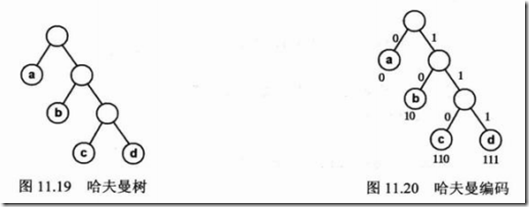
**霍夫曼树编码解码需求分析**

**一、设计内容和要求**

* 运用哈夫曼编码的相关知识对任意文本文件进行编码、解码。
  + 根据要编码的文件中字符出现的频率生成对应的哈夫曼编码。
  + 得到采用哈夫曼编码后的目标文件，并保存。
  + 根据要解码的文件对应的哈夫曼码表对文件进行解码。
  + 得到解码后的目标文件并保存。
* 扩展功能：
  + 通过机器学习算法实现ocr文字识别，对一张图片上的的文字自动识别并压缩
  + 设计动画描述霍夫曼树的构建过程
  + 运用霍夫曼树对位图进行编码、解码
  + 能够在android平台运行

**二、所需数据结构和算法**

* 霍夫曼树
  + 基本思想:用较少的比特表示出现频率高的字符,用较多的比特表示出现频率低的字符
  + 假设需要把一个字符串*如"abcdabcaba"*进行编码,将它转化为唯一的**二进制码**,要求**转换出的二进制码的长度最小**
  + 假设每个字符在字符串在字符串中出现的**频率**为W,其编码长度为L,编码字符n个,则编码后的二进制码的总长度为W1L1 + W2L2 + ... + WnLn.因此可采用哈夫曼树的构造原理进行二进制编码,从而使得电文长度最短
  + **构建过程**: 将**字符在字符串中出现的次数**作为权值,构建好一棵完整的哈夫曼树之后,从**根节点**开始,对左子树分配代码0,对右子树分配1,一直到达叶子节点.然后将**从树根沿着每条路径到达叶子节点的代码排列起来**,便得到每个叶子节点的哈弗曼编码.然后把每个字符编码连接起来就得到对应的字符串的二进制编码

//霍夫曼树结点的实现  
private static class Node implements Comparable<Node> {  
        private char ch;        //叶子节点中需要被编码的字符(内部结点不会使用该变量)  
        private int freq;       //展开过程不会使用该变量  
        private final Node left, right;  
​  
        public Node(char ch, int freq, Node left, Node right) {  
            this.ch = ch;  
            this.freq = freq;  
            this.left = left;  
            this.right = right;  
       }  
​  
        public boolean isLeaf() {  
            return left == null && right == null;  
       }  
​  
        @Override  
        public int compareTo(Node o) {  
            return this.freq - o.freq;  
       }  
   }  
​

* 优先队列
* 相关机器学习算法

**三、团队开发版本控制**

* 利用github同步开发进程
* 仓库地址: https://github.com/ZhaoLizz/DataCompression.git

**四、代码命名及风格**

* 使用驼峰命名法

private Key[] pq;                      
private int n;                        
private Comparator<Key> comparator;  
private static final int R = 256;

* 首大括号同行

private void resize(int capacity) {  
        assert capacity > n;  
        Key[] temp = (Key[]) new Object[capacity];  
        for (int i = 1; i <= n; i++) {  
            temp[i] = pq[i];  
       }  
        pq = temp;  
   }

* 注释风格
  + 普通注释//或者/\*\*aaa\*\*/都可以
  + 方法注释用java标准文档注释

/\*\*  
     \* Initializes a priority queue from the array of keys.  
     \* <p>  
     \* Takes time proportional to the number of keys, using sink-based heap construction.  
     \*  
     \* @param keys the array of keys  
     \*/  
    public MinPQ(Key[] keys) {  
        n = keys.length;  
        pq = (Key[]) new Object[keys.length + 1];  
        for (int i = 0; i < n; i++)  
            pq[i+1] = keys[i];  
        for (int k = n/2; k >= 1; k--)  
            sink(k);  
        assert isMinHeap();  
   }

**五、团队分工**

* 赵励志：实现文件压缩、解码相关算法及android端的架构
* 杨玉林：实现ocr文字识别并在android端移植
* 秦楚霁：对中文字符及位图进行预处理
* 蒋昱葳：设计自定义View实现霍夫曼树的构建动画

| **姓名** | **学号** | **任务** |
| --- | --- | --- |
| 赵励志 | 1607094117 | 实现文件压缩、解码相关算法及android端的架构 |
| 杨玉林 | 1607094117 | 实现ocr文字识别并在android端移植 |
| 蒋昱葳 | 1607094117 | 设计自定义View实现霍夫曼树的构建动画 |
| 秦楚霁 | 1607094117 | 对中文字符及位图进行预处理 |