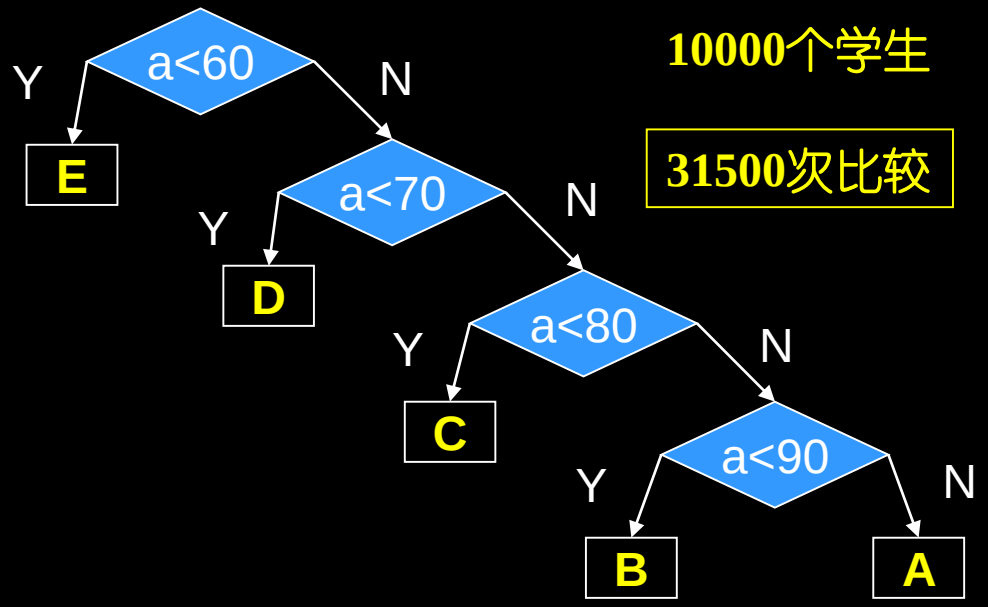
6.5 Huffman tree and coding

- Background

Score	0-59	60-69	70-79	80-89	90-100
Grade	E	D	C	B	A
Percent	0.05	0.15	0.40	0.30	0.10

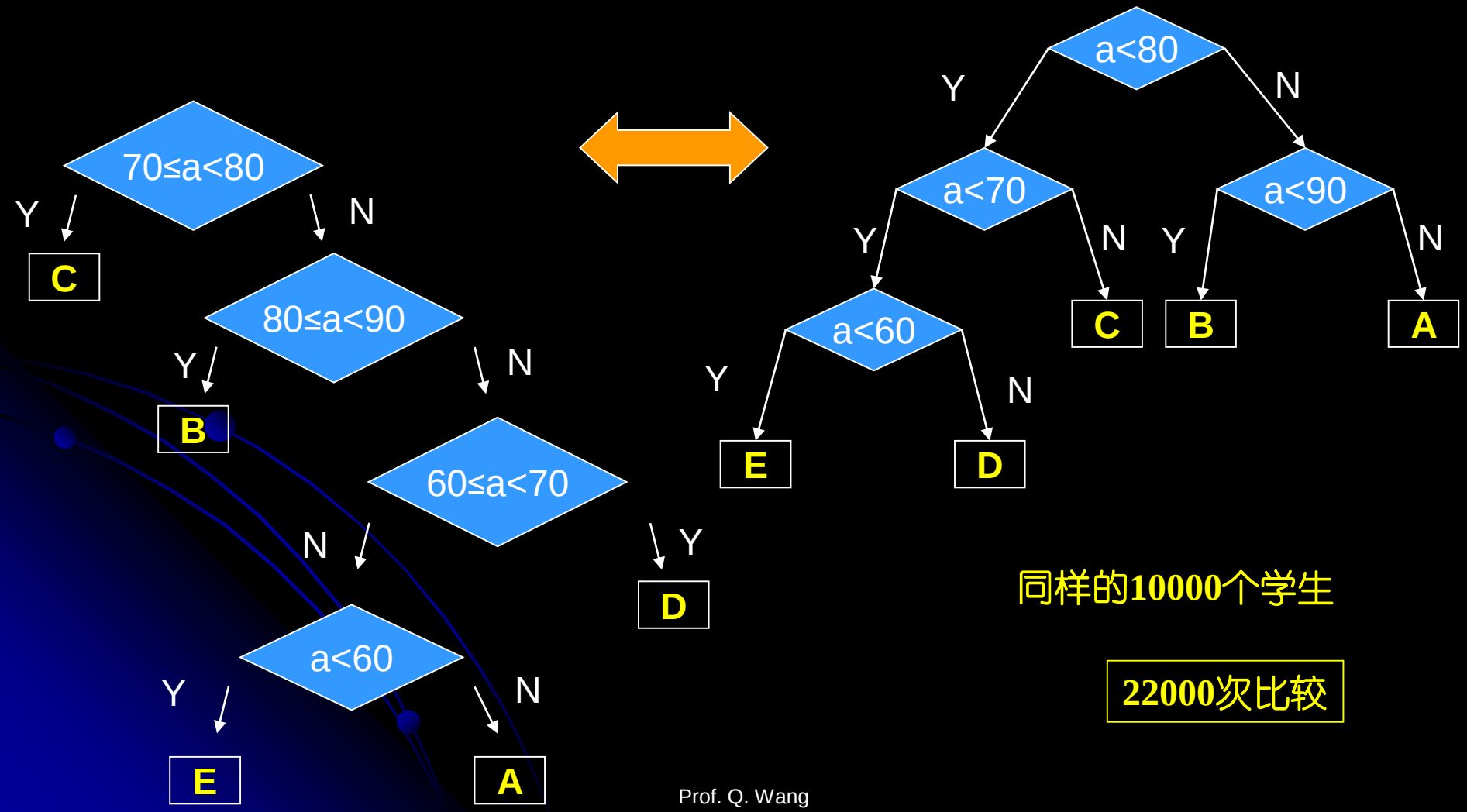
```
if (a<60) b='E';
   else if (a<70) b='D';
      else if (a<80) b='C';
         else if (a<90) b='B';
            else b='A';
```



10000个学生

31500次比较

Score	0-59	60-69	70-79	80-89	90-100
Grade	E	D	C	B	A
Percent	0.05	0.15	0.40	0.30	0.10



同样的10000个学生

22000次比较

```
if (a<60) b='E';
    else if (a<70) b='D';
        else if (a<80) b='C';
            else if (a<90) b='B';
                else b='A';
```

```
if (a<80) {
    if (a<70) {
        if (a<60) b='E';
        else b='D';
    }
    else b='C';
}
else {
    if (a<90) b='B';
    else b='A';
}
```

