



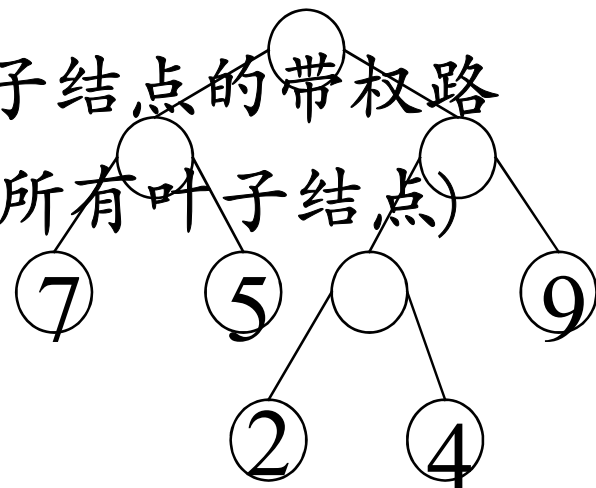
6.6 赫夫曼树及其应用

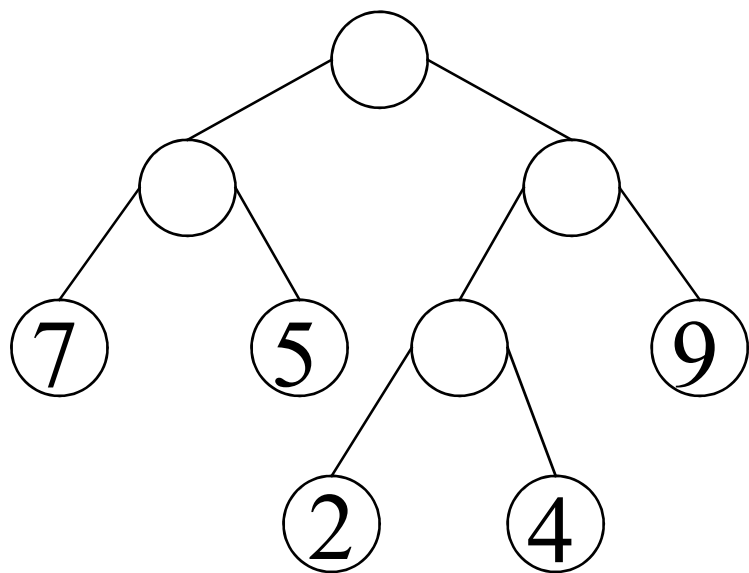
- 最优二叉树（**赫夫曼树**）的定义
- 如何构造最优二叉树
- **赫夫曼**编码
- **赫夫曼树**存储表示及构造算法



6.6.1 最优二叉树 (赫夫曼树)

- **结点的路径长度**: 从根结点到该结点的路径上分支的数目。
- **树的路径长度**: 树中每个结点的路径长度之和
- **结点的带权路径长度**: 从根结点到该结点的路径长度(l_k)与结点上权(w_k)的乘积。
- **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--- $WPL(T) = \sum w_k l_k$ (对所有叶子结点)



