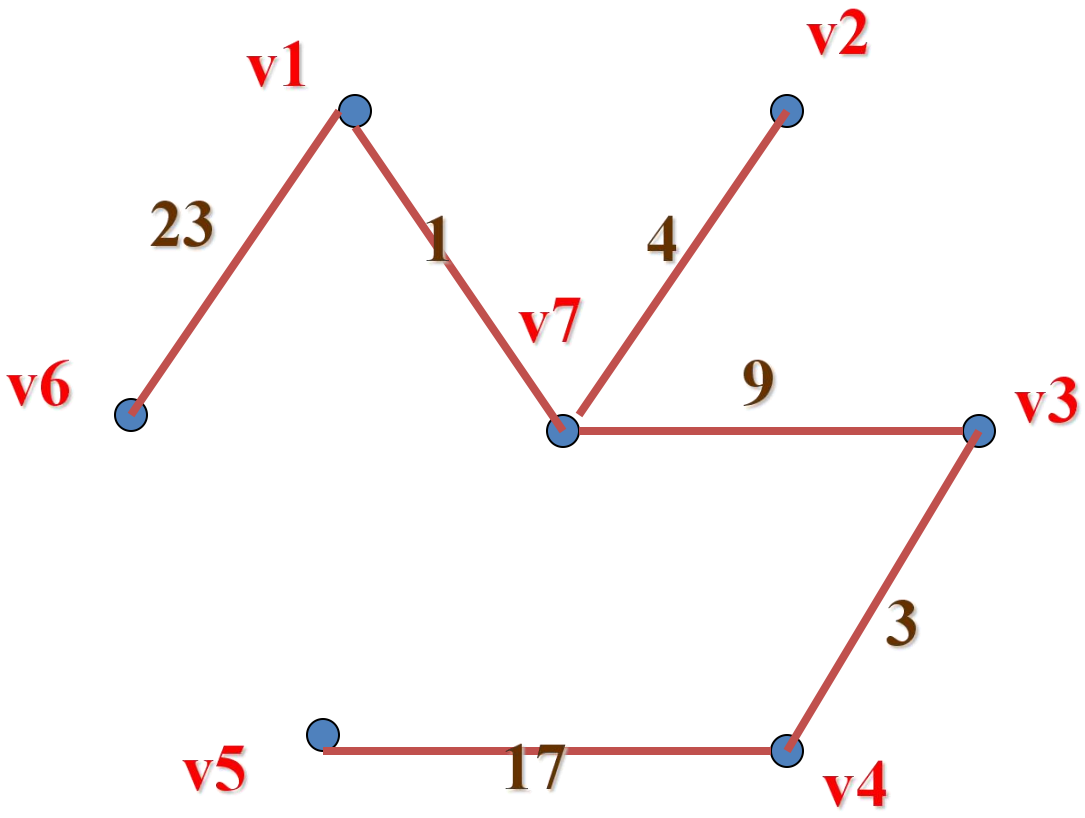

v1
v2
23
1
4
15
v7
v6
9
v3
28
25
16
3
v5
17
v4


v1
v2
23
1
4
v7
v6
9
v3
28
25
3
v5
17
v4


v1
v2
23
1
4
v7
v6
9
v3
28
25
3
v5
17
v4


v1
v2
23
1
4
v7
v6
9
v3
28
3
v5
17
v4


v1
v2
23
1
4
v7
v6
9
v3
28
3
v5
17
v4



总造价

=1+4+9+3+17+23=57

