6.6 赫夫曼树及其应用

6.6.1 最优二叉树

结点的路径长度：

从树中一个结点到另一个结点之间的分支构成这两个结点之间的路径，路经上的分支数目称作路经长度。

树的路径长度：

从树根到每个结点的路径长度之和。

结点的带权路径长度：

从该结点到树根之间的路经长度与结点上权的乘积。

树的带权路径长度：

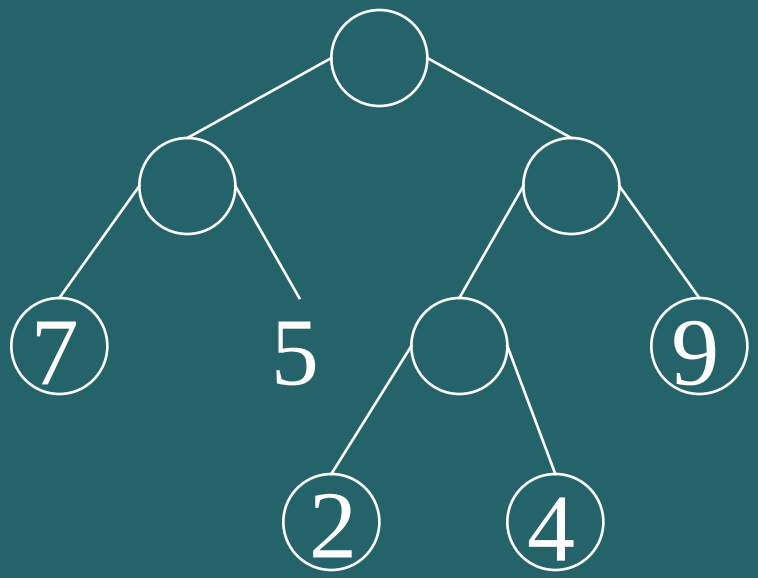
树中所有**叶子结点的带权路径长度**之和。记作：

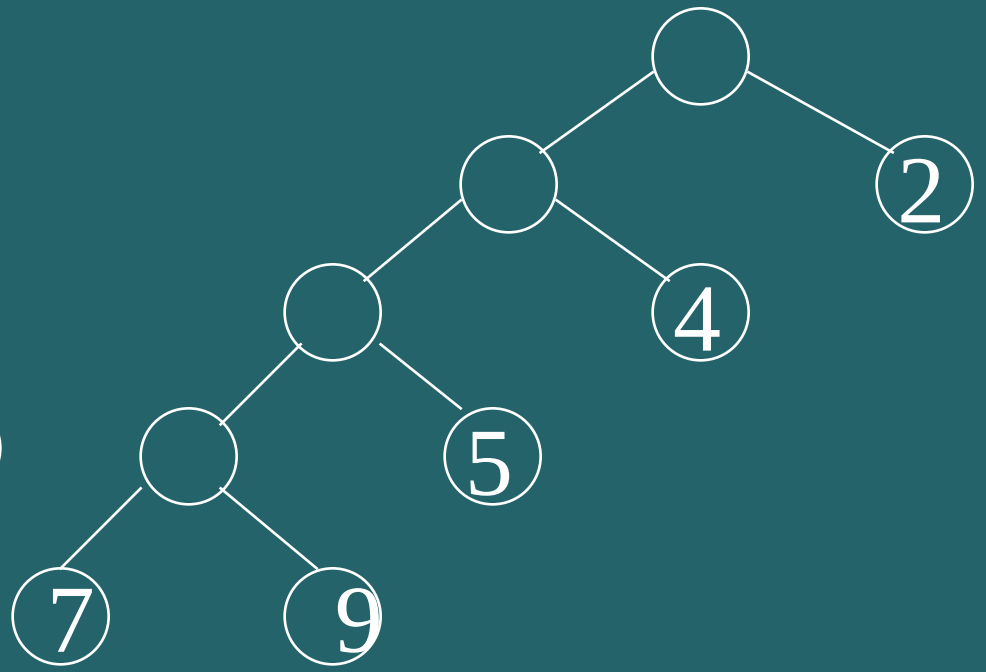
$WPL(T) = \Sigma w_k l_k$ (对所有叶子结点)。

其中：w_k，第k个叶结点的权值；

l_k，第k个叶结点到根结点的路径长度。

例如：





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

在所有含 *n* 个叶子结点、并带相同权值的 m 叉树中，必存在一棵其**带权路径长度取最小值**的树，称为“**最优树**”。

假设有n个权值$\{w_1,w_2,...w_n\}$，试构造一棵有 *n* 个叶子结点的二叉树，每个叶结点带权为w_i，则其中**带权路径长度WPL最小**的二叉树称为**最优二叉树**或**哈夫曼树**。