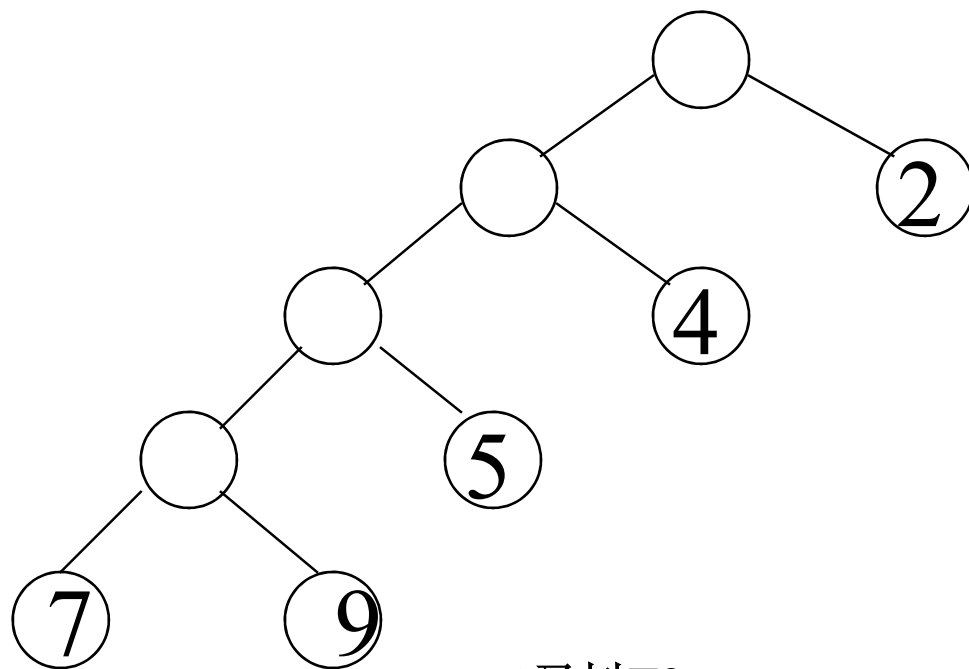


二叉树T1

WPL(T2)=

$$7 \times 2 + 5 \times 2 + 2 \times 3 + 4 \times 3$$

$$+ 9 \times 2 = 60$$



二叉树T2

WPL(T2)=

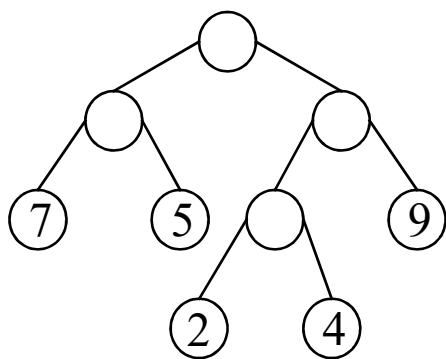
$$7 \times 4 + 9 \times 4 + 5 \times 3 + 4 \times 2$$

$$+ 2 \times 1 = 89$$

树的带权路径长度计算示例

6.6.1 最优二叉树 (赫夫曼树)

- 在所有含 n 个叶子结点、并且叶子结点带相同权值的二叉树中，必存在一棵其带权路径长度WPL最小的树，称为“最优二叉树(赫夫曼树)”。

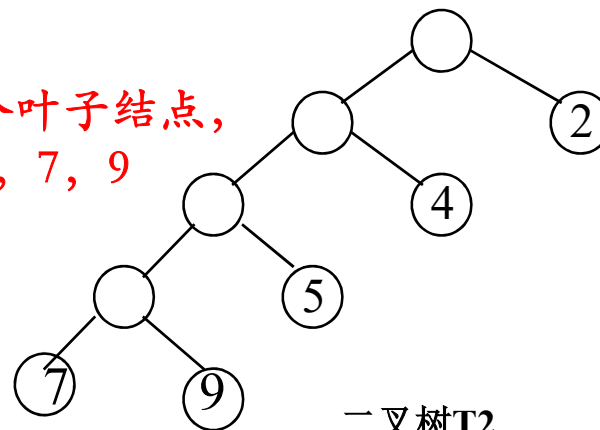


二叉树T1

T1是不是最优二叉树?

