### 实验9哈夫曼树的建立

1.实验目的

（1）掌握哈夫曼树概念和特点；

（2）熟练掌握建立哈夫曼树方法并能用算法实现；

（3）理解哈夫曼树的应用。

2.实验内容

（1）输入n个叶子及权值，建立huffman树并输出；

（2）对建立的huffman树各个叶子结点进行编码并输出。

3.程序分析

（1）由于叶子结点个数n已知，根据二叉树性质得知，整棵二叉树结点总数为（2n-1）个，故可建立长度为（2n-1）的数组。

（2）数组每个元素为二叉树的一个结点，由5个域组成，所以应采用结构体类型，包括结点值、权值、左孩子、右孩子、双亲，定义如下：

**class** Node {

**char** value;

**int** weight;

**int** lc;

**int** rc;

**int** parent;

}

（3）输入叶子个数及初始化：如叶子结点个数为5，数组长度就为9。

（4）输入各个叶子的值、权值，并对数组初步赋值。

如各个叶子值分别为A、B、C、D、E，各个权值分别为18，12，10，7，11。此部分信息由键盘一边输入，一边对数组初步赋值，结果如表9-1所示。

**表9-1** huffman数组初始化

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数组下标** | **结点值** | **权值** | **左孩子** | **右孩子** | **双亲** |
| 0 | A | 18 | -1 | -1 | -1 |
| 1 | B | 12 | -1 | -1 | -1 |
| 2 | C | 10 | -1 | -1 | -1 |
| 3 | D | 7 | -1 | -1 | -1 |
| 4 | E | 11 | -1 | -1 | -1 |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |

（5）计算二叉树其余结点信息。

由于有5个叶子结点，要合并4次。每次从无双亲（无双亲代表还未合并过）的结点中选择权值最小的两个结点进行合并，新结点下标为这两个的双亲，新结点的权值为这两个结点权值之和，左孩子为最小结点下标，右孩子为次小结点下标。

如表9-2所示为合并一次后的结果，如此循环4次即可：

**表9-2** huffman数组一次合并

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数组下标** | **结点值** | **权值** | **左孩子** | **右孩子** | **双亲** |
| 0 | A | 18 | -1 | -1 | -1 |
| 1 | B | 12 | -1 | -1 | -1 |
| 2 | C | 10 | -1 | -1 | 5 |
| 3 | D | 7 | -1 | -1 | 5 |
| 4 | E | 11 | -1 | -1 | -1 |
| 5 |  | 17 | 3 | 2 | -1 |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |

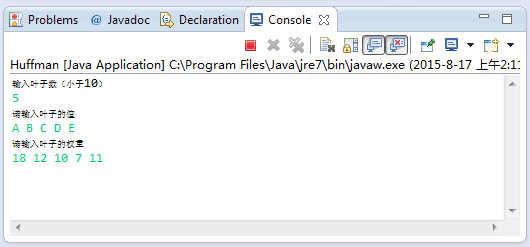
（6）输出二叉树（即输出数组），将二叉树按上表以表格形式在屏幕上输出即可。

（7）huffman编码

从叶子开始往上逆向编码，如某结点为其双亲的左孩子，编码0，否则编码1。输出编码与该编码顺序相反。

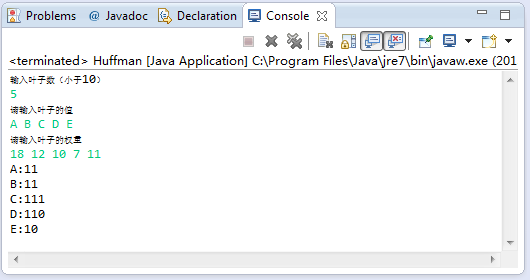
4.调试测试数据

（1）输入：按提示输入叶子结点的个数、叶子的值（字符型，输入不用空格隔开）、以及对应各个叶子的权值。如图9-1所示。



**图9-1** 输入

（2）输出：生成的huffman树各个结点的信息，以及每个结点的huffman编码。如图9-2所示。



**图9-2** 输出