1. Definition of Huffman tree (定义及构造方法)

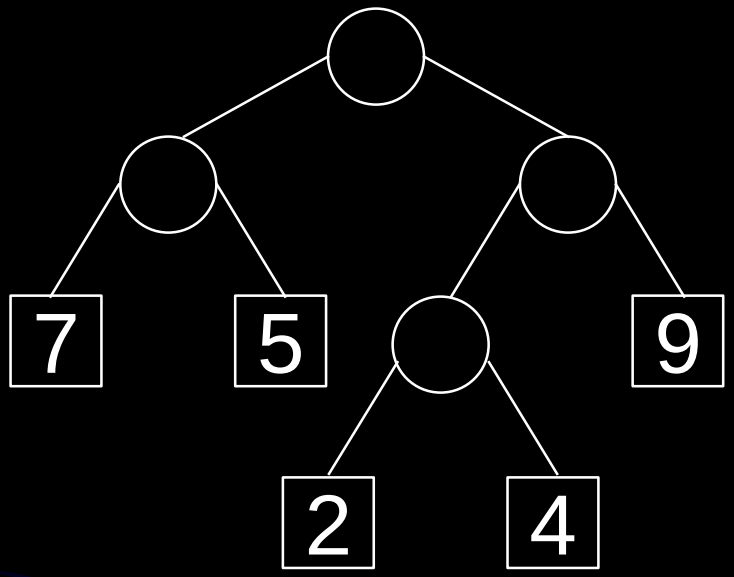
1）路径与路径长度：
从树中一个结点到另外一个结点之间的分支称为这两个结点之间的路径，路径上分支的数目称做路径长度。

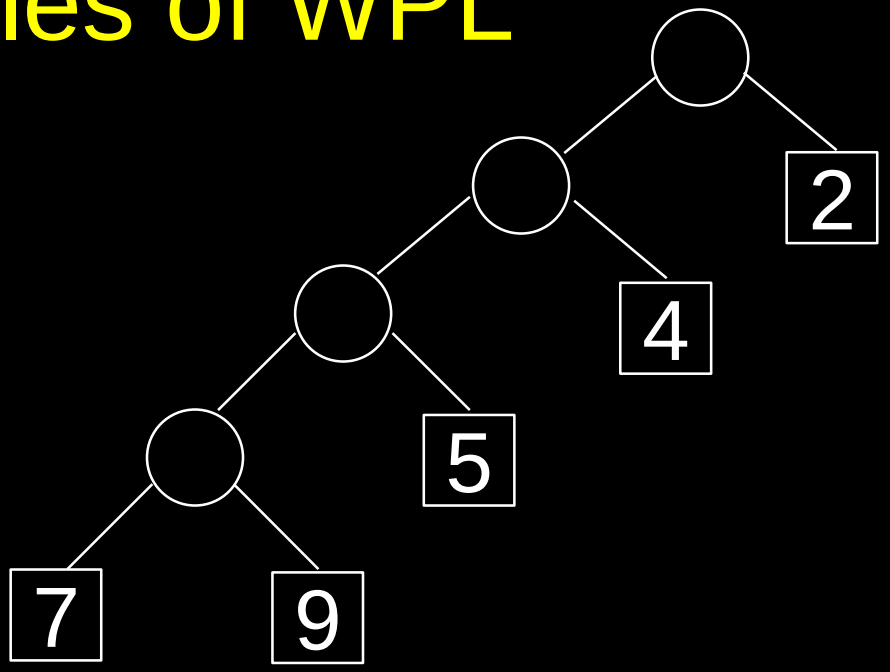
2）结点的带权路径长度：
从根结点到结点之间的路径长度与该结点上权值的乘积。

3）树的带权路径长度：
树中所有叶子结点的带权路径长度之和。

$$WPL = \sum_{k=1}^{n} W_k L_k$$ **(对所有叶子结点)**

Examples of WPL





WPL(T)=

7×2+5×2+2×3+4×3

+9×2=60

WPL(T)=

7×4+9×4+5×3+4×2

+2×1=89

Question

- What kind of binary tree has shortest path length?
- Answer: complete binary tree

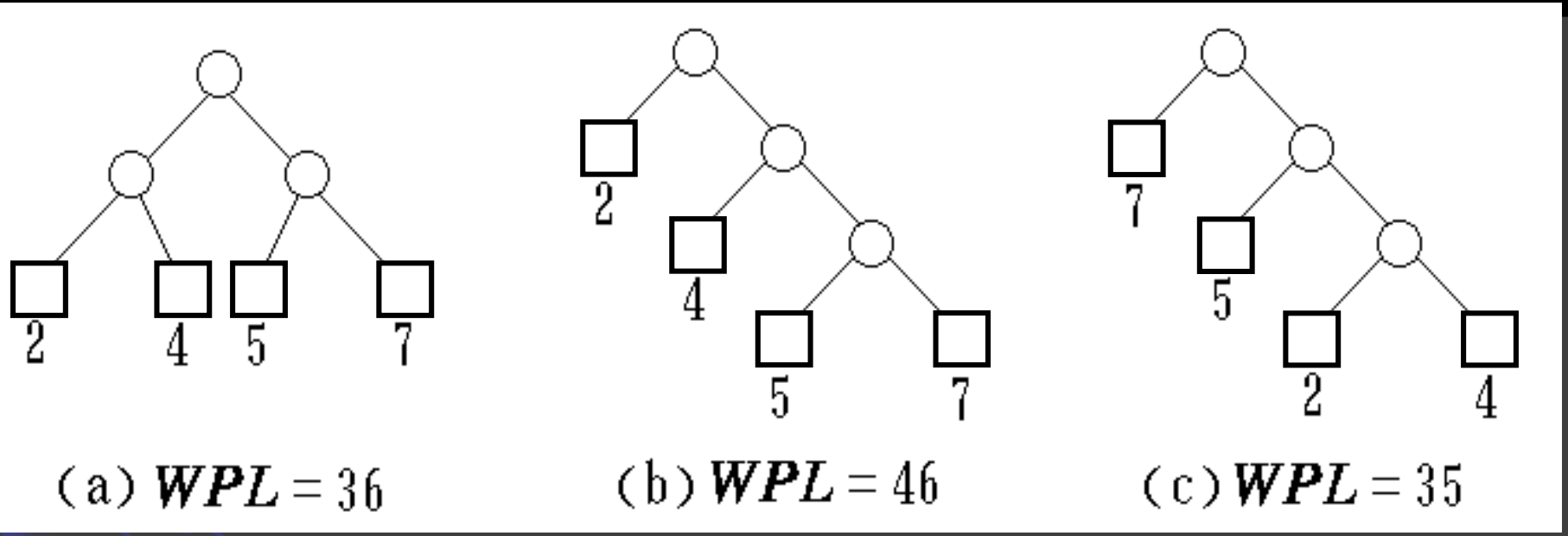
2. 哈夫曼树的基本概念

n 个带权叶子结点构成的所有二叉树中，必存在一棵其**带权路径长度WPL最小值**的树，我们称之为**最优二叉树**或**哈夫曼树**。

特点：权值越大的叶子结点离根结点越近！

哈夫曼(Huffman)于1952年提出了得到这种树的算法－－哈夫曼算法。

WPL comparisons of same weights



(a) WPL=(2+4+5+7)×2=36
(b) WPL=2×1+4×2+(5+7)×3=46
(c) WPL=7×1+5×2+(2+4)×3=35

Question: how to establish an extended binary tree with smallest WPL?