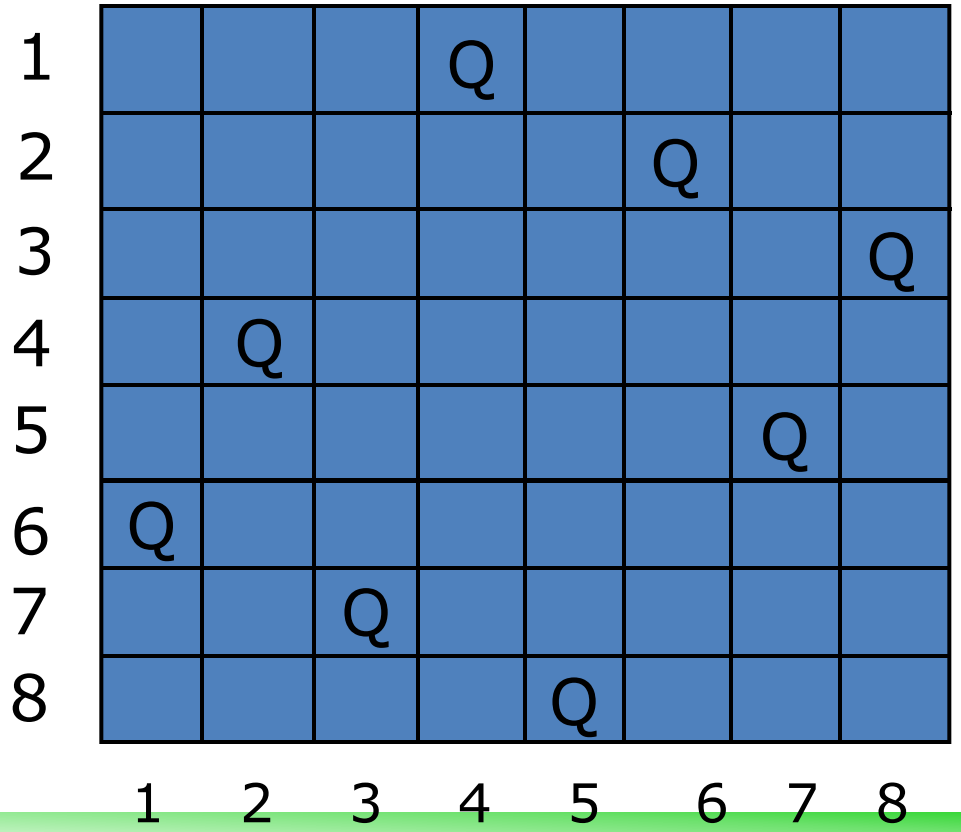
回溯法

有许多问题，当需要找出它的解集或者要求回答什么解是满足某些约束条件的最佳解时，往往要使用回溯法。回溯法的基本做法是搜索，或是一种组织得井井有条的，能避免不必要搜索的穷举式搜索法。这种方法适用于解一些组合数相当大的问题。

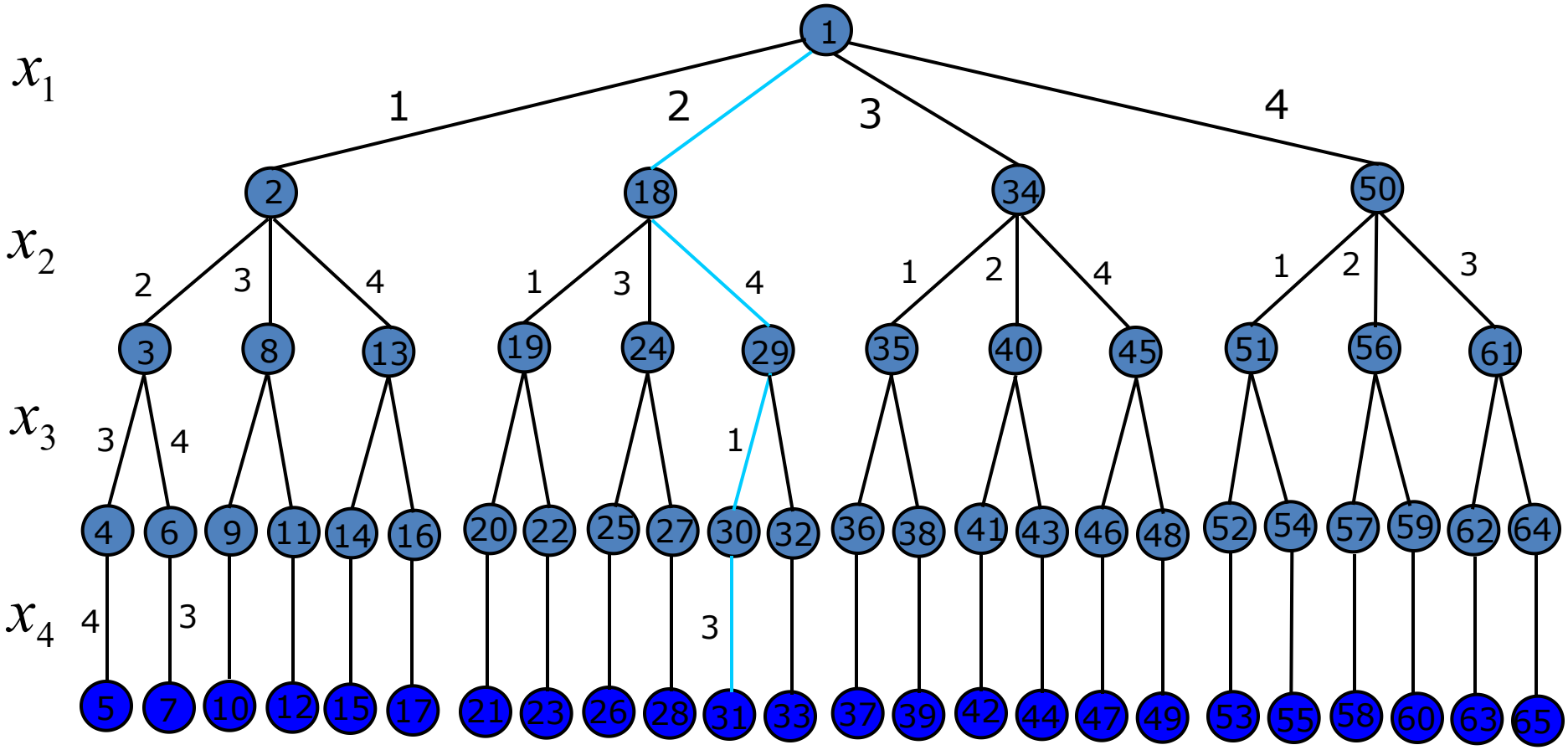
回溯法在问题的解空间树中，按深度优先策略，从根结点出发搜索解空间树。算法搜索至解空间树的任意一点时，先判断该结点是否包含问题的解。如果肯定不包含，则跳过对该结点为根的子树的搜索，逐层向其祖先结点回溯；否则，进入该子树，继续按深度优先策略搜索。

