数据结构与算法实验

实验十：哈夫曼树

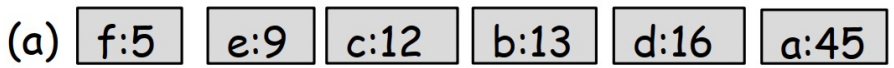
彭振辉
中山大学人工智能学院
2023年秋季学期

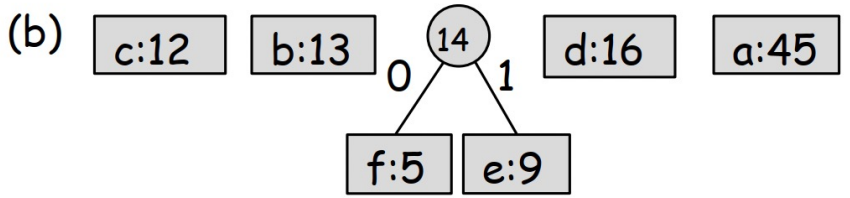
实验目的和要求

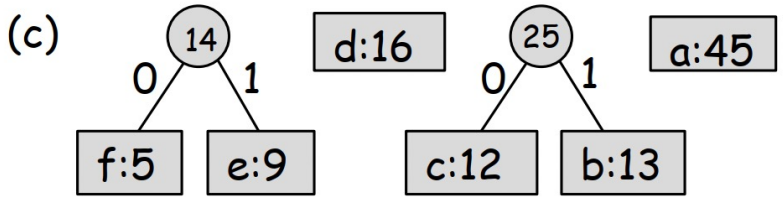
- 复习二叉树的逻辑结构和存储结构；
- 熟练掌握哈夫曼树的生成算法；
- 熟练掌握哈夫曼编码的方法。

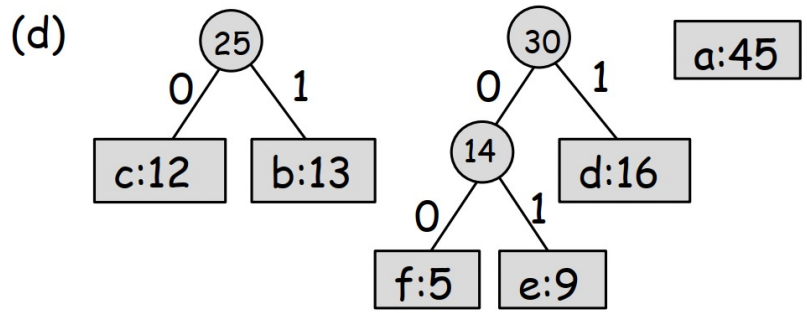
- 哈夫曼算法的实现以库文件方式实现，不得在测试主程序中直接实现；
- 程序有较好可读性，各运算和变量的命名直观易懂，符合关键工程要求；
- 程序有适当的注释。