如何构造最优二叉树呢？哈夫曼最早给出了一个带有一般规律的算法，俗称哈夫曼算法。

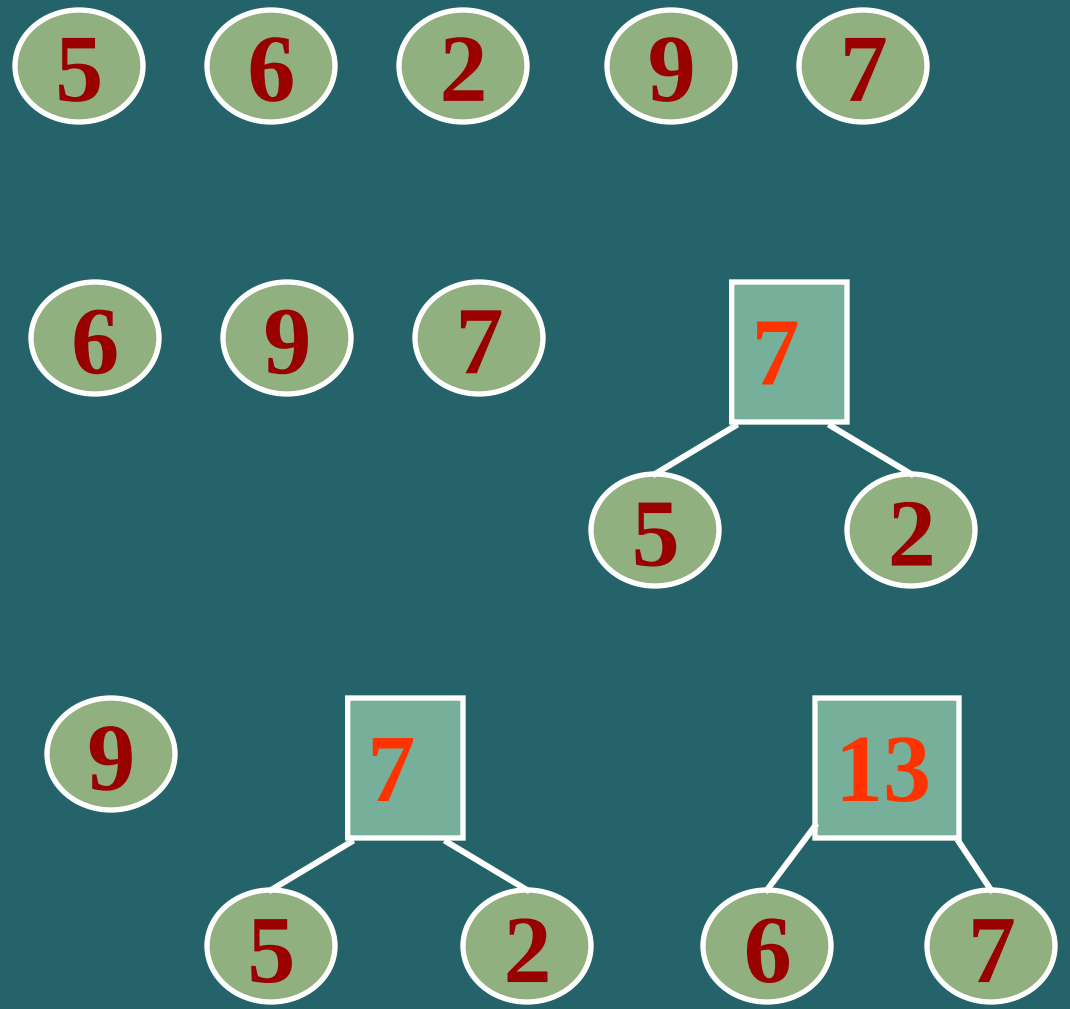
（1）根据给定的 n 个权值 $\{w_1, w_2, \ldots, w_n\}$， 构造 n 棵二叉树的集合 $F = \{T_1, T_2, \ldots, T_n\}$，其中每棵二叉树$T_i$中均只含一个带权值为 w_i 的根结点，其左、右子树为空树；

