Huffman编解码算法实现与压缩效率分析

1）Huffman编码的数据结构

Typedef struct huffman\_node\_tag //Huffman节点标签

{

unsigned char isLeaf //是否为树叶

unsigned long count; //节点代表的符号加权和

struct huffman\_node\_tag \*parent; //父节点指针

union

{

struct

{

struct huffman\_node\_tag \*zero, \*one; //子节点指针,分别代表0,1子节点指针

};

unsigned char symbol; //节点代表的符号

};

} huffman\_node;

typedef struct huffman\_code\_tag //Huffman编码标签

{

unsigned long numbits; //该码所用的比特数

unsigned char \*bits; //指向该码比特串的指针

} huffman\_code;

2）Huffman编码的流程

从指定文件中读取数据，统计每个符号发生的概率，并建立相应的树叶节点

注意在此处pSF代入的是地址

第一次扫描：统计文件中各个字符出现频率

构建Huffman树及生成Huffman码

按字符概率由小到大将对应节点排序

得到文件出现的字符种类数

构建Huffman树

对码树编码

将码表及其他必要信息写入输出文件

第二次扫描：对源文件进行编码并输出