Huffman Coding

Motivation

Huffman coing is a data compression. Supposing that we have a large a amount of text that we wish to store on a computer disk in an efficient wat. The simpest way to do this is simply to assign a binary code to each character and store the binary codes consecutively in the computer memory.

There are two different way to represent the character using binary code: fixed length binary coding or variable length bunary coding.

However, different characters occur in different frequence, and some characters occur far more frequently than others. So, it is possinle to save space by using a variable length code where the more frequently occurring characters are given shorter codes.

In order to be able to decode the variable length code properly, one character's binary code can not be other character's prefix code.

Huffman's Algorithm

1. The algorithm starts by creating a forest of single nodes, each representing one character, and each with an associated value, being the frequency of occurrence of that character. These values are places in a priority queue.
2. Remove two nodes L and R with the lowest value, and create a internal node of the binary tree whose left child is L and right child is R.
3. Compute the value of the new node as the sum of the values of L and R and insert this into priority queue.

Example

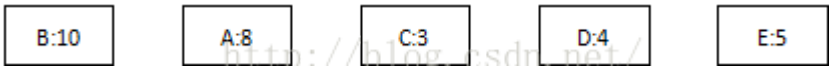
现有一个由5个不同符号组成的30个符号的字符串：

BABACAC ADADABB CBABEBE DDABEEEBB

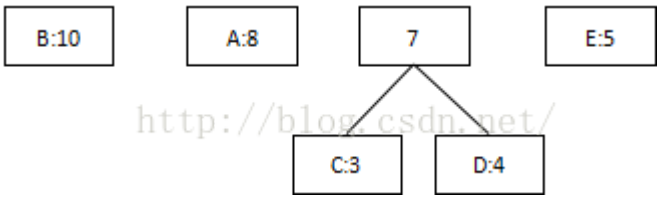
1首先计算出每个字符出现的次数（概率）：

字符	次数
B	10
A	8
C	3
D	4
E	5

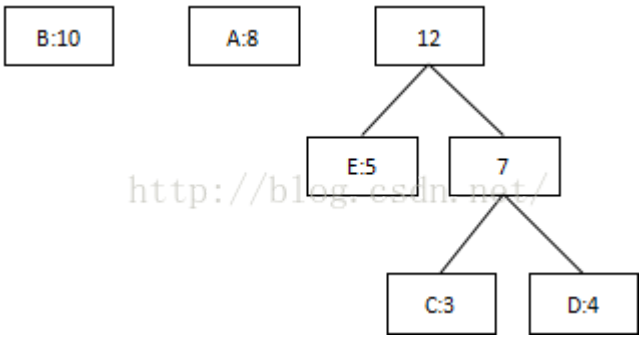
2把出现次数（概率）最小的两个相加，并作为左右子树，重复此过程，直到概率值为1



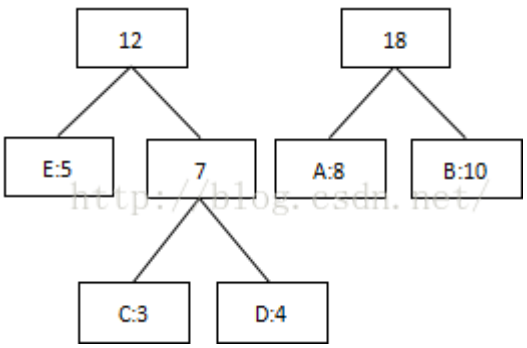
第一次：将概率最低值3和4相加，组合成7：



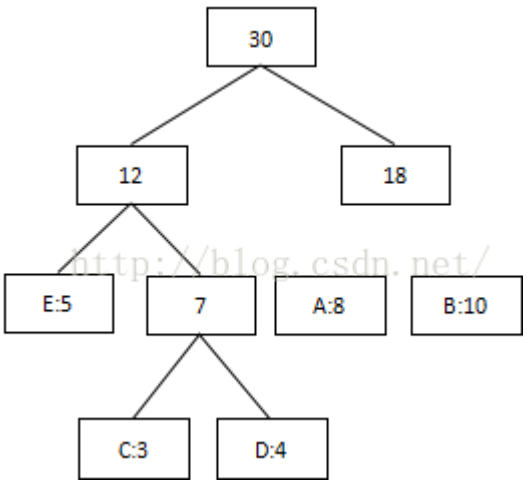
第二次：将最低值5和7相加，组合成12：



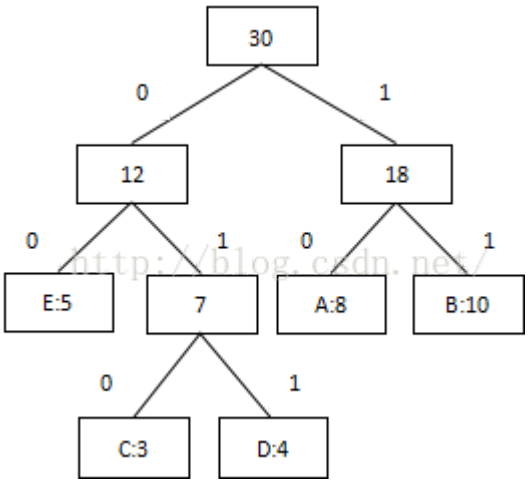
第三次：将8和10相加，组合成18：



第四次：将最低值12和18相加，结束组合：



3 将每个二叉树的左边指定为0，右边指定为1



4 沿二叉树顶部到每个字符路径，获得每个符号的编码

字符	次数	编码
B	10	11
A	8	10
C	3	010
D	4	011
E	5	00