$$WPL(T) = 7 \times 2 + 5 \times 2 + 2 \times 3 + 4 \times 3 + 9 \times 2 = 60$$

这2棵二叉树均有5个叶子结点，
它们的权值2, 4, 5, 7, 9



二叉树T2

T2肯定不是最优二叉树

$$WPL(T) = 7 \times 4 + 9 \times 4 + 5 \times 3 + 4 \times 2 + 2 \times 1 = 89$$



n个结点的二叉树数 $h(n)$

穷举不可行，指数量级

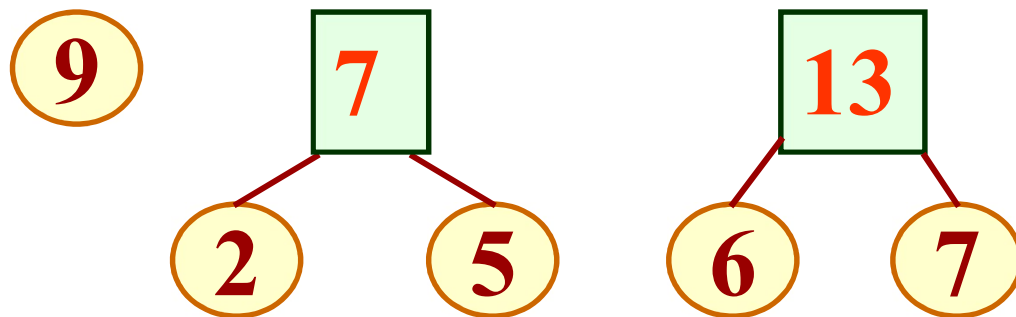
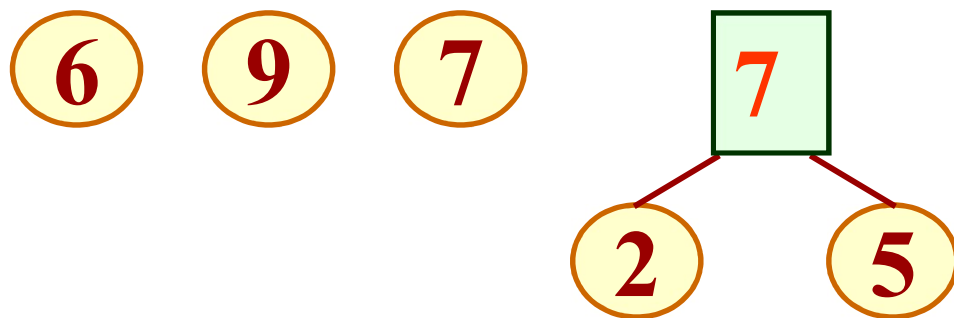
$$h(n) = \frac{1}{n+1} \frac{(2n)!}{n!n!} = \frac{1}{n+1} \frac{\sqrt{4\pi n} \left(\frac{2n}{e}\right)^{2n}}{\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \left(\frac{n}{e}\right)^n} = \frac{4^n}{(n+1)\sqrt{\pi n}} = O(4^n)$$