8后问题

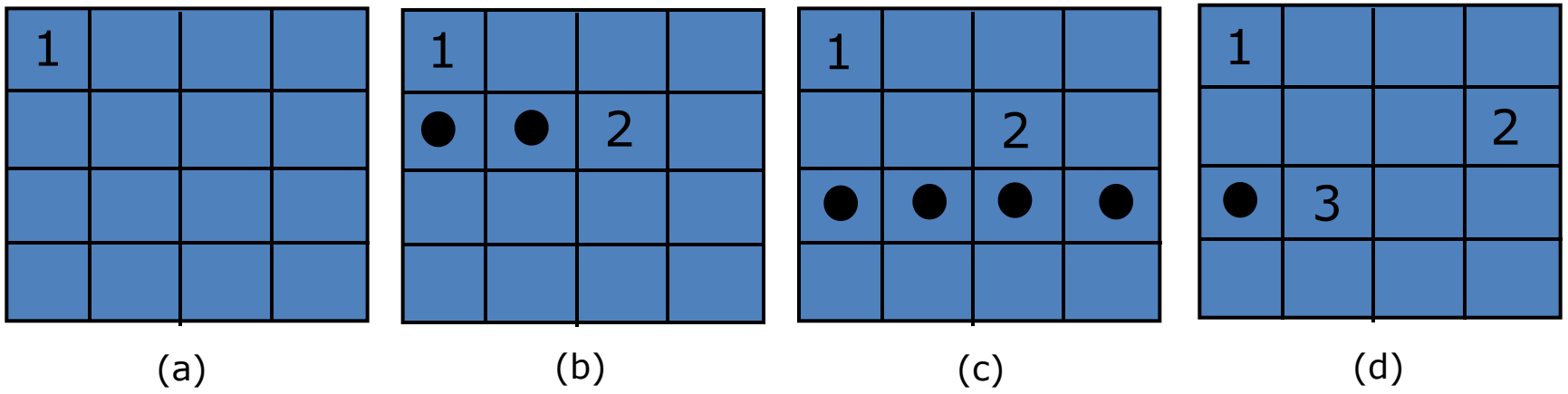
国际象棋中的皇后可吃掉放在同一行或同一列或同一斜对角线上对方的任何棋子。现在要在棋盘上放8个皇后，使得任何两个皇后都不形成相互攻击的局面，也就是说不能有任何两个皇后放在同一行或同一列或同一斜对角线上7



4后问题解空间的树结构



4后问题回溯法求解的例子



本章总结

- 最优二叉树
- 最小生成树
- 最优二叉树应用