**南昌航空大学**

**21学年—22学年第 2 学期 智能医疗信息处理专业实验**

**实验二**

专业名称： 生物医学工程 实验学时： 2

学号： 19084127 姓名： 周亚诺

实验题目：二叉树的应用－哈夫曼编码的实现

实验环境： Pycharm

实验目的：

1．掌握二叉树的定义；

2．掌握哈夫曼树和哈夫曼编码算法的实现。

实验内容：

实现一个哈夫曼编码系统，系统包括以下功能：

(1) 字符信息统计：读取待编码的源文件SourceFile.txt，统计出现的字符及其频率。

(2) 建立哈夫曼树：根据统计结果建立哈夫曼树。

(3) 建立哈夫曼码表：利用得到的哈夫曼树，将各字符对应的编码表保存在文件Code.txt中。

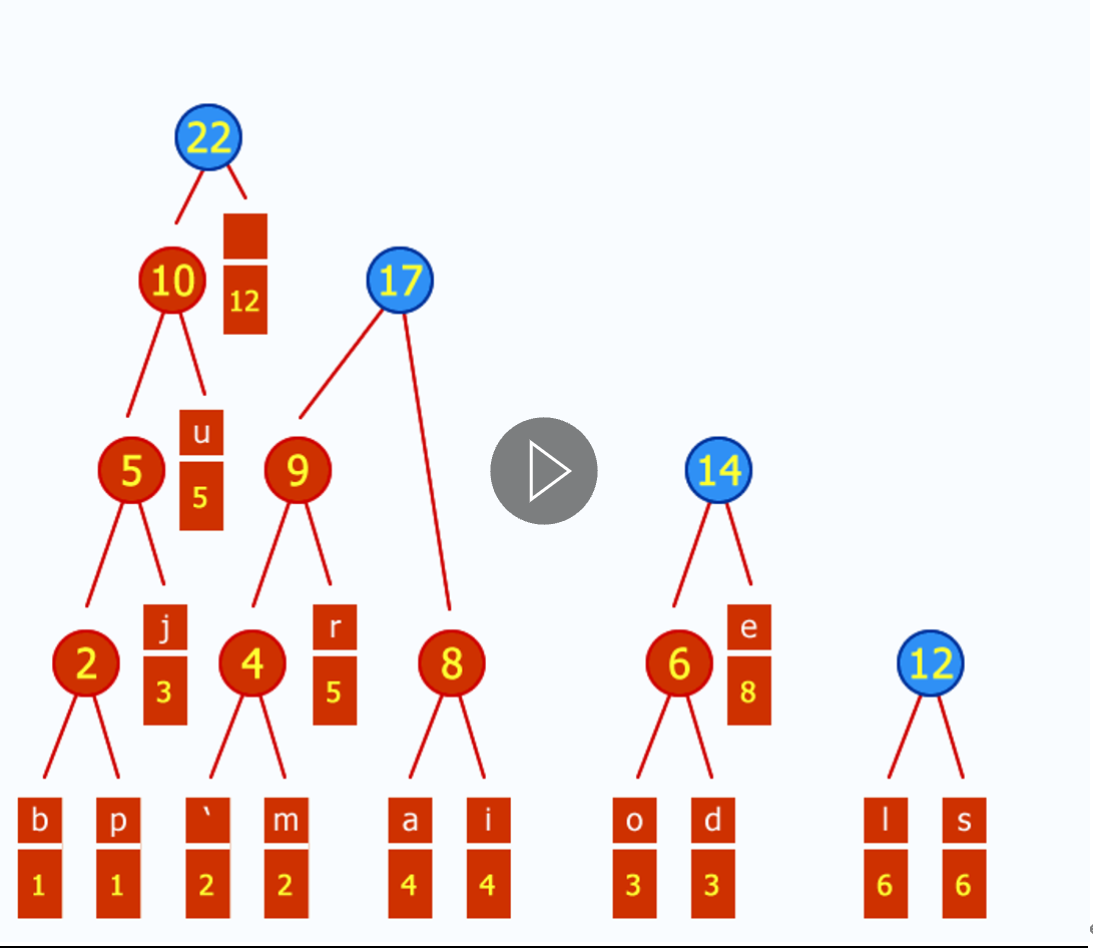
(4) 对源文件进行编码：根据哈夫曼码表，将SourceFile.txt中的字符转换成相应的编码文件ResultFile.txt。