3. Huffman algorithm

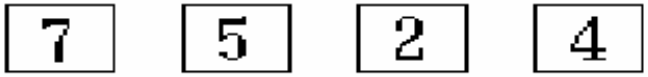
(1) 根据给定的n个权值$\{w_1, w_2, \ldots, w_n\}$，构成n棵二叉树的集合$F=\{T_1,T_2,\ldots,T_n\}$，其中每一棵二叉树$T_i$中只有一个带权为$w_i$的根结点，其左右子树为空。

(2) 在F中选取**两棵权值最小**的树作为左右子树(原则上没有要求左边的小或者右边的小)构造一棵新的二叉树，且新二叉树的根结点的权值为**其左右子树的根结点权值之和**。

(3) 在F中删除这两棵树，同时将新得到的二叉树加入F中。

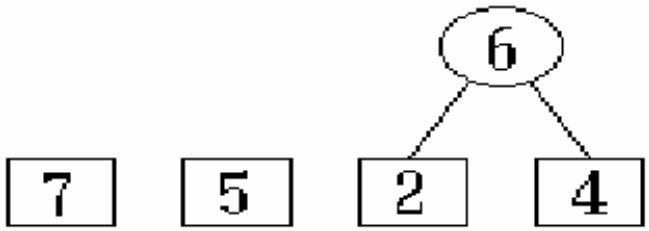
(4) 重复(2)和(3)，直到F中只含一棵树为止。该树即为哈夫曼树。

F : { 7 } { 5 } { 2 } { 4 }



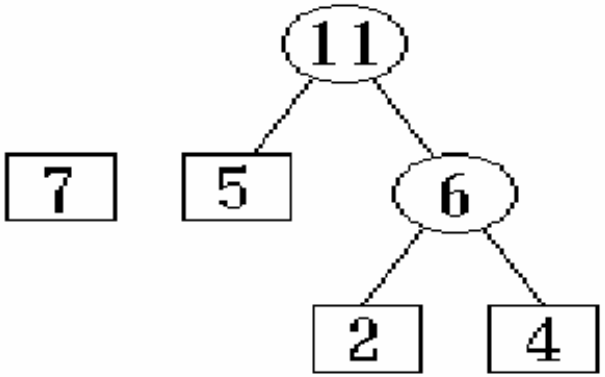
(a) Initilization

F : { 7 } { 5 } { 6 }



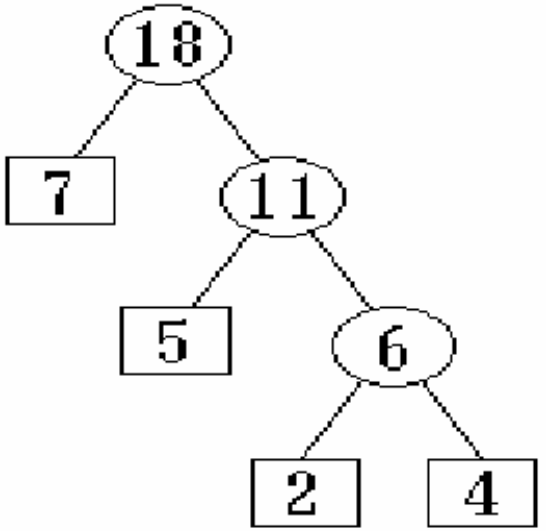
(b) Merge {2} and {4}

F : { 7 } { 11 }



(c) Merge {5} and {6}

F : { 18 }



(d) Merge {7} and {11}

Example

- A communication system has eight symbols

c1, c2, c3, c4, c5, c6, c7, c8

and the probability of each symbol is

5, 25, 3, 6, 10, 11, 36, 4

Try to design a coding method.

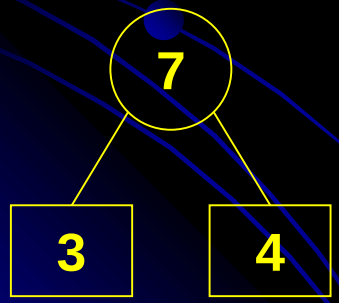
Step1: establish a Huffman tree;
Step2: coding.

Step 0:

5	25	3	6	10	11	36	4

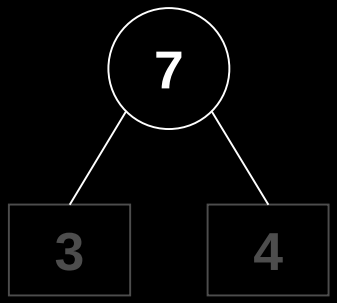
Step 1:

