2020 年软件学院上机测试题目

1.考试要求

1) 使用 C、Java、Python 等语言编写程序完成题目要求。
2) 你的所有源代码应保存在名为 SE_*****(注:此处为个人学号)_**(注:此处为姓名)的文件夹中。注意在考试过程中随时保存你的文件。**考试结束前**将你的源代码保存在 U 盘中。考试结束后等待监考老师收取 U 盘。
3) 请在代码注释中也标明姓名、学号。
4) 程序关键算法、数据结构、变量等必须包含有相应的注释说明。
5) 本次考试总分为 100 分，评分标准请见试题末。注意：如果不能完成全部程序功能，也请不要担心，我们会根据每一步骤独立评分。

2.考试题目：哈夫曼压缩

哈夫曼编码（Huffman Coding），是一种用于无损压缩编码的熵编码方法。其由美国计算机科学家 David Albert Huffman 在 1952 年发明。

哈夫曼编码使用变长编码表对原符号（如文件中的字符）进行编码。其中变长编码表通过评估符号出现的频率得到。出现频率高的字母使用较短的编码，而出现频率低的编码则使用较长的编码。这种编码方式使得编码之后的字符串平均长度的期望值降低，从而达到无损压缩数据的目的。

在本题目中，你需要使用哈夫曼编码对给定的文件进行压缩，并按照要求输出指定信息。**具体过程**如下：

步骤一：统计。

读取输入文件，并统计其中各个字符（所有 ASCII 字符，包括空格、换行等不可见字符）出现的次数。在统计完毕后，你需要在标准输出中打印出四行信息：

第一行为一个数字 N，表示输入文件中共出现过多少种不同的 ASCII 字符；

其余三行为文件中出现次数最多的三个字符和它们的出现个数，字符与其出现字数之间使用一个空格隔开。对于出现次数相同的两个字符，优先打印字符 ASCII 码较小的字符。

该步骤应实现在一个名为 do_statistics 的函数中。所有的字符与其出现个数，应保存在一个名为 freqTable 的变量中。

步骤二：建立哈夫曼树；

该步骤应按照要求建立一棵哈夫曼树，建立方法可见文档最后的提示。

为了保证哈夫曼树的唯一性，在选择两个树合并时，若有多个树的权值相同，则优先选择 ASCII 码最小的树。（一个树的 ASCII 码，为这个树中所有节点的 ASCII 码的最小值。）同时，ASCII 码较小的子树应作为左子树，ASCII 码较大的子树为右子树。

此步骤完成后，请在标准输出中打印出该哈夫曼树的深度（提示：根节点的深度为 0）。

此步骤应实现在一个名为 build_tree 的函数中。哈夫曼树结构（或哈夫曼树的根）应保存在一个名为 huffmanTree 的变量中。

步骤三：编码。

在本步骤中，你需要根据哈夫曼树的结构对各个字符进行编码，并将编码保存在指定名称的编码文件中。

在进行编码时，向左走（左子树）应赋值为 0，向右走（右子树）应赋值为 1。

编码文件包括 N 行信息，其中应以字符的 ASCII 码大小为序，从小到大打印每个字符和其编码的二进制表示，中间以空格隔开。（由于换行字符也会被编码，文件的实际行数可能会大于 N。）

在编码文件输出完毕后，请打印字符 e 所对应的编码的二进制表示（测试数据保证题目中存在字符 e）。

该步骤应实现在一个名为 encode 的函数中。每个字符和其编码，应保存在一个名为 codingTable 的变量中。

步骤四：压缩文件。

根据每个字符的编码，对原文件进行压缩，并保存到输出文件中。输出文件的前八个字节，以小端模式保存一个数字，为该文件中**压缩数据**的有效比特数。此后保存压缩数据。

由于编码（即压缩）后的数据为比特流，在转换成字符流时，应按照大端顺序。以下为一个示例（编码为样例 1 中的编码）：

```
字符流：this<SPC>is<SPC> （<SPC>为空格）
比特流：101 0111 1000 001 110 1000 001 110
重新组合：1010 1111 0000 0111 0100 0001 1100 0000（最后 5 个 0 为补足）
字节流：0xAF 0x07 0x41 0xC0
```

如果比特流末尾不足以转换成字符流，则通过在后面以二进制 0 进行补足（注意此时补足所用的 0 不算作有效数据）。

在此步骤的最后，应在标准输出中打印压缩文件中有效数据的比特数。

此步骤应实现在一个名为 compress 的函数中。

提示：哈夫曼树

哈夫曼压缩中需要使用到哈夫曼树。给定 N 个权值作为 N 个叶子结点，构造一棵二叉树，若该树的带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树（Huffman Tree）。哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。