实验要求：

(1) 程序要具在一定的健壮性，即当输入数据非法时，程序也能适当地做出反应。

(2) 程序要添加适当的注释，程序的书写要采用缩进格式。

实验流程图：



实验代码：

from collections import Counter  
  
  
def find\_char\_freq(text):  
    result = dict()  
    with open(text, 'r') as f:  
        for line in f.readlines():  
            line = line.lower()  
            for i in line:  
                if i.isalpha():  
                    if i in result:  
                        result[i] += 1  
                    else:  
                        result.update({i: 1})  
    return result  
  
  
class *Node*:  
    def \_\_init\_\_(self):  
        self.frequency = 0  
        self.name = None  
        self.lchild = None  
        self.rchild = None  
        self.code = None  
  
    def \_\_lt\_\_(self, other):  
        return self.frequency < other.frequency  
  
  
# establish the Huffman Tree  
def estblishHuffmanTree(info\_dict):  
    # output: the base node  
    node\_list = []  
    for i in info\_dict:  
        a = Node()  
        a.frequency = info\_dict[i]  
        a.name = i  
        node\_list.append(a)  
    while len(node\_list) > 1:  
        node\_list.sort(reverse=True)  
        node\_1 = node\_list.pop()  
        node\_2 = node\_list.pop()  
        new\_node = Node()  
        new\_node.frequency = node\_1.frequency + node\_2.frequency  
        new\_node.lchild = node\_1  
        new\_node.rchild = node\_2  
        node\_list.append(new\_node)  
    return new\_node  
  
  
def encode(node, rst\_dict, code):  
    if node.name:  
        rst\_dict.update({node.name: code})  
        return  
    code += '0'  
    encode(node.lchild, rst\_dict, code)  
    code = code[:-1]  
    code += '1'  
    encode(node.rchild, rst\_dict, code)  
    return rst\_dict  
  
  
def encode\_text(code\_dict, text, code\_text):  
    string = ''  
    with open(text, 'r') as f:  
        for line in f.readlines():  
            line = line.lower()  
            for i in line:  
                if i.isalpha():  
                    string += code\_dict[i]  
                else:  
                    string += '\n'  
    with open(code\_text, 'w') as f:  
        f.write(string)  
  
  
def decode(text\_de, result\_address, base\_node):  
    text\_string = ''  
    a = base\_node  
    with open(text\_de, 'r') as f:  
        for line in f.readlines():  
            for i in line:  
                if i == '0':  
                    b = a.lchild  
                    if b.name:  
                        text\_string += b.name  
                        a = base\_node  
                    else:  
                        a = b  
                elif i == '1':  
                    b = a.rchild  
                    if b.name:  
                        text\_string += b.name  
                        a = base\_node  
                    else:  
                        a = b  
                else:  
                    text\_string += '\n'  
    with open(result\_address, 'w') as f:  
        f.write(text\_string)  
  
# TODO: print encodings with issue, needs to fix.  
def print\_dict(d):  
    print(sorted(d.items(), key=lambda x: int(x[1])))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
    text = "SourceFile.txt"  
    char\_freq = find\_char\_freq(text)  
    print\_dict(char\_freq)  
  
    base\_node = estblishHuffmanTree(char\_freq)  
    code\_dict = encode(base\_node, {}, '')  
    print\_dict(code\_dict)  
  
    code\_text = "ResultFile.txt"  
    encode\_text(code\_dict, text, code\_text)  
  
    de\_address = "decode.txt"  
    decode(code\_text, de\_address, base\_node)

心得体会：

掌握了二叉树的定义，并掌握了哈夫曼树和哈夫曼编码算法的实现。