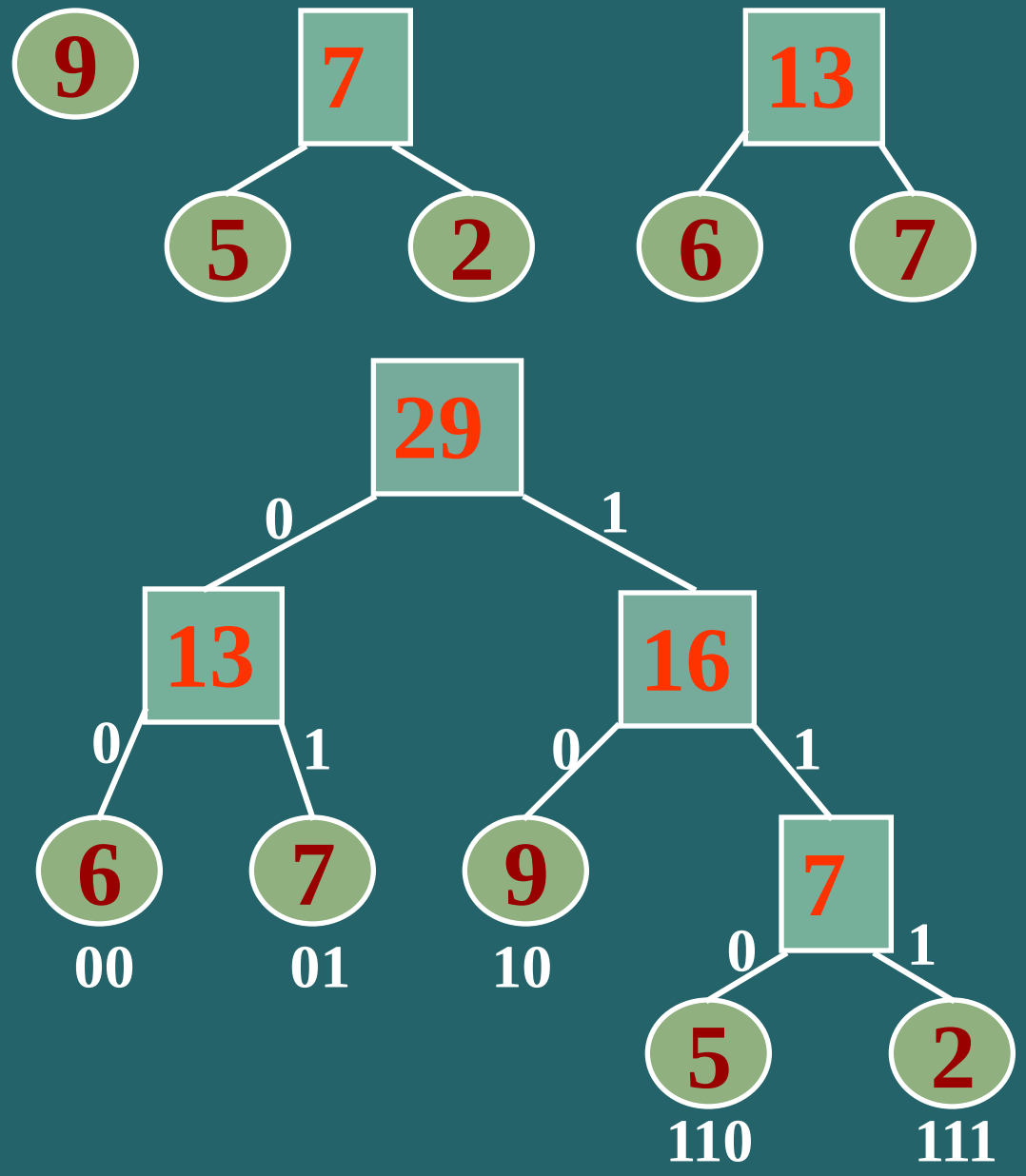
（2）在 F 中选取其根结点的权值为最小的两棵二叉树，分别作为左、右子树构造一棵新的二叉树，并置这棵新的二叉树根结点的权值为其左、右子树根结点的权值之和；

（3）从F 中删去这两棵树，同时加入刚生成的新二叉树；

（4）重复 **(2)** 和 **(3)** 两步，直至 F 中只含一棵树为止。这棵树便是哈夫曼树。

例如: 已知权值 W={ 5, 6, 2, 9, 7 }




9
7
5
2
13
6
7
29
0
1
13
16
0
1
0
1
6
7
9
7
00
01
10
0
1
5
2
110
111

6.6.2 哈夫曼编码

在传送电文时，希望总长度尽可能短。如果对每个字符设计长度不等的编码，且让电文中出现次数较多的字符采用尽可能短的编码，则传送电文的总长度便可减少。

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，这种编码称作前缀编码。

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**。

// - - - - 赫夫曼树和赫夫曼编码的存储表示 - - - -

```
typedef struct {
    unsigned int    weight ;
    unsigned int    parent, lchild, rchild ;
}HTNode, * HuffmanTree; //动态分配数组存储赫夫曼树

typedef char * * HuffmanCode;
                                //动态分配数组存储赫夫曼编码表
```

具体算法讲义P147-P148，算法6.12和6.13