# 第三次算法设计作业

姓名： 刘宇骋 学号： 2021141440374

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）和代码。作业提交时请只保留选做的题，删除空白题。

【题一】给定一个字符集合以及每个字符的出现频率，设计一个算法构建霍夫曼编码树，并输出字符集合的霍夫曼编码。霍夫曼编码是一种变长编码，它根据字符出现的频率来构建不同长度的编码，使得频率高的字符具有短的编码长度，频率低的字符具有长的编码长度。

【输入格式】 第一行输入一个整数 n 表示字符集合的大小。 接下来 n 行，每行输入一个字符和该字符出现的频率，用空格分隔。

【输出格式】 输出 n 行，每行输出一个字符以及对应的霍夫曼编码。

【示例】

输入：5

A 45

B 13

C 12

D 16

E 9

输出：A 0

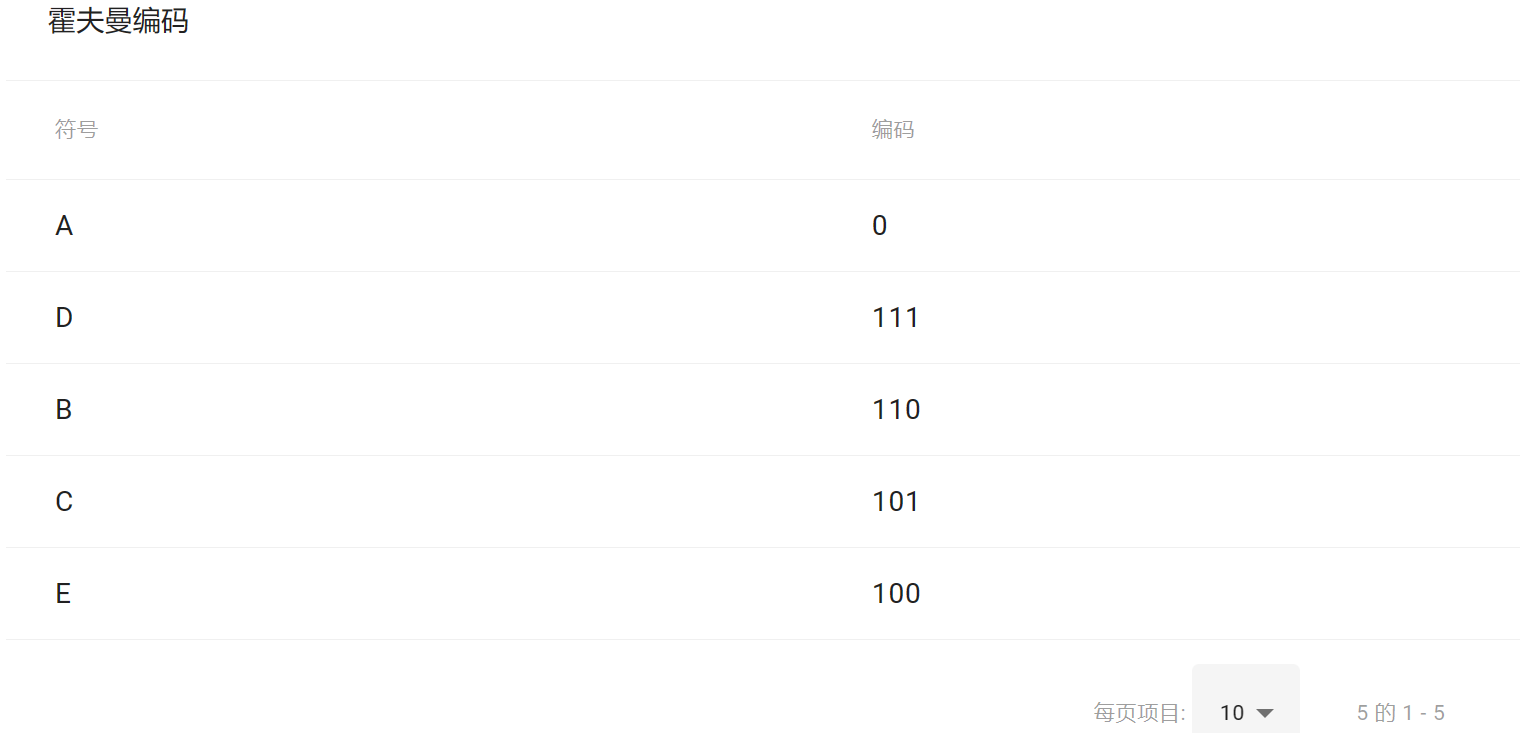
B 10

C 110

D 1110

E 1111

这里的编码有误吧，我用在线霍夫曼编码器，结果是这个：



【解题思路】

首先，利用频次表为每一个字符创建一颗二叉树，并将其出现频次存储在节点中，节点数据结构定义为TreeNode。在列表中寻找根节点中存有最小频率值的两颗二叉树，创建一个不存储任何字符的节点，将这两颗二叉树作为新建节点的左右子树，并将其频次值之和存储到新建节点中。在这6个节点中，频次之和最小的两个是b、c，将左节点的val更新为0，右节点的val更新为1，合并为一个新的二叉树。得到最终的二叉树后，就可以进行编码了，依次遍历二叉树的每一条路径，取每个节点的val值，遍历到叶结点后，叶结点ch值所对应的的路径去除第一个字符即为该字符的编码。