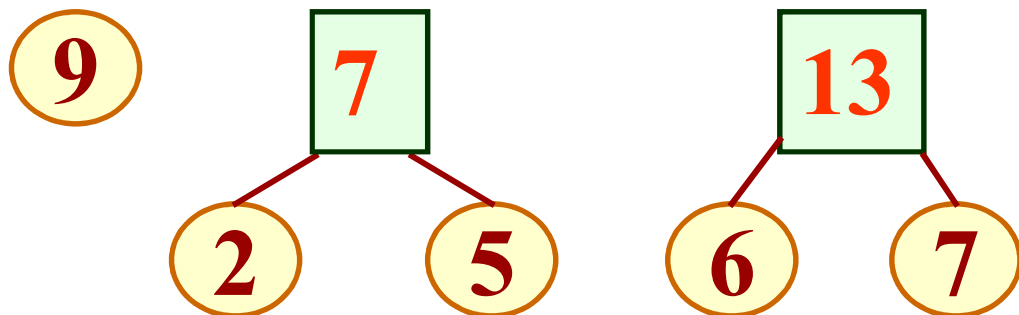
6.6.1 最优二叉树（赫夫曼树）

如何构造最优二叉树——赫夫曼算法

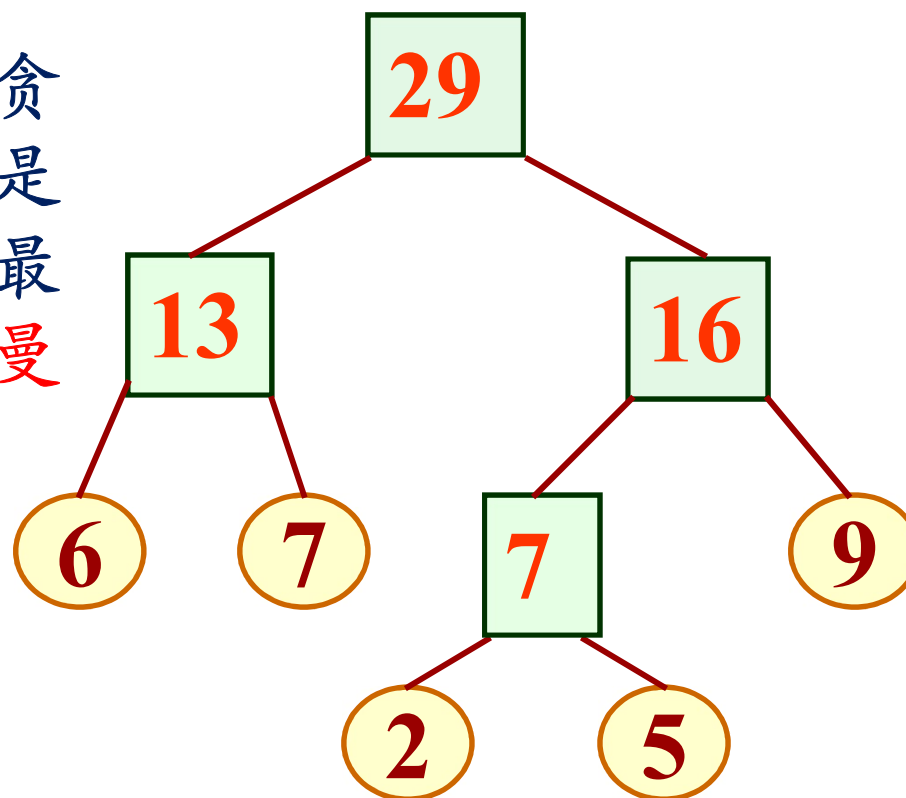
- 输入： n 和 n 个权值 $\{w_1, w_2, \dots, w_n\}$
 - 输出： 赫夫曼树
1. 根据给定的 n 个权值 $\{w_1, w_2, \dots, w_n\}$ ，构造 n 棵二叉树的集合 $F = \{T_1, T_2, \dots, T_n\}$ ，其中每棵二叉树中均只含一个带权值为 w_i 的根结点，其左、右子树为空树；
 2. 在 F 中选取其根结点的权值为最小和次小的两棵二叉树，分别作为左、右子树构造一棵新的二叉树，并置这棵新的二叉树根结点的权值为其左、右子树根结点的权值之和；
 3. 从 F 中删去这两棵树，同时加入刚生成的新树；
 4. 重复 2 和 3 两步，直至 F 中只含一棵树为止

例如：已知权值 $W=\{5, 6, 2, 9, 7\}$





赫夫曼算法采用的是贪心策略，贪心策略不是对所有问题都能得到最优解。可以证明赫夫曼算法得到的是最优解





6.6.2 赫夫曼编码

- 赫夫曼编码是赫夫曼树的一个应用。赫夫曼编码应用广泛，如 JPEG 中就应用了赫夫曼编码。
- 数据通信中，需要将传送的文字转换成 **二进制** 的字符串，用 0, 1 码的不同排列来表示字符。
- 例如，需传送的报文为 “AFTER DATA EAR ARE ART AREA”，这里用到的字符集为 “**A, E, R, T, F, D**”，各字母出现的次数为 {8, 4, 5, 3, 1, 1}。现要求为这些字母设计编码。
- 要区别 6 个字母，最简单的二进制编码方式是等长编码，固定采用 3 位二进制，可分别用 000、001、010、011、100、101 对 A, E, R, T, F, D 进行编码发送，当对方接收报文时再按照三位一分进行译码。
- **显然编码的长度取决报文中不同字符的个数。若报文中可能出现 26 个不同字符，则固定编码长度为 5。**



6.6.2 赫夫曼编码

- 然而，传送报文时总是希望总长度尽可能短。在实际应用中，各个字符的出现频度或使用次数是不相同的，如A、B、C的使用频率远远高于X、Y、Z，自然会想到设计编码时，让使用频率高的用短码，使用频率低的用长码，以优化整个报文编码。
- 同时 编码不能产生“二义性”
- 例如：a: 0, c: 1, t: 01
- 那么0101代表什么？
- 消除“二义性”，采用前缀编码



6.6.2 赫夫曼编码

- 前缀编码：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。
- 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。

例如: a, b, c, d, e 5个字符的出现频次为 5, 6, 2, 9, 7
求其编码?