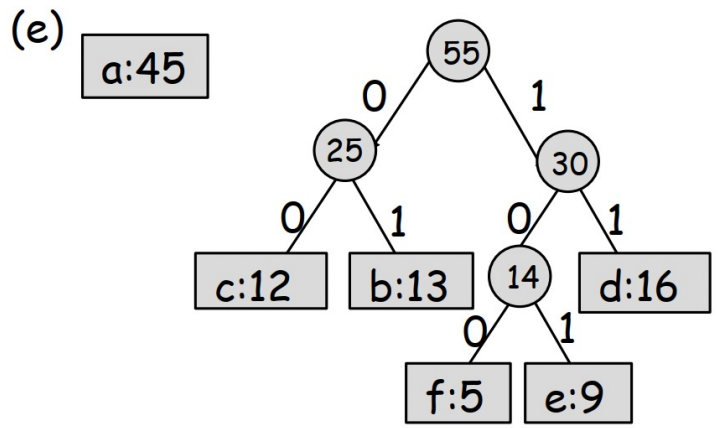
回顾：哈夫曼树构造例子

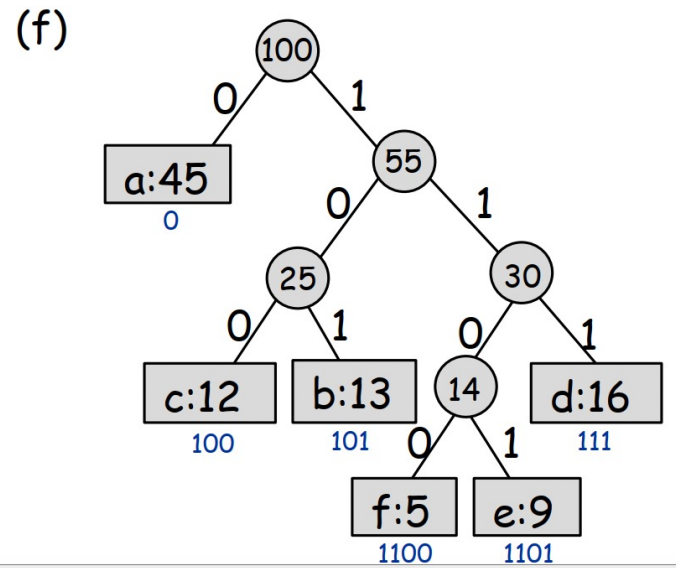

(a)
f:5
e:9
c:12
b:13
d:16
a:45


(b)
c:12
b:13
14
0
1
d:16
a:45
f:5
e:9


(c)
14
0
1
f:5
e:9
d:16
25
0
1
c:12
b:13
a:45


(d)
25
0
1
c:12
b:13
30
0
1
14
d:16
0
1
f:5
e:9
a:45


(e)
a:45
55
0
1
25
30
0
1
0
1
c:12
b:13
14
d:16
0
1
f:5
e:9


(f)
100
0
1
a:45
0
55
0
1
25
30
0
1
0
1
c:12
100
b:13
101
14
d:16
111
0
1
f:5
1100
e:9
1101

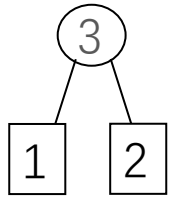
伪代码

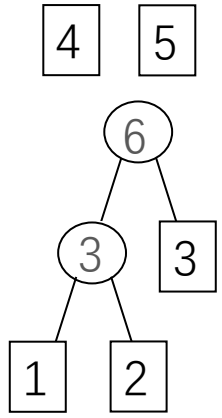
```
Huffman(A):
create a min-priority queue Q on A, with weight as key
for i ← 1 to n − 1
    allocate a new node z
    x ← Extract-Min(Q)
    y ← Extract-Min(Q)
    z.left ← x
    z.right ← y
    z.weight ← x.weight + y.weight
    Insert(Q, z)
return Extract-Min(Q) // return the root of the tree
```

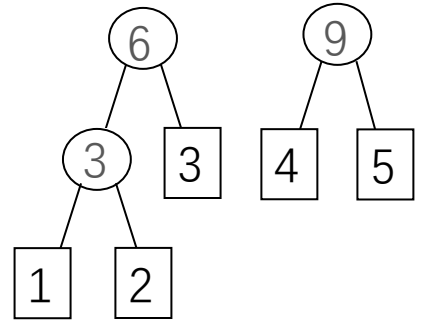
Running time: $O(n \log n)$

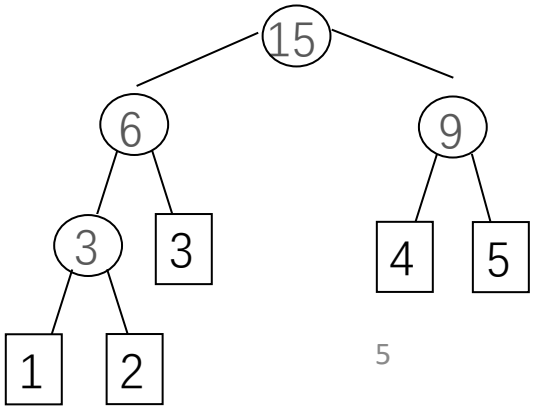
```
typedef struct TreeNode *HuffmanTree;
struct TreeNode{
    int Weight;
    HuffmanTree Left, Right;
}
HuffmanTree Huffman( MinHeap H )
{   /* 假设H->Size个权值已经存在H->Elements[]->Weight里 */
    int i;  HuffmanTree T;
    BuildMinHeap(H); /*将H->Elements[]按权值调整为最小堆*/
    for (i = 1; i < H->Size; i++) { /*做H->Size-1次合并*/
        T = malloc( sizeof( struct TreeNode) ); /*建立新结点*/
        T->Left = DeleteMin(H);
                  /*从最小堆中删除一个结点，作为新T的左子结点*/
        T->Right = DeleteMin(H);
                  /*从最小堆中删除一个结点，作为新T的右子结点*/
        T->Weight = T->Left->Weight+T->Right->Weight;/*计算新权值*/
        Insert( H, T ); /*将新T插入最小堆*/
    }
    T = DeleteMin(H);
    return T;
}
```