【代码】

package algorithm;  
  
import java.util.\*;  
  
public class Huffman {  
 public Map<Character, String> encode(Map<Character, Integer> dataMap) {  
 Map<Character, String> encodeMap = new HashMap<>();  
 TreeNode tree = constructTree(dataMap);  
 findPath(tree, encodeMap, new StringBuilder());  
 findPath(tree, encodeMap, new StringBuilder());  
 return encodeMap;  
 }  
  
 /\*  
 \* 根据字符串建立二叉树  
 \* @param s：要编码的源字符串  
 \*/  
 private TreeNode constructTree(Map<Character, Integer> dataMap) {  
 //遍历dataMap,初始化二叉树节点，并将所有初始化后的节点放到nodeList中，并进行排序  
 LinkedList<TreeNode> nodeList = new LinkedList<TreeNode>();  
 for (Map.Entry<Character, Integer> entry : dataMap.entrySet()) {  
 Character ch = entry.getKey();  
 int freq = entry.getValue();  
 int val = 0;  
 TreeNode tmp = new TreeNode(ch, val, freq, null, null);  
 nodeList.add(tmp);  
 }  
 //对存放节点的链表进行排序，方便后续进行组合  
 Collections.sort(nodeList, new Comparator<TreeNode>() {  
 public int compare(TreeNode t1, TreeNode t2) {  
 return t1.freq - t2.freq;  
 }  
 });  
  
 //size==1,代表字符串只包含一种类型的字母  
 if (nodeList.size() == 1) {  
 TreeNode t = nodeList.get(0);  
 return new TreeNode(null, 0, nodeList.get(0).freq, t, null);  
 }  
  
 //利用排序好的节点建立二叉树，root为初始化根节点  
 TreeNode root = null;  
 while (nodeList.size() > 0) {  
 //因为nodeList在前面已经排好序，所以直接取出前两个节点，他们的和肯定为最小  
 TreeNode t1 = nodeList.removeFirst();  
 TreeNode t2 = nodeList.removeFirst();  
 //左子树的val赋值为0，右子树的val赋值为1  
 t1.val = 0;  
 t2.val = 1;  
 //将取出的两个节点进行合并  
 if (nodeList.size() == 0) {  
 //此时代表所有节点合并完毕，返回结果  
 root = new TreeNode(null, 0, t1.freq + t2.freq, t1, t2);  
 } else {  
 //此时代表还有可以合并的节点  
 TreeNode tmp = new TreeNode(null, 0, t1.freq + t2.freq, t1, t2);  
  
 //t1、t2合并后，需要将得到的新节点加入到原链表中，继续与其他节点合并，  
 //此时需要保证原链表的有序性，需要进行排序  
 if (tmp.freq > nodeList.getLast().freq) {  
 nodeList.addLast(tmp);  
 } else {  
 for (int i = 0; i < nodeList.size(); i++) {  
 int tmpFreq = tmp.freq;  
 if (tmpFreq <= nodeList.get(i).freq) {  
 nodeList.add(i, tmp);  
 break;  
 }  
 }  
 }  
 }  
 }  
 //返回建立好的二叉树根节点  
 return root;  
 }  
  
 //对已经建立好的二叉树进行遍历，得到每个字符的编码  
 private void findPath(TreeNode root, Map<Character, String> res, StringBuilder path) {  
 if (root.left == null && root.right == null) {  
 path.append(root.val);  
 res.put(root.ch, path.substring(1));  
 path.deleteCharAt(path.length() - 1);  
 return;  
 }  
 path.append(root.val);  
 if (root.left != null) findPath(root.left, res, path);  
 if (root.right != null) findPath(root.right, res, path);  
 path.deleteCharAt(path.length() - 1);  
 }  
  
 //内部类 二叉树节点  
 private class TreeNode {  
 Character ch;  
 int val;  
 int freq;  
 TreeNode left;  
 TreeNode right;  
  
 public TreeNode() {  
 }  
  
 public TreeNode(Character ch, int val, int freq, TreeNode left, TreeNode right) {  
 this.ch = ch;  
 this.val = val;  
 this.freq = freq;  
 this.left = left;  
 this.right = right;  
 }  
 }  
}  
  
class HuffmanTest {  
 public static void main(String[] args) {  
 int charNum;  
 Scanner scanner = new Scanner(System.in);  
 charNum = scanner.nextInt();  
 Map<Character, Integer> charSet = new HashMap<>();  
 for (int i = 0; i < charNum; i++) {  
 Character c = scanner.next().charAt(0);  
 Integer freq = scanner.nextInt();  
 charSet.put(c, freq);  
 }  
 Huffman huffman = new Huffman();  
 Map<Character, String> encodeMap = huffman.encode(charSet);  
 System.out.println(encodeMap);  
 }  
}

【结果验证】1、输入：5

A 45

B 13

C 12

D 16

E 9

