1. 该次PJ是利用哈夫曼树构造哈夫曼编码进而压缩文件大小，使得高频字符的字符变短，从而达到将文件压缩的效果，然而由于头文件的原因，所以当文件大小太小的时候可能造成压缩文件的大小变大，所以该pj所写的压缩工具适用于压缩大文件，而不是小文件。
2. 该次pj首先要建立一个Huffman节点的数据类型，我们可以重写compare方法使得运用Collections.sort排列HuffmanNode(以weight（权重）进行排列)。由于文件读出的编码为255以内的ASCII编码；所以我们可以创建一个int数组长度为256个，这样可以很容易的统计文件中出现的字符次数，方便我们创建Huffmancode;当我们获得所有的节点时，我们可以开始建树了，由于我用的是Arrarylist,可以用remove（）方法获得权重最小的两个，创建新的父节点，同时将数据大于0的节点存入叶节点数组中方便后面的创建哈夫曼编码。（创建的空的父节点我将其数据设置-1这要我们方便我们存入叶节点）。
3. 之后我们开始构建编码，我们通过叶节点数组将一个个子叶的编码编好，从叶节点出发，直到到达根部，这样我们就可以获得要被压缩的文件内每一个ASCII编码的哈夫曼编码。

同时我们可以利用java中自带的hashmap将ASCII码和哈夫曼编码联系到一起，这样在将ASCII编码转换成哈夫曼编码的时间效率提高；

1. 现在首先难点是写入头文件，一开始的时候头文件很简单，但随着不断考虑后头文件的内容在不断的增加，最后我将其缩短成相对文件夹并且通过网上（室友）得知文件命中不能含有<>，通过镶入这个编码将文件名读出来，之后写入是否为文件夹（为将来的将文件夹压缩成为一个压缩文件做准备），是否为空文件（储存了文件大小的长度），并根据这个数值在解压时我们可以知道读几位就可以得到文件的长度。然后存入字符种类（通过两个字节储存），然后循序存入ASCII编码和其哈夫曼编码长度，我们可以通过前面的两个字符种类字节来得到我们对应的ASCII编码和他的哈夫曼长度，然后将所有的哈弗曼编码按照之前的顺序将其串接在一起，之后解压时开始切割其哈夫曼编码；这样我们解压缩就可以较为轻松的获得所需文件的哈弗曼表。串接在一起的哈弗曼编码我将其转换成8位8位的byte存起来，进一步节省空间，之后就是根据文件的内容转换成对应的哈曼编码的String,并将该String转换成byte存入。同时由于有可能会出现String无法满足8位形成一个byte我们要根据情况补0；这样我们才算是将文件压入；
2. 读文件则根据我们写入的东西一一对应的记录，在获得哈夫曼表后根据长度来决定我们反写入解压后的文件内，最后获得解压文件。由于我们曾经记录过文件的长度，所以我们可以忽视补零的个数（当写到文件的长度达到时，即可跳出循环）

6. 之后是将由于要将文件夹及其包含的文件进行统一压缩，我重新构建了一个压缩和解压的文件类，压缩时我们要除去绝对路径的一部分，保留下相对路径的一部分，这样可以让我们解压出的文件不会回到源文件，而是在当前文件中创建的文件夹及其内容。同时由于我们写入了长度，所以我们在解压时可以很轻松的将不同文件文件夹的区分开来，分别解压。

7. 由于时间不太够，所以没写GUI，同时还存在一些bug待修复，同时我用了一些网上的将8位01的String转成int和int转成8位01String的方法，以及递归遍历文件内容（用来压缩文件夹）。以上。还有些详细的思路来不及写到开发档案orz，等到时面试的时候再回答助教