学号 WA2214014 专业 人工智能 姓名 杨跃浙

实验日期 **6.17** 教师签字 成绩

实验报告

【实验名称】 哈夫曼树

【实验目的】

让学生更好的编写数据结构的算法

【实验原理】

实验题目  
。通过scanf输入n个叶子结点的权重，构建相应的huffman树，并给出每个叶子节点的编码码字(重视代码的理解!)  
·给出字符串“data structure i love you"中每个字符的huffman编码的码字  
，给定一个文件，通过huffman编码对该文件进行压缩，输出压缩后的文件，解压缩该压缩文件）

【实验内容】

、#include <iostream>

#include <string>

using namespace std;

#define OK 1

#define ERROR 0

#define OVERFLOW -2

#define MAXINT 10000

#pragma warning (disable:4996)

typedef struct {

int weight;

int parent, lchild, rchild;

}HTNode, \* HuffmanTree;

typedef char\*\* HuffmanCode;

void Select(HuffmanTree HT, int n, int& i, int& j)

{

i = j = 0;

for (int k = 1; k <= n; k++)

if ((HT[k].parent == 0) && (HT[k].weight < HT[i].weight))

i = k;

for (int k = 1; k <= n; k++)

if ((HT[k].parent == 0) && (HT[k].weight < HT[j].weight) && (k != i))

j = k;

}

void InitHuffmanTree(HuffmanTree& HT, int n)

{

if (n <= 1) return;

int m = 2 \* n - 1;

HT = new HTNode[m + 1];

HT[0].weight = MAXINT;

for (int i = 1; i <= m; i++)

{

HT[i].parent = 0;

HT[i].lchild = 0;

HT[i].rchild = 0;

}

}

void CreatHuffmanTree(HuffmanTree& HT, int n)

{

int m = 2 \* n - 1;

int s1, s2;

for (int i = n + 1; i <= m; i++)

{

Select(HT, i - 1, s1, s2);

HT[s1].parent = i;

HT[s2].parent = i;

HT[i].lchild = s1;

HT[i].rchild = s2;

HT[i].weight = HT[s1].weight + HT[s2].weight;

}

}

void CreatHuffmanCode(HuffmanTree HT, HuffmanCode& HC, int n)

{

HC = new char\* [n + 1];

char\* cd = new char[n];

cd[n - 1] = '\0';

for (int i = 1; i <= n; i++)

{

int start = n - 1; //层数 n-start

int c = i;

int f = HT[i].parent;

while (f)

{

start--;

if (HT[f].lchild == c) cd[start] = '0';

else cd[start] = '1';

c = f;

f = HT[f].parent;

}

HC[i] = new char[n - start];

strcpy(HC[i], &cd[start]);

}

delete cd;

}

void PrintCode(HuffmanCode HC, int n)

{

for (int i = 1; i <= n; i++)

cout << HC[i] << endl;

}

void PrintCodeString(HuffmanCode HC, int n, int\* tong)

{

int i = 1;

char c;

for (int j = 0; j < 27; j++)

{

if (tong[j])

{

if (j != 26)

{

c = 'a' + j;

cout << c << '\t' << HC[i] << endl;

}

else

cout << ' ' << '\t' << HC[i] << endl;

i++;

}

}

}

void Code()

{

char c = '\0';

int a[27] = { 0 };

getchar();

while ((c = getchar()) != '\n')

if (c != ' ') a[c - 'a']++; else a[26]++;

int n = 0;

HuffmanTree HT;

for (int i = 0; i < 27; i++)

if (a[i]) n++;

InitHuffmanTree(HT, n);

int k = 1;

for (int i = 0; i < 27; i++)

if (a[i])

{

HT[k].weight = a[i];

k++;

}

CreatHuffmanTree(HT, n);

HuffmanCode HC;

CreatHuffmanCode(HT, HC, n);

PrintCodeString(HC, n, a);

}

int main()

{

HuffmanTree HT;

HuffmanCode HC;

int n;

cin >> n;

InitHuffmanTree(HT, n);

for (int i = 1; i <= n; i++)

cin >> HT[i].weight;

CreatHuffmanTree(HT, n);

CreatHuffmanCode(HT, HC, n);

PrintCode(HC, n);

Code();

return 0;

}

【小结或讨论】

通过本次实验我了解了有关哈夫曼树的一系列操作，并能运用哈夫曼树实习对文件的简单压缩。