5	25	3	6	10	11	36	4



5	25	6	10	11	36

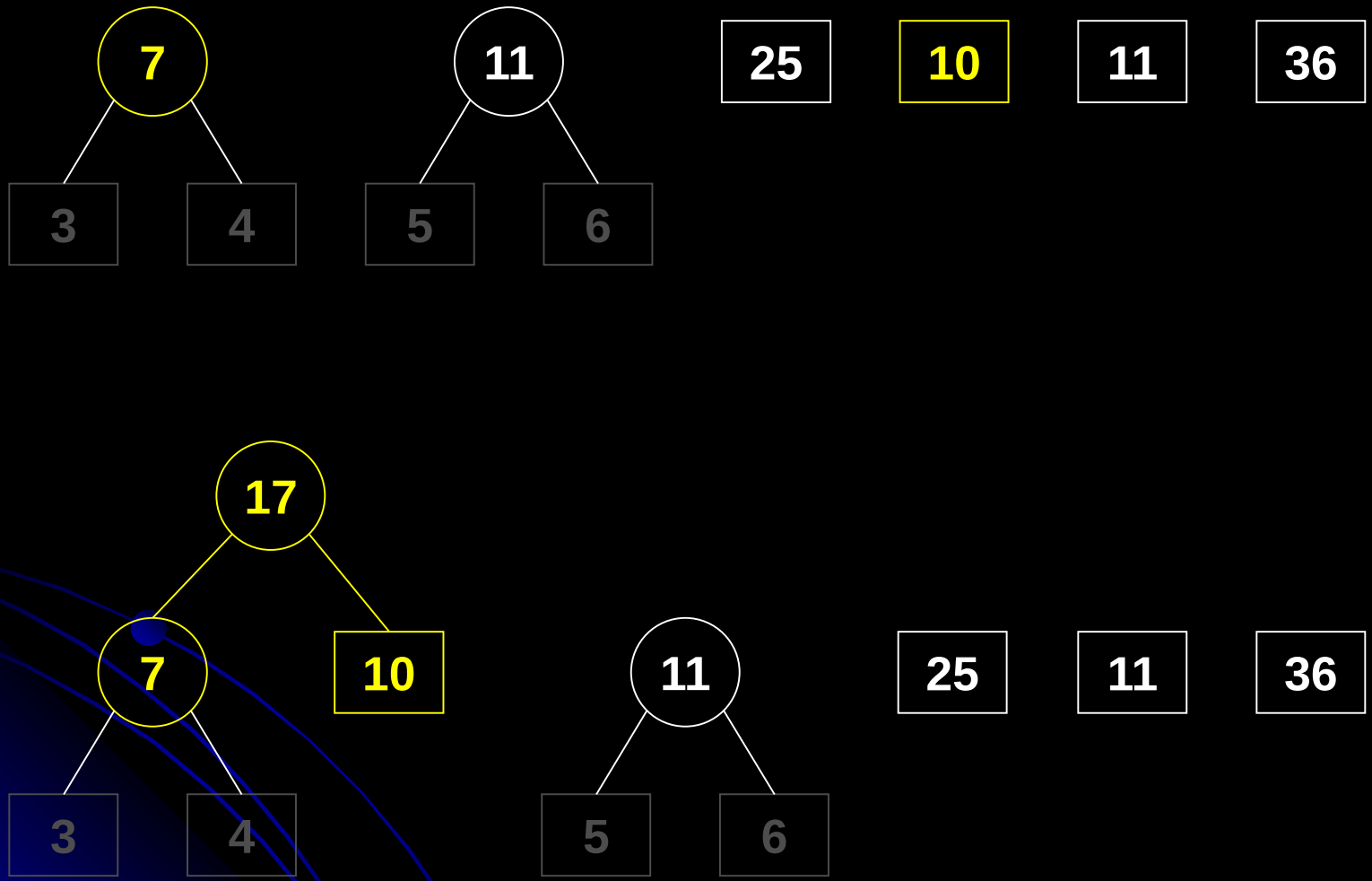
Step 2:



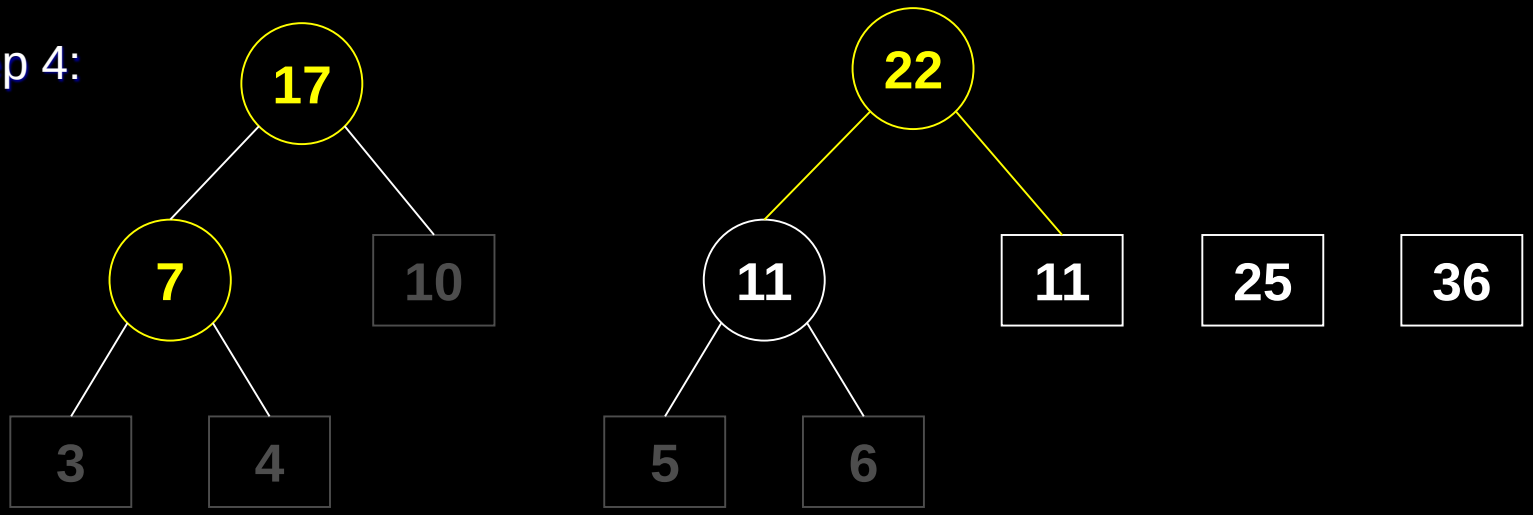
5 25 6 10 11 36

25 10 11 36

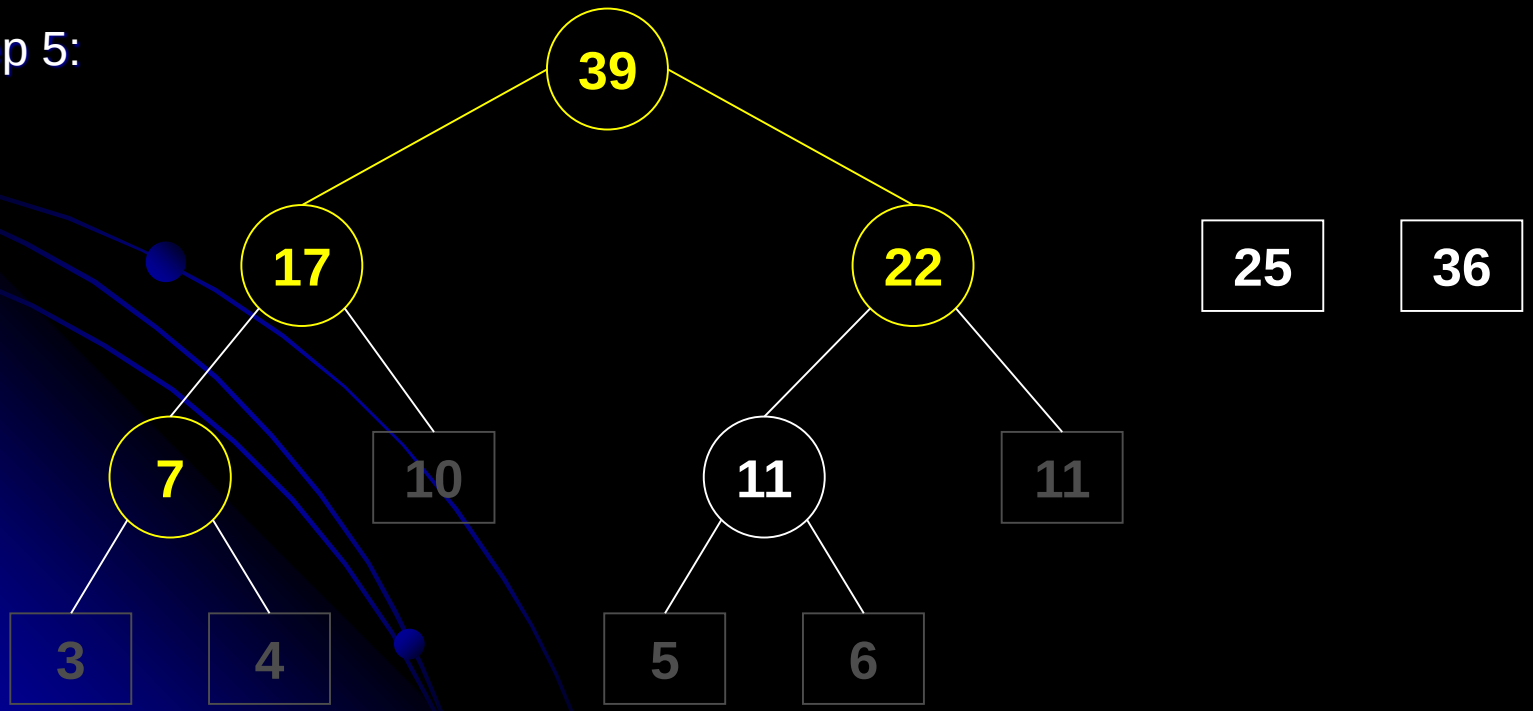
Step 3:



