哈尔滨工程大学

实 验 报 告

实 验 名 称：­­

班 级：

学 号：

姓 名：

实 验 时 间：

成 绩：

指 导 教 师：

实验室名称：

**哈尔滨工程大学实验室与资产管理处 制**

|  |
| --- |
| 实验名称：文件压缩 |
| 一、问题描述  哈夫曼编码是一种常用的数据压缩技术，对数据文件进行哈夫曼编码可大大  缩短文件的传输长度，提高信道利用率及传输效率。要求采用哈夫曼编码原理，  统计文本文件中字符出现的词频，以词频作为权值，对文件进行哈夫曼编码以达  到压缩文件的目的，再用哈夫曼编码进行译码解压缩。  二、数据结构设计  在本实验中，我采用结构体来存储哈夫曼树，用数组存储哈夫曼编码      // Huffman树结点定义，双亲存储结构    struct HTNode {        ElemType value;     // 权值，此处拓展为存储在此结点上的值        ElemType parent;    // 父节点        ElemType lchild;    // 左孩子位置        ElemType rchild;    // 右孩子位置    };    HTNode\* tree;   //Huffman树根节点指针    char\*\* code;    //Huffman编码    long lv;        //结点总数    long lencode;   //编码的长度  三、算法设计    本实验中设计了找到最小字符的函数，其算法如下：  void select(HuffmanTree<int>& hTree, int end, int& s1, int& s2){    int w1 = INT\_MAX;    int w2 = INT\_MAX;    for (int i = 1; i <= end; i++) {        if (hTree.tree[i].parent != 0) continue;        if (hTree.tree[i].value < w1) {            s2 = s1; w2 = w1;//w2次小的，w1。。。            s1 = i; w1 = hTree.tree[i].value;        }        else if (hTree.tree[i].value >= w1 && hTree.tree[i].value < w2) {            s2 = i; w2 = hTree.tree[i].value;//找到次小的权值w2        }    }}接着，便是建立哈夫曼树了，其算法如下：  void createHufTree(){  HT = (HufTree)malloc(m \* sizeof(HTnode)); //根结点  for(int i = 1, j = 0; i <= n; i++){  while(!cnt[j])  ++j;  HT[i] = HTnode(cnt[j], 0, (char)j);  ++j;  }  for(int i = n + 1; i < m; ++i)  HT[i] = HTnode(0, 0, '#');  for(int i = n + 1; i < m; ++i){  int s1 = select(i - 1);  HT[s1].p = i, HT[i].l = s1; //左孩子选择  int s2 = select(i - 1);  HT[s2].p = i, HT[i].r = s2; //右孩子选择  HT[i].weight = HT[s1].weight + HT[s2].weight; //权值合并处理  }  } //创建哈夫曼树  在建立哈夫曼树后，我们就要完成哈夫曼编码，其函数如下：  void initHuffman(HuffmanTree<int>& hTree,HuffmanTree<int>::\_rawdata\* datas,int n) //由已知的各个字符的编码完成全文的编码  {  int m = n \* 2; //树节点数  hTree.lv = m; //结点总数  hTree.lencode = n; //有效字符数  hTree.tree = (HuffmanTree<int>::HTNode\*)malloc((m + 1) \* sizeof(HuffmanTree<int>::HTNode)); //申请空间  if (!hTree.tree)exit(OVERFLOW); //1  hTree.tree[0].value = n;  hTree.tree[0].parent = -1;  hTree.tree[0].lchild = -1;  hTree.tree[0].rchild = -1;  HuffmanTree<int>::\_rawdata\* w = datas;  HuffmanTree<int>::HTNode\* p = hTree.tree;  int i = 0;  for (p = hTree.tree + 1, i = 1;  i <= n;  ++i, ++p, ++w) {  (\*p) = { (\*w).w,0,0,0 };  }  for (; i <= m; ++i, ++p) { (\*p) = { 0,0,0,0 }; }  //开始建树  int s1 = 0;  int s2 = 0;  for (i = n + 1; i < m; i++)  {  select(hTree, i - 1, s1, s2);  hTree.tree[s1].parent = i;  hTree.tree[s2].parent = i;  hTree.tree[i].lchild = s1;  hTree.tree[i].rchild = s2;  hTree.tree[i].value = hTree.tree[s1].value + hTree.tree[s2].value;  }  //Unkown  for (int i = 0; i < hTree.lv; i++)  {  if (hTree.tree[i].parent == 0 && hTree.tree[0].lchild == -1)hTree.tree[0].lchild = i;  else if (hTree.tree[i].parent == 0 && hTree.tree[0].rchild == -1)hTree.tree[0].rchild = i;  }  ////Huffman编码  hTree.code = (char\*\*)malloc((n + 1) \* sizeof(char\*));  if (!hTree.code)exit(OVERFLOW);  memset(hTree.code, 0, (n + 1) \* sizeof(char\*));  char\* tmp = (char\*)malloc(n \* sizeof(char));  if (!tmp)exit(OVERFLOW);  //哈夫曼编码00101  tmp[n - 1] = 0;  for (int i = 1; i <= n; ++i) {  int start = n - 1;  memset(tmp, 0, n \* sizeof(char));  for (int c = i, f = hTree.tree[i].parent; f != 0; c = f, f = hTree.tree[f].parent) {  if (hTree.tree[f].lchild == c) {  tmp[--start] = '0';  }  else {  tmp[--start] = '1';  }  }  hTree.code[i] = (char\*)malloc((n - start) \* sizeof(char));  if (!hTree.code[i])exit(OVERFLOW);  strcpy(hTree.code[i],&tmp[start]);  }  cout<<"各数据的依字频生成的Huffman编码为:"<<endl;  for (int i = 1; i <= n; i++)  {  printf("数据(%c) 对应的Huffman编码为:%s\n",datas[i - 1].c,hTree.code[i]);  }  cout<<endl;  free(tmp);  }void getCodeFile(){  ifstream in("CodeFile.dat");  if(!in.is\_open()){  cout<<"文件打开失败"<<endl;  exit(1);  }  string lll="";  char mm;  while(in >> mm)//从codefile里接收字符  {  lll += mm;  for(int i = 1;i <= 128;i++)//101001010  {  if(lll == HC[i]){  printf("%c",i);//向用户窗口输出字符  lll = "";//重置为空  break;  }  }    }  in.close();  }//文件解压  四、界面设计  本程序直接使用终端，一键完成编码压缩，再到解压的过程。     1. 运行测试与分析   1.运行程序，根据提示，如下图所示：      2.无效输入示例：    3.查看输入的文本：    4.查看生成的哈夫曼树文本：    5.查看生成的哈夫曼编码：    8.查看利用哈夫曼编码对原文字的压缩：    图9  六、实验收获与思考  这次实验我遇到的最大的困难就是读写文件，由于当时学c语言时学的不够扎实，读写文件不知道该如何操作，于是我上网查询了资料，学会了读写文件的操作函数。同时通过实现哈夫曼编码，使我对哈夫曼树有了更深的了解。也学会了很多编程技巧，让我受益匪浅。 |