哈夫曼树构造过程如下：

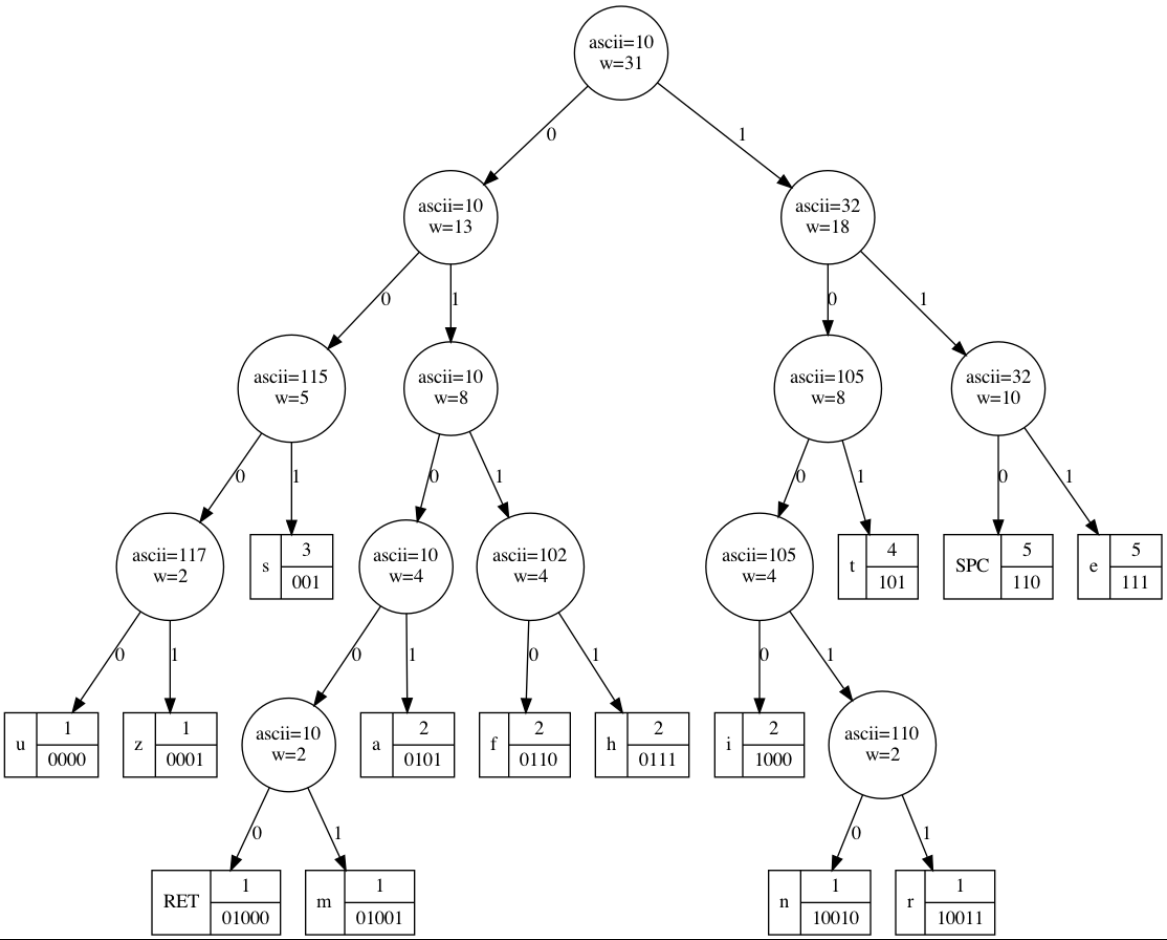
假设有 n 个权值，则构造出的哈夫曼树有 n 个叶子结点。 n 个权值分别设为 w1、w2、...、wn，则哈夫曼树的构造规则为：

(1) 将 w1、w2、...，wn 看成是有 n 棵树的森林（每棵树仅有一个结点）；

(2) 在森林中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和；
(3) 从森林中删除选取的两棵树，并将新树加入森林；
(4) 重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求得的哈夫曼树。

一个哈夫曼树的例子：

this is a huffman tree testeez



图中叶子结点分别保存了字符、权重和编码，其中 RET 为回车(\n)，SPC 为空格。中间结点则为该子树的 ascii 码和权重。

3.测试样例

样例 1:
标准输入

```
sample.txt sample.huffidx sample.huffzip
```

sample.txt 文件（注意，最后有换行符\n）

```
this is a huffman tree testeez
```

输出
标准输出

```
14
  5
e 5
t 4
5
111
111
```

说明：其中第一行的 14 为输入文件中不同的字符数。之后三行为出现最多的三个字符和其出现次数。在之后的 5 为哈夫曼树的深度（见上面的哈夫曼例子的图）。之后的 111 为字母 e 的编码。最后的 111 表示压缩后有效数据大小 111 个比特。

sample.huffidx 文件内容（第一行为字符\n）

```

 01000
  110
a 0101
e 111
f 0110
h 0111
i 1000
m 01001
n 10010
r 10011
s 001
t 101
u 0000
z 0001
```

sample.huffzip 为二进制文件，请见给出的文件。

小规模测试:

标准输入

```
small.txt small.huffidx small.huffzip
```

small.txt 文件（注意，最后有换行符\n）

```
It was like just before the sun goes to bed down on the bayou.
There was a million sparkles on the river.
```

输出

标准输出

```
27
  20
e 12
o 8
7
011
448
```

small.huffidx 文件内容（第一行为字符\n）

```

 1110111
```

```
  00
. 100110
I 1001110
T 1001111
a 11110
b 01010
d 101100
e 011
f 1011010
g 1011011
h 10111
i 11000
j 1100100
k 110011
l 11100
m 1100101
n 11111
o 1101
p 1110100
r 0100
s 1010
t 1000
u 01011
v 1110101
w 10010
y 1110110
```

small.huffzip 为二进制文件，请见给出的文件。

4.评分标准

本测试的评分标准如下：

1）文件读入和统计（步骤 1）占 15 分；
2）哈夫曼树的建立（步骤 2）占 20 分；
3）字符的编码（步骤 3）占 15 分；
4）压缩文件（步骤 4）占 20 分；
5）小规模测试的正确性占 10 分；
6）大规模测试的正确性占 10 分；
7）代码规范性（合理的注释、程序的可理解性等）占 10 分。