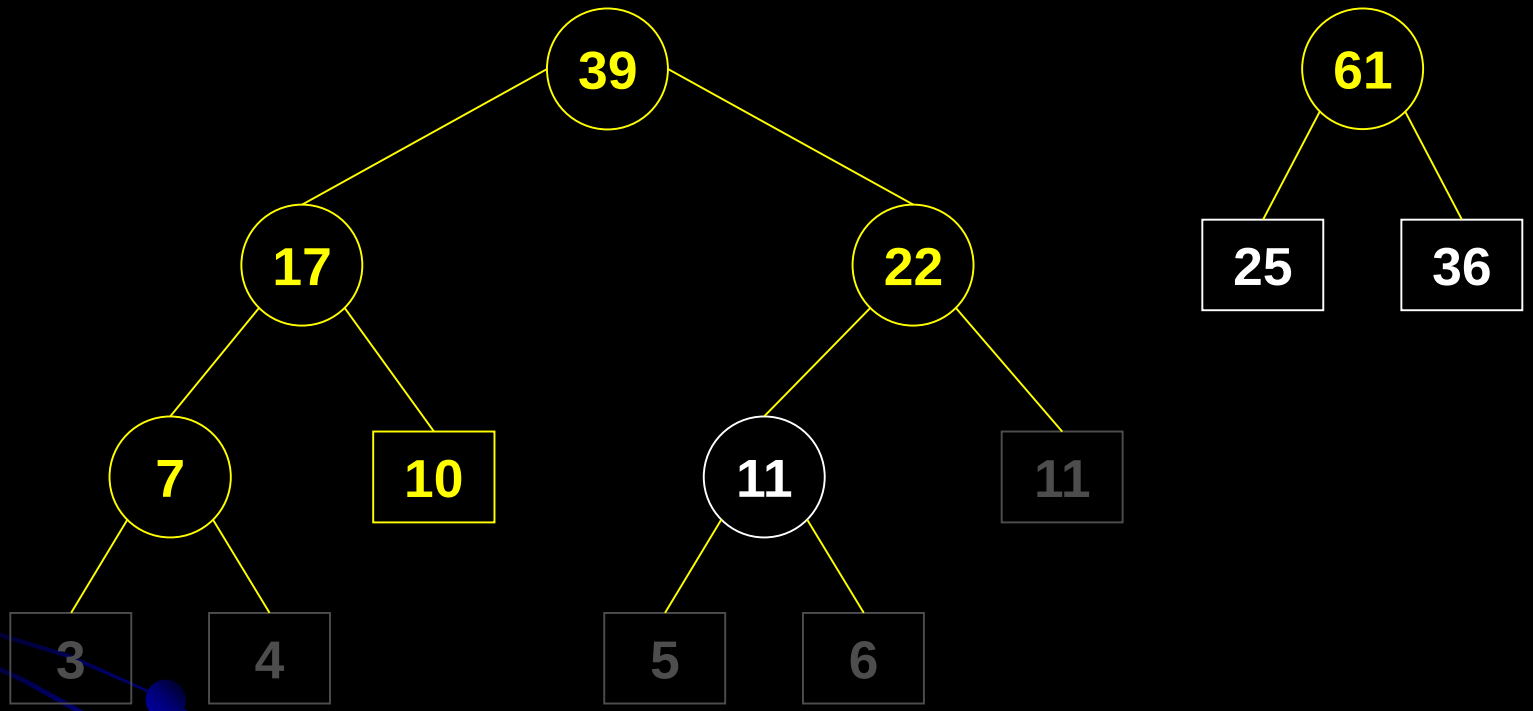
Step 4:



