

# 《数据结构》实验报告3

班级: 10012006

姓名: 夏卓

学号: 2020303245

E-mail: [2769223717@qq.com](mailto:2769223717@qq.com)

日期: 2022.4.30

## 3.1 哈夫曼编/译码器

### 实验内容:

#### 实验3.1: 哈夫曼编/译码器

Time Limit: 3000ms, Memory Limit: 10000KB, Accepted: 0, Total Submissions: 0

VIDEO1

#### Description

写一个哈夫曼码的编/译码系统, 要求能对要传输的报文进行编码和解码。构造哈夫曼树时, 权值小的放左子树, 权值大的放右子树, 编码时右子树编码为1, 左子树编码为0。

#### Input

输入表示字符集大小为 $n$  ( $n \leq 100$ ) 的正整数, 以及 $n$ 个字符和 $n$ 个权值 (正整数, 值越大表示该字符出现的概率越大); 输入串长小于或等于100的目标报文。

#### Output

经过编码后的二进制码, 占一行;  
以及对应解码后的报文, 占一行;  
最后输出一个回车符

#### Sample Input

```
5?a?b?c?d?e?12?40?15?8?25  
bbbaddecbbb
```

#### Sample Output

```
00011111110111010110110000  
bbbaddecbbb
```

### 一、需求分析:

#### 1. 输入:

输入表示字符集大小为 $n$  ( $n \leq 100$ ) 的正整数, 以及 $n$ 个字符和 $n$ 个权值 (正整数, 值越大表示该字符出现的概率越大), 输入串长小于或等于100的目标报文。

#### 2. 输出:

经过编码后的二进制码, 占一行;

以及对应解码后的报文, 占一行;

最后输出一个回车符

#### 3. 程序所能达到的功能:

写一个哈夫曼码的编/译码系统，要求能对要传输的报文进行编码和解码。构造哈夫曼树时，权值小的放左子树，权值大的放右子树，编码时右子树编码为1，左子树编码为0。

## 二、概要设计：

### 核心思想：

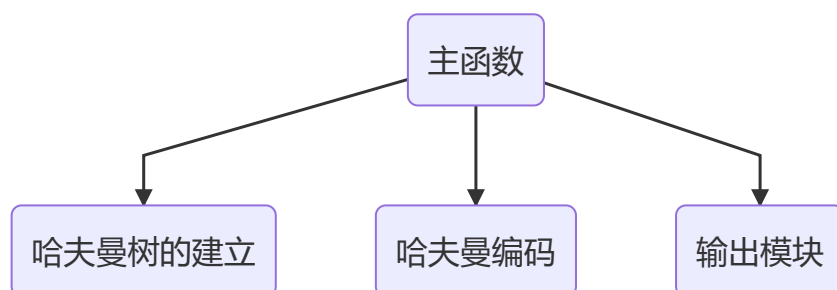
- 使用`map`对输入的 $n$ 个字符进行映射，以便对输入串进行解码
- 构造哈夫曼树对这 $n$ 个字符进行编码，编码时右子树编码为1，左子树编码为0
- 构造第 $i$ 个哈夫曼树节点的过程中需要选择前 $i - 1$ 个节点中权值最小的两个节点作为孩子节点，使用优先队列`priority_queue`的小根堆进行排序，以取得最小的两个权值
- 哈夫曼编码时，采用自底向上的搜索方式，通过节点的`parent`值找到父节点，然后判断该节点是父节点的左孩子还是右孩子，从而进行标记，编码值保存在`string`数组中
- 哈夫曼译码时，每次对所译码的二进制串进行子串匹配，可以使用`string`中的`find`函数进行子串匹配，若匹配的初始位置是0，则说明是该编码值对应的字符，接着将二进制串删去匹配成功部分，直到将二进制串全部删除完毕