整体复杂度为：O(N logN)









思考1：为什么用到了最小堆？

思考2：在for的第一次循环过程中，各变量(如 H, T->Left, T->Right, T->Weight)在右边的这个例子中具体分别是什么？

实验任务1

- **已知每一个字符出现的频次，构造哈夫曼树**
  - **从终端读入N个字符和相应权重，建立哈夫曼树**
  - **对从终端读入的字符串进行编码，并显示编码结果**

注意：输出的哈夫曼编码不唯一，可以按任意字符顺序进行输出。请先自行验证哈夫曼编码是否正确。

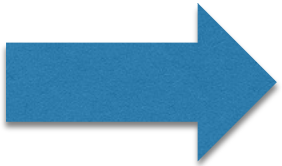
示例：

请输入要编码的字符,并以空格隔开（个数任意）：

A B C D E F G H I G

请输入每个字符对应的频次，并以空格隔开：

19 23 25 18 12 67 23 9 32 33



```
对应的Huffman编码如下:
G: 111
C: 110
I: 101
G: 100
F: 01
D: 0011
A: 0010
H: 00011
E: 00010
B: 0000
请按任意键继续. . .
```

实验任务2

- 给定一段文字，若统计出字母出现的频率，可以根据哈夫曼算法给出一套编码，使得用此编码压缩原文可以得到最短的编码总长。然而哈夫曼编码并不是唯一的。例如对字符串"aaaxuaxz"，容易得到字母 'a'、'x'、'u'、'z' 的出现频率对应为 4、2、1、1。我们可以设计编码 {'a'=0, 'x'=10, 'u'=110, 'z'=111}，也可以用另一套 {'a'=1, 'x'=01, 'u'=001, 'z'=000}，还可以用 {'a'=0, 'x'=11, 'u'=100, 'z'=101}，三套编码都可以把原文压缩到 14 个字节。但是 {'a'=0, 'x'=01, 'u'=011, 'z'=001} 就不是哈夫曼编码，因为用这套编码压缩得到 00001011001001 后，解码的结果不唯一，"aaaxuaxz" 和 "aazuaxax" 都可以对应解码的结果。**本题就请你判断任一套编码是否哈夫曼编码**

实验任务2

输入格式：

- 首先第一行给出一个正整数N(2≤N≤63)，随后第二行给出 N 个不重复的字符及其出现频率，格式为：

  c[1] f[1] c[2] f[2] … c[N] f[N]

  其中c[i]是集合{ ‘0’ - ‘9’, a-z, A-Z, _ }中的字符；f[i]是c[i]的出现频率，为不超过1000的整数。再下一行给出一个正整数M(≤1000)，随后是M套待检的编码。每套编码占N行，格式为：

  c[i] code[i]

  其中c[i]是第i个字符；code[i]是不超过63个 '0’ 和 ‘1’ 的非空字符串

输出格式：

- 对每套待检编码，如果是正确的哈夫曼编码，就在一行中输出"Yes"，否则输出"No"。注意：最优编码并不一定通过哈夫曼算法得到。任何能压缩到最优长度的前缀编码都应被判为正确