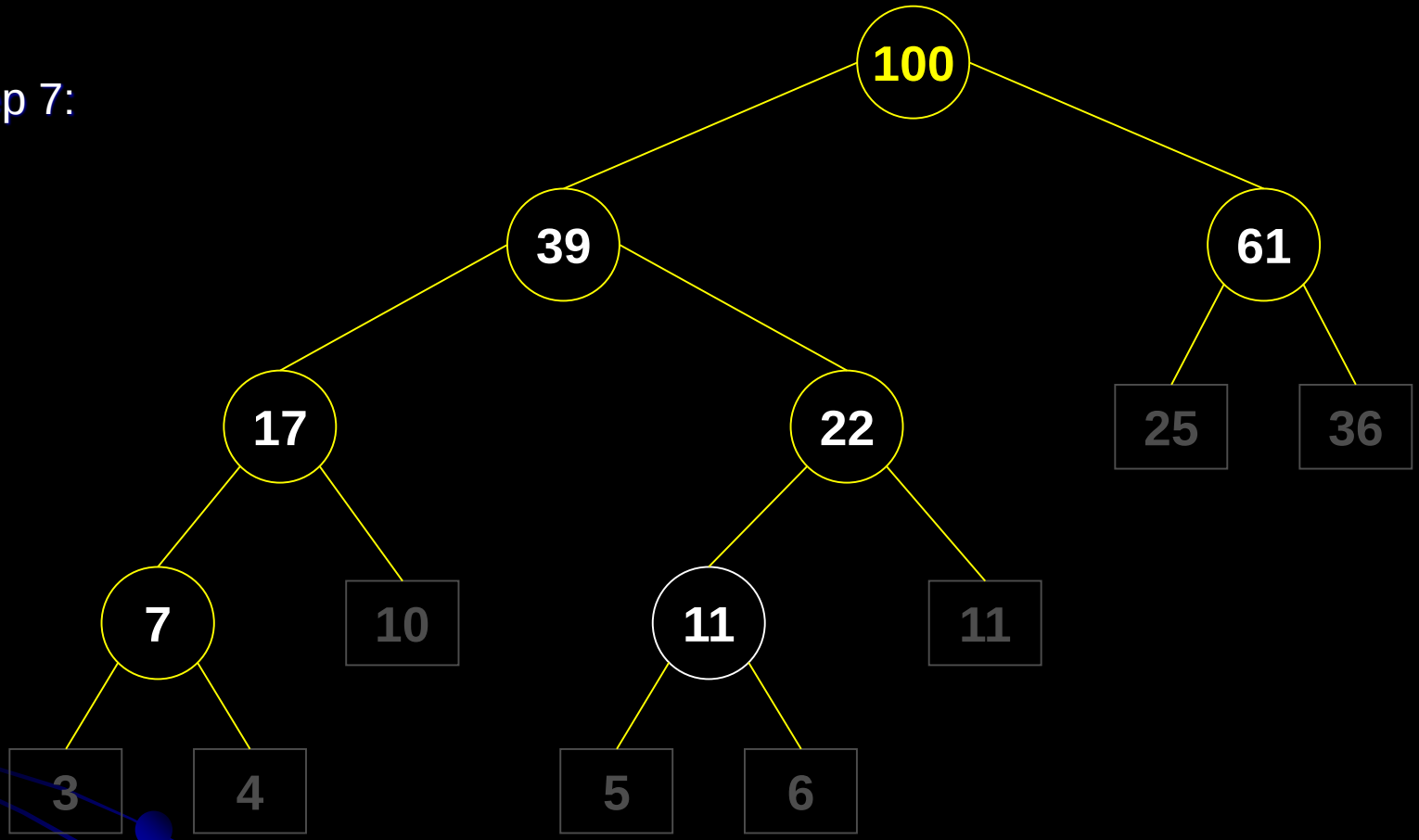
Step 5:



Step 6:



Step 7:



Done!

4. 哈夫曼算法的实现

由于**哈夫曼树中没有度为1的结点**(这类树又称严格的(strict)(或正则的)二叉树)；则一棵有n个叶子结点的哈夫曼树共有2n-1个结点（因$n_0=n_2+1$)，可以用有2n-1个元素的数组来存储。

每个结点包含其双亲信息和孩子结点的信息，所以使用**静态三叉链表**表示。

4. 哈夫曼算法的实现—类型定义

```
#define N  20                    /*叶子结点个数的最大值*/
#define M   2*N-1                /*所有结点个数的最大值*/
typedef struct
{   int weight;
    int parent;                  /*双亲的下标*/
    int LChild;                  /*左孩子的下标*/
    int RChild;                  /*右孩子的下标*/
}HTNode,HuffmanTree[M+1];        /*Huffman 树，0号
                                   单元未用* /
```

weight	parent	LChild	RChild
权值	双亲	左孩子	右孩子

4. 哈夫曼算法的实现—创建哈夫曼树

	1	2	3	4	5	6	7	8	9
F:	5	7	3	2	8	5	10	15	25

	weight	parent	LChild	RChild
1	5	7	0	0
2	7	8	0	0
3	3	6	0	0
4	2	6	0	0
5	8	8	0	0
6	5	7	4	3
7	10	9	1	6
8	15	9	2	5
9	25	0	7	8

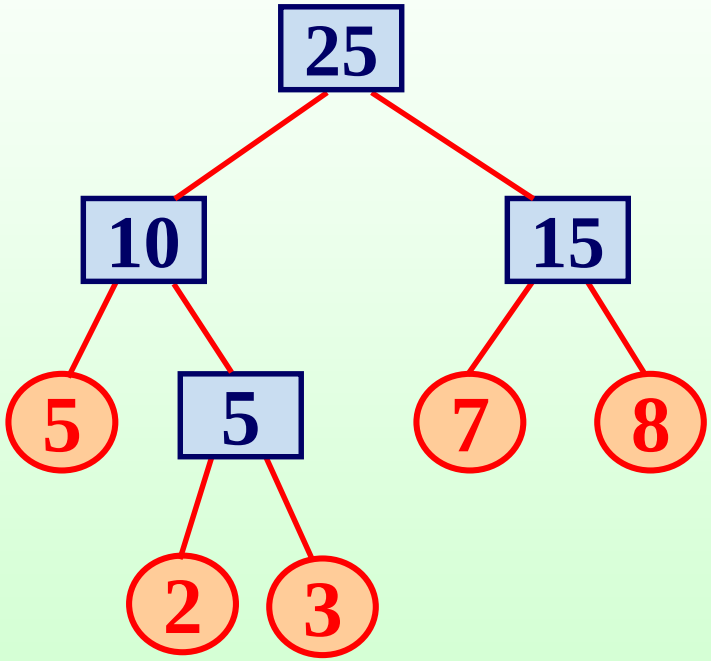
叶子结点

分支结点

s1

s2

i

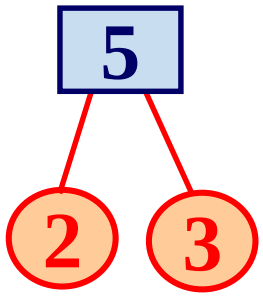


直至只含一棵树为止

```
void CrtHuffmanTree(HuffmanTree ht,int w[ ],int n)
{   int i, j ,k , s1 , s2;
    for(i=1;i<=n;i++)          /*初始化*/
    {   ht[i].weight=w[i];      ht[i].Parent=0;
        ht[i].LChild=0;         ht[i].RChild=0;
    }
    m=2*n-1;
    for(i=n+1;i<=m;i++)   /*初始化*/
    {   ht[i].weight=0;      ht[i].Parent=0;
        ht[i].LChild=0;         ht[i].RChild=0;
    }
     /*选择   合并*/
}
```

4. 哈夫曼算法的实现—创建哈夫曼树

```
/*选择、合并n-1次*/
for(i=n+1;i<=m;i++)
{   select (ht,i-1,&s1,&s2);
    ht[i].weight=ht[s1].weight+ht[s2].weight;
    ht[s1].parent=i;
    ht[s2].parent=i;
    ht[i].LChild=s1;
    ht[i].RChild=s2;
}
```



4. Huffman算法实现—选择函数

```
#define  MAXINT  32767
select(HuffmanTree ht, int pos, int *s1, int *s2 )
{  int  j , m1, m2;    /*m1存放最小权值,s1是m1在数组的下标*/
   m1=m2=MAXINT;  /*m2存放次小权值,s2是m2在数组的下标*/
   for(j=1;j<= pos;j++)
   {   if (ht[j].weight<m1 && ht[j].parent==0)
       {    m2 = m1;     *s2=*s1;          *s1=j;
            m1 = ht[j].weight;
       }
       else  if(ht[j].weight<m2 && ht[j].parent==0)
       {    m2 = ht[j].weight;
            *s2 = j;
       }
   }
}
```

