### 哈夫曼编码

1. 哈夫曼编码原理剖析
2. 传输的字符串：i like like like java do you like a java
3. 统计各个字符出现的次数：d:1,y:1,u:1,j:2,v:2,o:2,l:4,k:4,e:4,i:5,a:5, :9
4. 按照各个字符出现的次数构造一颗哈夫曼树，次数作为权值
5. 哈夫曼树构造步骤：

1>将数据从小到大排序，每一个数据都看成一个节点，也可以看做一颗最简单的二叉树

2>取出根节点权值最小的两颗二叉树

3>将取出的节点组成一颗新的二叉树，组成的新的二叉树的根节点的权值就是前面取出的两个节点权值的和

4>再排序，重复1-2-3-4步骤，直到所有数据都被处理过，哈夫曼树就构建完成了

d:1,y:1,u:1,j:2,v:2,o:2,l:4,k:4,e:4,i:5,a:5, :9（空格数）

1. 根据哈夫曼树给各个字符规定编码(前缀编码，每个字符的编码都不会成为另一个字符的前缀，不会出现字符匹配的同意性)，向左路径为0，向右路径为1，如下图：

o:1000 u:10010 d:100110 y:100111 i:101

a:110 k:1110 e:1111 j:0000 v:0001 l:001 :01

1. i like like like java do you like a java编码后：

1010100110111101111010011011110111101001101111011110100001100001110011001101000011001111000100100100110111101111011100100001100001110

压缩率：(编码前的字符编码长度-压缩后的字符编码长度)/编码前的字符编码长度

哈弗曼编码是无损压缩

编码是将传输的信息进行二进制编码

**注意：**哈夫曼树根据排序不同，构建成的哈夫曼树就有可能不同，对应的哈弗曼编码也不尽相同，但WPL都是最小的

K:4

e:4

8

a:5

13

d:1

y:1

u:1

2

3

o:2

i:5

5

v:2

J:2

L:4

4

10

：9

8

40

17

23

0 1

0 1 0 1

0 1 0 1

0 1

1 0 1

0

0 1

0 1

1. 功能：根据哈弗曼编码压缩数据的原理，需要创建i like like like java do you like a java

对应的哈夫曼树

思路：

1.创建节点类：Node{data(存放数据),weight(权值),left,right}

2.得到i like like like java do you like a java对应的byte[]数组

3.编写方法，构建哈夫曼树的Node，节点存放到List中，体现

d:1,y:1,u:1,j:2,v:2,o:2,l:4,k:4,e:4,i:5,a:5,:9(空格数) ，形式为[Node[data=97,weight=5],Node[data=32,weight=9].......],97是a的阿斯科马志

1. 通过List创建对应的哈夫曼树