示例：

Input:

```
7
A 1 B 1 C 1 D 3 E 3 F 6 G 6
4
A 00000
B 00001
C 0001
D 001
E 01
F 10
G 11
A 01010
B 01011
C 0100
D 011
E 10
F 11
G 00
A 000
B 001
C 010
D 011
E 100
F 101
G 110
A 00000
B 00001
C 0001
D 001
E 00
F 10
G 11
```

给出一个正整数N

N 个不重复的字符及其出现频率

M套待检的编码

每套编码占N行

每套编码占N行

每套编码占N行

Output:

```
7
A 1 B 1 C 1 D 3 E 3 F 6 G 6
4
A 00000
B 00001
C 0001
D 001
E 01
F 10
G 11
Yes
A 01010
B 01011
C 0100
D 011
E 10
F 11
G 00
Yes
A 000
B 001
C 010
D 011
E 100
F 101
G 110
No
A 00000
B 00001
C 0001
D 001
E 00
F 10
G 11
No

--------------------------------
Process exited after 14.83 seconds with return valu
请按任意键继续. . .
```

对每套待检编码，如果是正确的哈夫曼编码，就在一行中输出"Yes"，否则输出"No"。注意：最优编码不一定通过哈夫曼算法得到。任何能压缩到最优长度的前缀编码都应被判为正确

实验任务3

- 从文件中读入任意一篇英文文本文件，分别统计英文文本文件中各字符(包括标点符号和空格)的使用频率；
- 根据已统计的字符使用频率构造哈夫曼编码树，并给出每个字符的哈夫曼编码(字符集的哈夫曼编码表)；
- 将文本文件利用哈夫曼树进行编码，存储成压缩文件(哈夫曼编码文件)；
- 计算哈夫曼编码文件的压缩率(e.g., chnum*1.0/8/total*100，其中total是要编码字符总数，每个字符占八位二进制，chnum是编码所有字符得到的二进制位数)；
- 将哈夫曼编码文件译码为文本文件，并与原文件进行比较；
- 利用堆结构，优化的哈夫曼编码算法。

拓展(难)：实验任务3

读取的示例文件 original.txt

original - 记事本

文件 编辑 查看

Hello,I am SYSU AIer!
I come from China.
Now I am a student, and I major in Artificial Intelligence.
#

输出的字符频率和编码

```
frequency of H is 0.009901
frequency of e is 0.069307
frequency of l is 0.049505
frequency of o is 0.049505
frequency of , is 0.019802
frequency of I is 0.059406
frequency of   is 0.158416
frequency of a is 0.069307
frequency of m is 0.049505
frequency of S is 0.019802
frequency of Y is 0.009901
frequency of U is 0.009901
frequency of A is 0.019802
frequency of r is 0.039604
frequency of ! is 0.009901
frequency of
 is 0.029703
frequency of c is 0.029703
frequency of f is 0.019802
frequency of C is 0.009901
frequency of h is 0.009901
frequency of i is 0.059406
frequency of n is 0.059406
frequency of . is 0.019802
frequency of N is 0.009901
frequency of w is 0.009901
frequency of s is 0.009901
frequency of t is 0.039604
frequency of u is 0.009901
frequency of d is 0.019802
frequency of j is 0.009901
frequency of g is 0.009901
H: 1111100
```

```
frequency of j is 0.009901
frequency of g is 0.009901
H: 1111100
e: 1001
l: 0010
o: 0011
,: 111000
I: 0101
 : 110
a: 1010
m: 0100
S: 111001
Y: 1111101
U: 1111110
A: 111010
r: 10110
!: 1111111

: 10000
c: 10001
f: 111011
C: 000000
h: 000001
i: 0110
n: 0111
.: 111100
N: 000010
w: 000011
s: 000100
t: 10111
u: 000101
d: 111101
j: 000110
g: 000111
Compression rate is 56.188119%
--------------------------------

Process exited after 0.08915 seconds with return value 0
请按任意键继续. . .
```

输出的编码文件和解码文件

compressed - 记事本

文件 编辑 查看

11111001001001000100011111000010111010100100110111001111110

decoding - 记事本

文件 编辑 查看

Hello,I am SYSU AIer!
I come from China.
Now I am a student, and I major in Artificial Intelligence.