5. Huffman coding (哈夫曼编码)

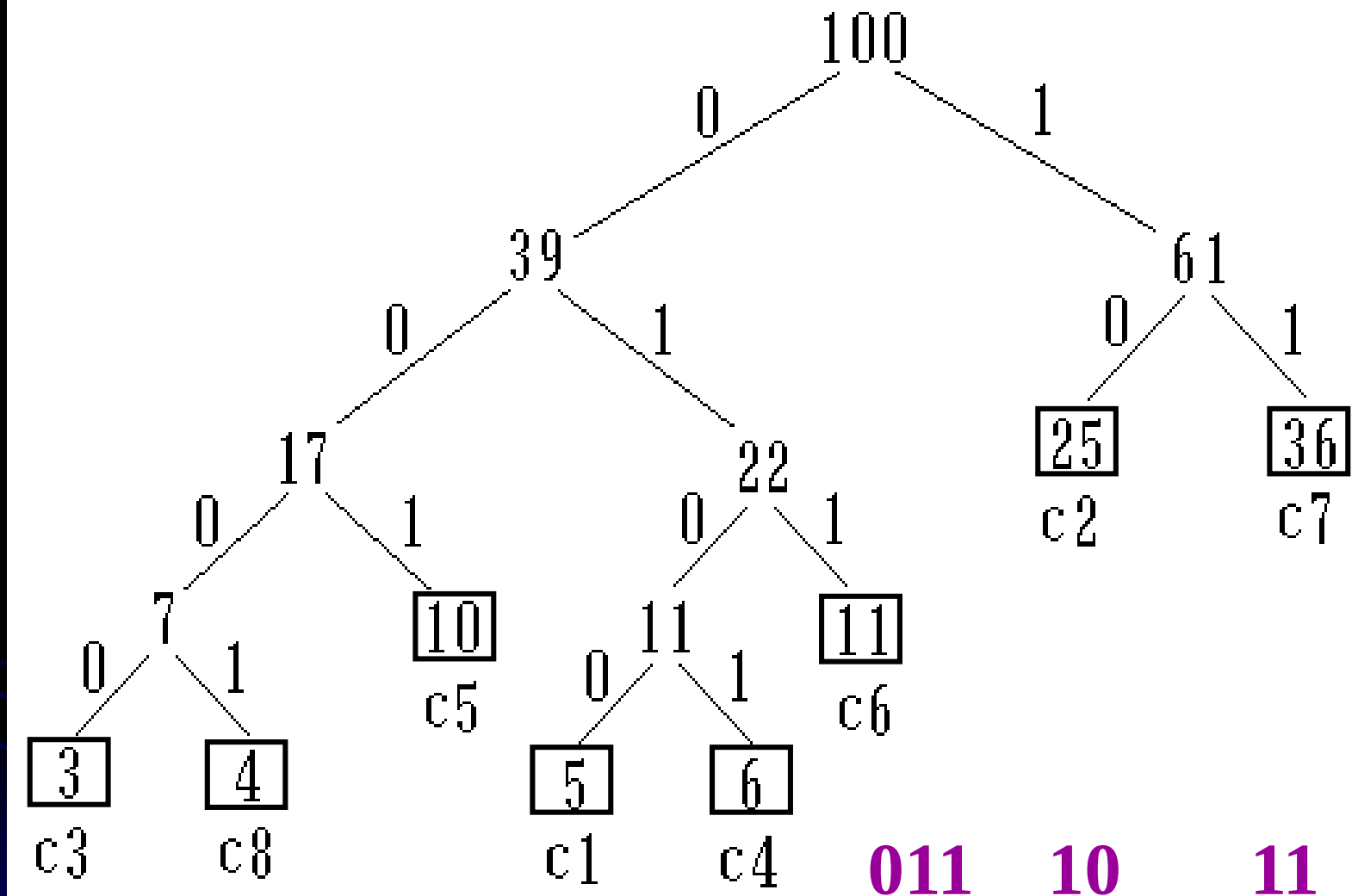
在一个字符集中，任何一个字符的编码**都不是**另一个字符编码的前缀，这种编码称为**前缀编码**。

我们可以利用二叉树来设计二进制的前缀编码。约定左分支表示字符'0'，右分支表示字符'1'，则可以用从根结点到叶子结点的路径上的分支字符串作为该叶子结点字符的编码。如此得到的编码必是*前缀编码*。

证明：

假设某一个叶子x结点的编码是另一个叶子结点y编码的前缀，说明从根结点到叶子结点y中间需经过结点x，从而说明x有左或右子树，这与x是叶子结点矛盾。

那么现在求最短的二进制编码实际上就是构造哈夫曼树的过程，由此得到的二进制编码，称为**哈夫曼编码**。



011 10 11

0000 0001 001 0100 0101

编解码举例

表 11.1.2　自然码和变长码编码例子

S_k　符号	$p_s(s_k)$	自然码	自然码 $l(s_k)$	变长码	变长码 $l(s_k)$
$r_0 = 0$　*a*	0.19	000	3	11	2
$r_1 = 1/7$　*b*	0.25	001	3	01	2
$r_2 = 2/7$　*c*	0.21	010	3	10	2
$r_3 = 3/7$　*d*	0.16	011	3	001	3
$r_4 = 4/7$　*e*	0.08	100	3	0001	4
$r_5 = 5/7$　*f*	0.06	101	3	00001	5
$r_6 = 6/7$　*g*	0.03	110	3	000001	6
$r_7 = 1$　*h*	0.02	111	3	000000	6