### 程序框架：

本程序包含五个模块：

1. 主程序模块；
2. 哈夫曼树的建立模块；
3. 哈夫曼编码模块；
4. 输出模块

### 模块调用图：



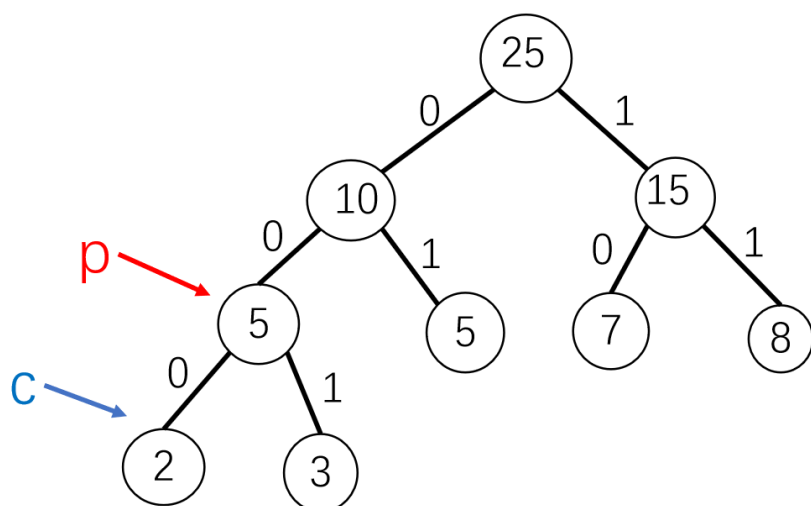
## 三、详细设计：

### 算法实现图：

以权值为{5, 2, 7, 3, 8}的节点构造哈夫曼树为例，其示意图应为

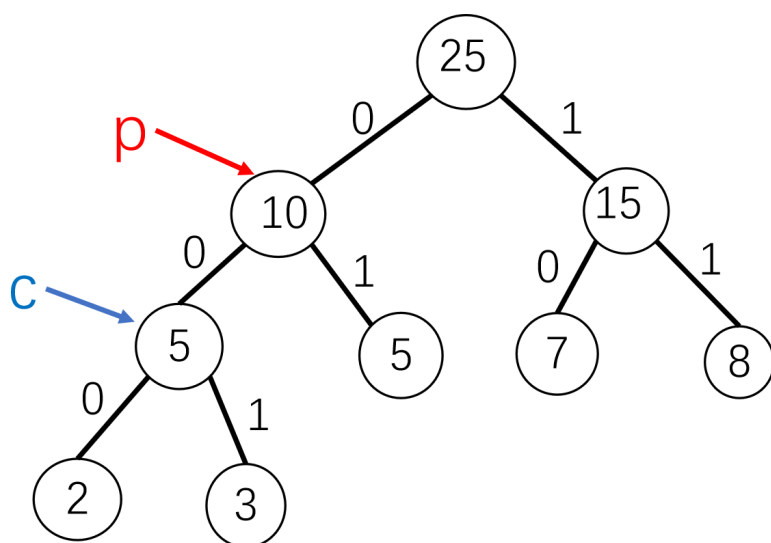


求哈夫曼编码的执行视图：



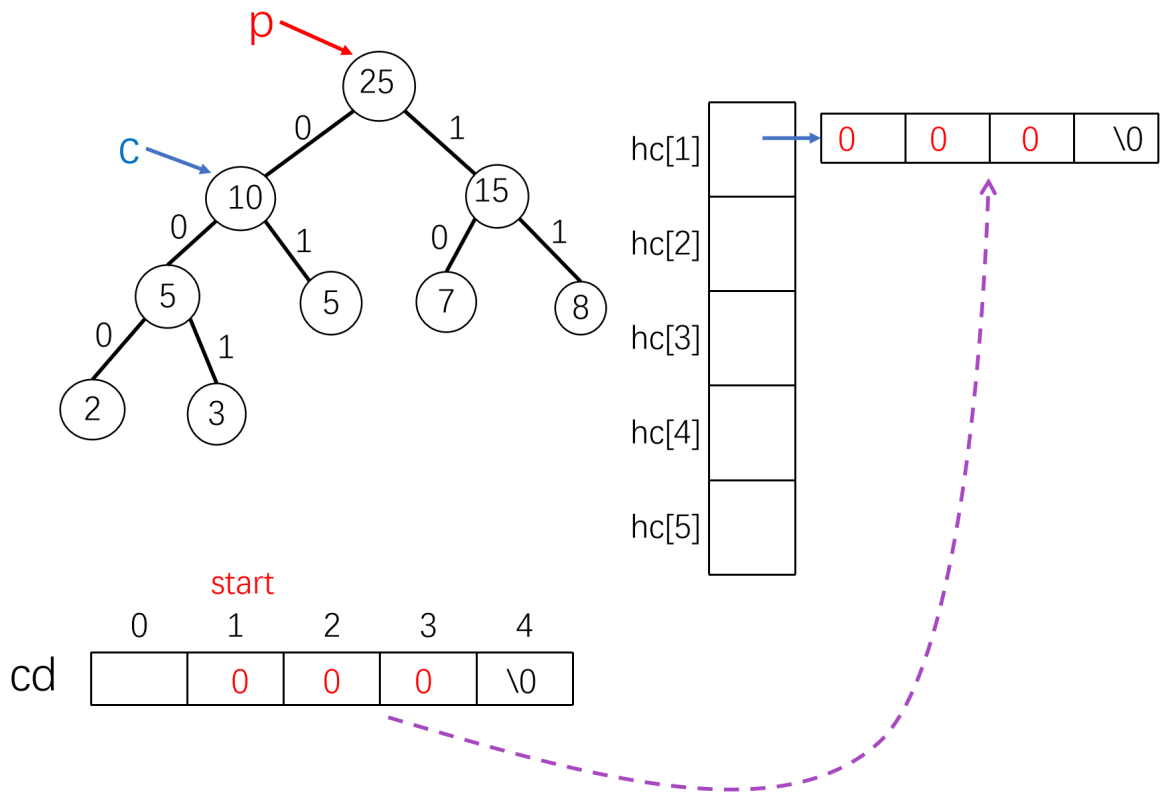
hc[1]	
hc[2]	
hc[3]	
hc[4]	
hc[5]	