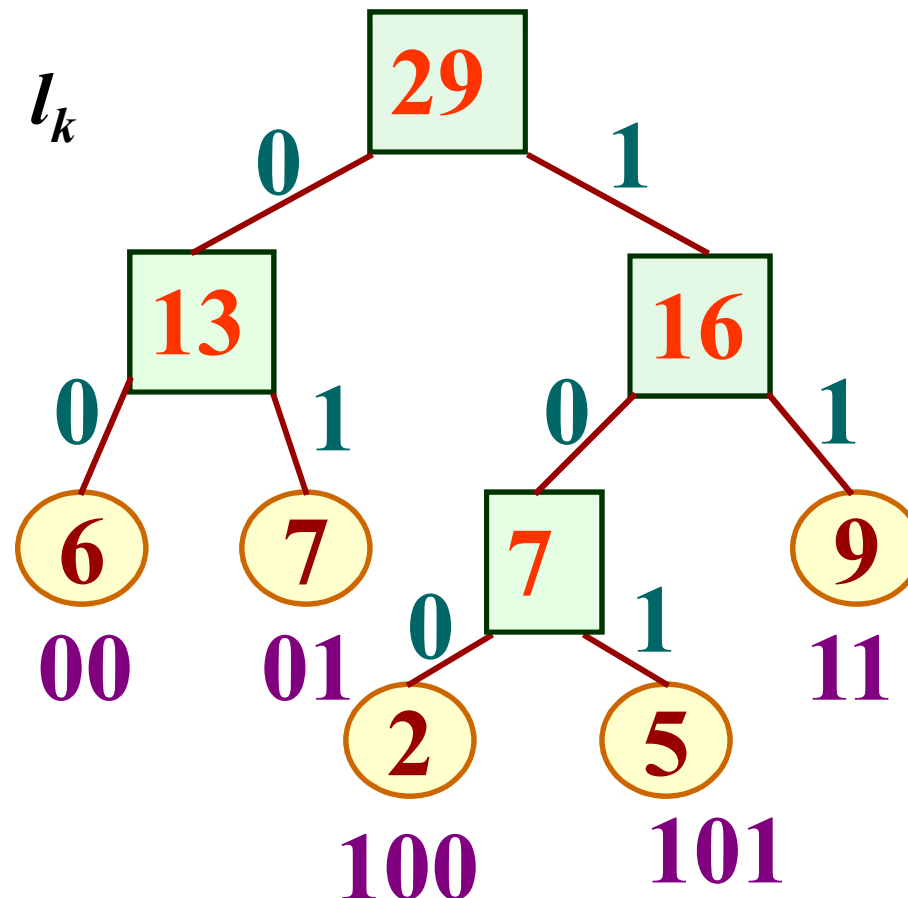
则: a --101, b --00, c --100, d --11, e --01

则: 码文长度= Σ (每个字符编码长度*出现频次)

每个字符编码长度: l_k

出现频次: w_k

利用哈夫曼树来设计二进制的
前缀编码, 既满足前缀编码的条件, 又
保证报文编码总长最短



赫夫曼编码

- 对需要编码的数据进行两遍扫描：
- 第一遍统计原数据中各字符出现的频率，利用得到的频率值创建赫夫曼树，并必须把树的信息保存起来，以便解压时创建同样的赫夫曼树进行解压；
- 第二遍则根据第一遍扫描得到的赫夫曼树进行编码，并把编码后得到的码字存储起来

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

	weight	parent	lchild	rchild
1				
2				
3				
4				
5				
6				
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

根据第一个权值生成第一棵只有根结点的二叉树

	weight	parent	lchild	rchild
1	5	0	0	0
2				
3				
4				
5				
6				
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

根据第二个权值生成第二棵只有根结点的二叉树

	weight	parent	lchild	rchild
1	5	0	0	0
2	6	0	0	0
3				
4				
5				
6				
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	0	0	0
2	6	0	0	0
3	2	0	0	0
4				
5				
6				
7				
8				
9				

根据第三个权值生成第三棵只有根结点的二叉树

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点

2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

根据第四个权值生成第四棵只有根结点的二叉树



	weight	parent	lchild	rchild
1	5	0	0	0
2	6	0	0	0
3	2	0	0	0
4	9	0	0	0
5				
6				
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点