平均码长=(0.19+0.25+0.21)*2+0.16*3+0.08*4+0.06*5+0.05*6=2.7

- 解码端收到01 bit stream

定长编码（自然码）

a c f e d f h d h g

0000101011000111011111011111110

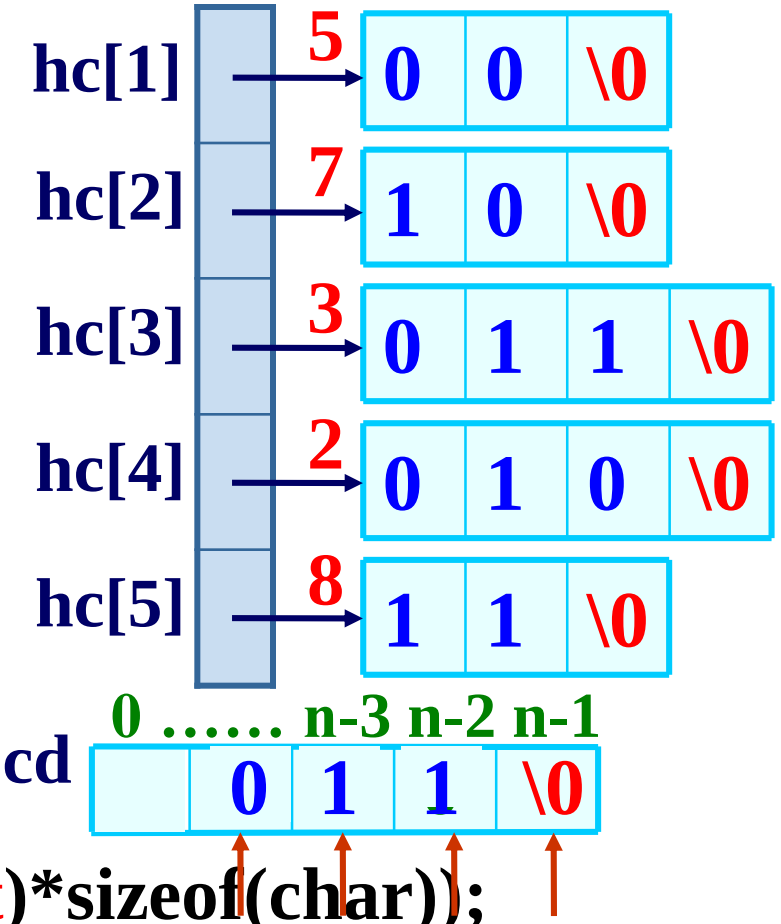
10个符号

变长编码（哈夫曼码）

f b b c d a b a c a a a c

13个符号

```
typedef char *HuffmanCode[n+1];          /*Huffman 编码*/
void CrtHuffmanCode(HuffmanTree ht,HuffmanCode hc,int n)
{   char *cd;   cd=(char *)malloc(n*sizeof(char));
    cd[n-1]='\0';
    for(i=1;i<=n;i++)
   {    start=n-1; c=i;
        p=ht[i].parent;
        while(p!=0)
       {   --start;
           if(ht[p].LChild==c)
               cd[start]='0';
           else
               cd[start]='1';
           c=p;  p=ht[p].parent;
       }
      hc[i]=(char *)malloc((n-start)*sizeof(char));
      strcpy(hc[i],&cd[start]);
    }
    free(cd);
}
```

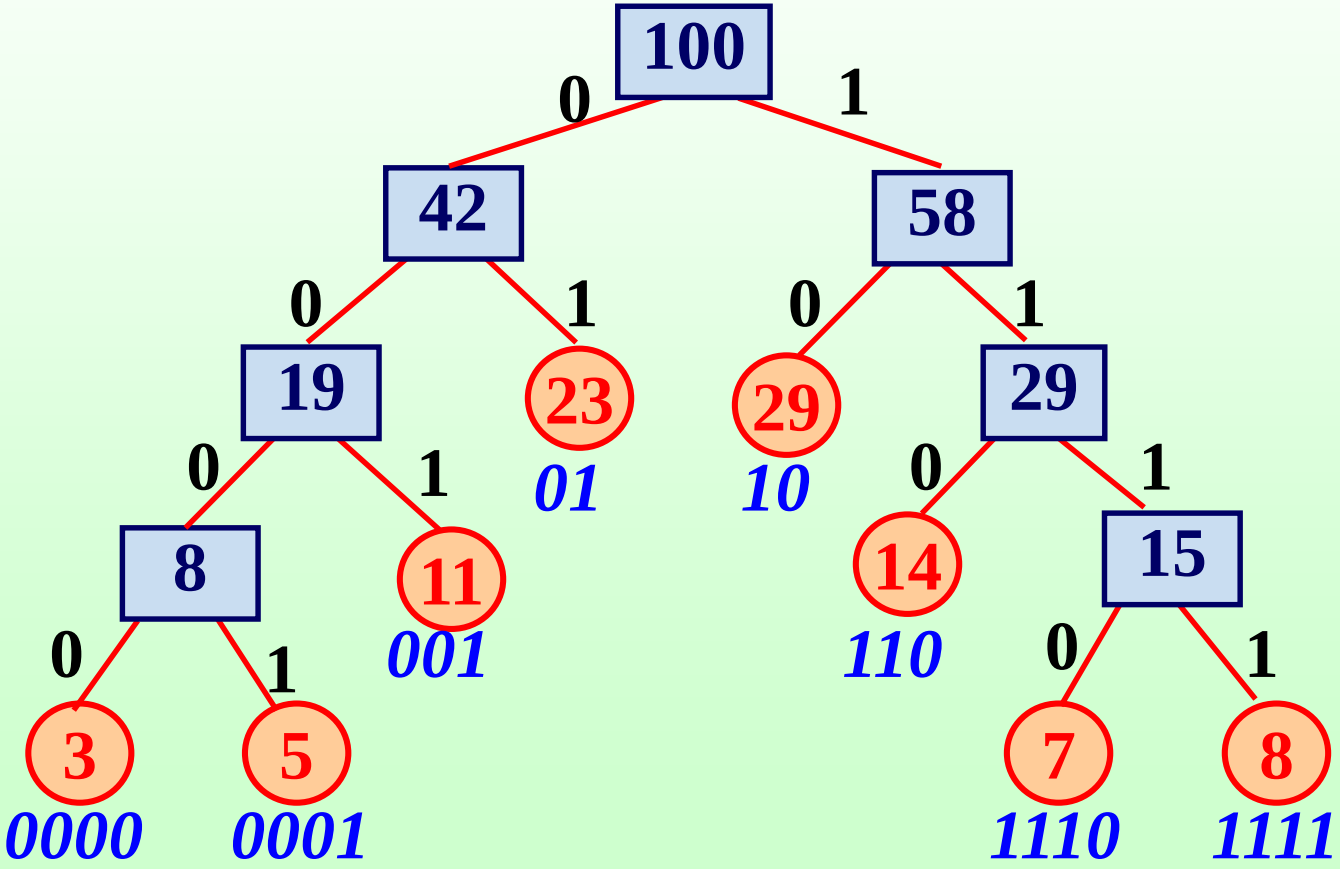


Huffman 译码

从哈夫曼树的根结点出发，逐个读入报文的二进制码，若读入0则走向左孩子，否则走向右孩子，一旦到达叶结点译出相应字符。然后重新从根结点出发继续译码，直到整个报文结束。

例如：假设某系统通信联络中只可能出现8种字符，其出现概率为0.05,0.29,0.07,0.08,0.14,0.23,0.03,0.11,试设计哈夫曼编码。

设w={5, 29, 7, 8, 14, 23, 3, 11 ,8 ,15 ,19 ,29 ,42 ,58 ,100}



End of 6.6

Conclusion

- Definition and notations of Tree and Forest (树和森林的定义)
- Notations and representation of binary tree (二叉树的概念及存储表示)
- Binary tree traversal (二叉树的遍历)
- Threading binary tree (线索二叉树)
- Reconstruction of binary tree (二叉树的构造)
- Transformation among Tree, Forest and binary tree (树、森林和二叉树的转换)
- Huffman tree and Huffman coding (哈夫曼树和哈夫曼编码)