	0	1	2	3	4
cd			0	0	\0



hc[1]	
hc[2]	
hc[3]	
hc[4]	
hc[5]	

	0	1	2	3	4
cd		0	0	0	\0



## 核心算法的伪代码框架：

```

1  const int N = 110;
2  const int M = 2 * N - 1;
3  typedef pair<int, int> PII;
4
5  //数据结构(静态三叉链表)
6  typedef struct {
7      int weight;
8      int parent, LChild, RChild;
9  }HTNode, HuffmanTree[M + 1];
10
11 //核心算法
12 //建立哈夫曼树:
13 void CreatHuffmanTree(HuffmanTree ht, int w[], int n) {
14     priority_queue<PII, vector<PII>, greater<PII>> heap; //用小根堆维持最小
    权值
15     //初始化静态三叉链表
16     for (int i = 1; i <= n; i++){
17         ht[i] = { w[i], 0, 0, 0 };
18         heap.push({ ht[i].weight, i });
19     } //1到n号单元存放叶子节点
20     int m = 2 * n - 1;
21     for (int i = n + 1; i <= m; i++)
22         ht[i] = { 0, 0, 0, 0 }; //n+1号到m号存放非叶子节点
23     for (int i = n + 1; i <= m; i++) {
24         int s1, s2;
25         s1 = heap.top().second; heap.pop();
26         s2 = heap.top().second; heap.pop(); //在前i-1个节点中取得权值最小的两个节
    点
27         ht[i].weight = ht[s1].weight + ht[s2].weight;
28         ht[i].LChild = s1; ht[i].RChild = s2;
29         ht[s1].parent = i; ht[s2].parent = i;
30         heap.push({ ht[i].weight, i }); //建立第i个节点, 其权值为两个最小权值之和
    
```

```

31     }
32 }
33
34 //哈夫曼编码:
35 void CreatHuffmanCode(HuffmanTree ht, string hc[], int n) {
36     for (int i = 1; i <= n; i++){
37         string str;
38         //自底向上遍历, 进行哈夫曼编码
39         for (int c = i, p = ht[i].parent; p != 0; c = p, p = ht[c].parent){
40             if (ht[p].LChild == c)
41                 str += '0';    //左子树编码为0
42             else str += '1';    //右子树编码为1
43         }
44         for (int j = str.size() - 1; j >= 0; j--)
45             hc[i] += str[j];    //将字符串反转保存
46     }
47 }
48
49 //译码器:
50 void decoder(string decoded_str, string hc[]) {
51     while (decoded_str.size() != 0) {
52         int i;
53         //找到二进制串中首子串对应的字符
54         for (i = 1; i <= n; i++)
55             if (decoded_str.find(hc[i]) == 0)
56                 break;
57         cout << ch[i];    //输出对应的字符
58         decoded_str.erase(0, hc[i].size()); //删去对应的二进制子串
59     }
60     return 0;
61 }

```

## 四、使用说明、测试分析及结果:

### 1. 说明如何使用你编写的程序

本程序的运行环境为visual studio 2019。

第一行输入表示字符集大小的正整数 $n$  ( $n \leq 100$ ), 以及 $n$ 个字符和 $n$ 个权值(正整数, 值越大表示该字符出现的概率越大); 第二行输入串长小于或等于100的目标报文。

程序会输出经过哈夫曼编码后的二进制码, 以及对应解码后的报文, 最后输出一个回车符


### 2. 测试结果与分析

本程序较好的完成了实验需求, 经分析, 编码的时间复杂度为 $O(n \log(n))$ , 其中 $n$ 为字符的个数

### 3. 调试过程中遇到的问题及解决方法

在处理如何将输入的字符串与每一个得到的哈夫曼编码进行匹配时遇到了困难，一开始想着暴力循环匹配，但觉得时间复杂度会变高，因此选用了`map`容器进行字符与二进制码的映射；另外，在取前 $i - 1$ 个节点中最小的两个权值时也遇到了时间复杂度较高的问题，于是使用了小根堆进行最小值的维护；最后，在译码上，没有想到很好的优化方法，只能循环匹配首字符串，进行解码

### 4. 运行界面

 Microsoft Visual Studio 调试控制台

```
5 a b c d e 12 40 15 8 25
bbbaddecbb
0001111110111010110110000
bbbaddecbb
D:\Code\VS_code\homework\哈夫曼编码\Debug\哈夫曼编码.
按任意键关闭此窗口. . .
```

## 五、实验总结

- 本实验我在编程中用时23分钟
- 在调试中用时12分钟，主要是一开始把译码的字符串搞混了，以至于运行错误
- 要记忆哈夫曼树的存储结构以及计算方法，对于 $n$ 个字符的报文，需要 $2n - 1$ 个节点进行存储，并且从第 $n + 1$ 个节点开始，第 $i$ 个节点的权值是前 $i - 1$ 个节点中最小的两个权值之和
- 编码主要靠自底向上的遍历实现，译码主要通过字符串匹配实现