2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

根据第五个权值生成第五棵只有根结点的二叉树

	weight	parent	lchild	rchild
1	5	0	0	0
2	6	0	0	0
3	2	0	0	0
4	9	0	0	0
5	7	0	0	0
6				
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
当前森林中根 结点权值次小 的二叉树	5	0	0	0
2	6	0	0	0
当前森林中根 结点权值最小 的二叉树	2	0	0	0
4	9	0	0	0
5	7	0	0	0
6				
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	0	0	0
3	2	6	0	0
4	9	0	0	0
5	7	0	0	0
6	7	0	3	1
7				
8				
9				

以最小和次小分别为左右子树，生成的二叉树

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

当前森林中有4棵二叉树

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	0	0	0
3	2	6	0	0
4	9	0	0	0
5	7	0	0	0
6	7	0	3	1
7				
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	0	0	0
3	2	6	0	0
4	9	0	0	0
5	7	0	0	0
6	7	0	3	1
7				
8				
9				

当前森林中根
结点权值最小
的二叉树

当前森林中根
结点权值次小
的二叉树

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点

2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	7	0	0
3	2	6	0	0
4	9	0	0	0
5	7	7	0	0
6	7	0	3	1
7	13	0	2	5
8				
9				

以最小和次小分别为左右子树，生成的二叉树

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

当前森林中有3棵二叉树

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	7	0	0
3	2	6	0	0
4	9	0	0	0
5	7	7	0	0
6	7	0	3	1
7	13	0	2	5
8				
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	7	0	0
3	2	6	0	0
4	9	0	0	0
5	7	7	0	0
6	7	0	3	1
7	13	0	2	5
8				
9				

当前森林中根
结点权值次小
的二叉树

当前森林中根
结点权值最小
的二叉树

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点

2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	7	0	0
3	2	6	0	0
4	9	8	0	0
5	7	7	0	0
6	7	8	3	1
7	13	0	2	5
8	16	0	6	4
9				

以最小和次小分别为左右子树，生成的二叉树

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

当前森林中有2棵二叉树,分别为最小和次小

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	7	0	0
3	2	6	0	0
4	9	8	0	0
5	7	7	0	0
6	7	8	3	1
7	13	0	2	5
8	16	0	6	4
9				

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

已知权值 $W=\{ 5, 6, 2, 9, 7 \}$

算法实现：借助数组构造赫夫曼树

因为树中的结点从下标为1的数组元素开始存放，所以0代表空指针

	weight	parent	lchild	rchild
1	5	6	0	0
2	6	7	0	0
3	2	6	0	0
4	9	8	0	0
5	7	7	0	0
6	7	8	3	1
7	13	9	2	5
8	16	9	6	4
9	29	0	7	8

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

给定 n 个权值，说明赫夫曼树有 n 个叶子结点
赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

存储表示及算法

给定 n 个权值，说明赫夫曼树有 n 个叶子结点

赫夫曼树只有叶子结点和度为2的结点，所以结点总数为 $2n-1$

1. 用 $2n-1$ 个数组元素存放赫夫曼树的每个结点
2. 每个结点存放权值，双亲指针（双亲所在数组元素的下标）和左孩子指针（所在数组元素的下标）、右孩子指针（所在数组元素的下标）

```
typedef struct {  
    int weight;  
    int parent,lchild,rchild;  
}HTNode, *HuffmanTree; //哈夫曼树  
Typedef char **HuffmanCode; //哈夫曼编码
```

```
void huffmancodeing(HuffmanTree &HT, HuffmanCode &HC, int *w,int n)
```

```
{ if n<1 return; m=2*n-1;
```

```
HT=(HuffmanTree)malloc((m+1)*sizeof(HTNode));
```



```
for (p=HT+1,i=1;i<=n;++i,++p,++w) *p={*w, 0, 0, 0};
```

```
for (;i<=m;++i,++p) *p={0,0,0,0};
```

```
for (i=n+1;i<=m;++i) {
```

```
    Select (HT, i-1,s1,s2);
```

已知权值 W={ 5, 6, 2, 9, 7 }

```
    HT[s1].parent=i; HT[s2].parent=i;
```

```
    HT[i].lchild=s1; HT[i].rchild=s2;
```

```
    HT[i].weight=HT[s1].weight+HT[s2].weight;}
weight parent lchild rchild
```

•
•
•
•
•
•

	weight	parent	lchild	rchild
1				
2				
3				
4				
5				
6				
7				
8				
9				