|  |  |
| --- | --- |
| haut | 河南工业大学信息科学与工程学院 |

**《算法设计与分析》实验报告**

|  |  |
| --- | --- |
| 课程名称： | 算法设计与分析 |
| 学生姓名： | 刘文博 |
| 学生学号： | 201716040224 |
| 学生专业： | 软件工程1702 |

**实验五：哈夫曼编码**

苗建雨

2019 年 5 月 9 日

实验日期： 2019 年 5 月 30 日 班级： 软件工程1702

学号（后四位）：\_\_\_0224\_\_\_\_\_\_ 姓名：\_\_\_\_刘文博\_\_\_\_\_ 成绩：

**一、 实验目的：**

了解前缀码的概念，理解数据压缩的基本概念。

掌握最优子结构的证明方法

掌握贪心算法的设计思想明能熟练运用

**二、实验要求：**

了解前缀码的概念，理解数据压缩的基本概念。掌握最优子结构的证明方法，掌握贪心算法的设计思想明能熟练运用。并用代码实现。

**三、实验内容：**

**1.** **Dijkstra算法**

通过对最短路径求解的学习，掌握动态规划算法的基本思想。Dijkstra算法是一种经

典的使用动态规划思想解决问题的算法，其主要思想是：初始集合S包含路径的起点，随

后每次从集合之外的点中选择一个到起点最近的点

D[i] = min{D[j]|vj ∈ V − S}

加入该集合，最后按照如下条件更新所有经过该点的路径的长度

D[k] = D[i] + arcs[i][k], if D[i] + arcs[i][k] < D[k];

**代码实现：**

**#include <iostream>**

**#include<vector>**

**#include"AboutTree.h"**

**using namespace std;**

**static int i = 0,k = 0;**

**int main()**

**{**

**system("mode con cols=105 lines=35");**

**vector<string> element;**

**vector<Node\*> vec\_node;**

**/\*\* 带解密的密码 \*/**

**char passworld[100];**

**/\*\* 待加密字符 \*/**

**char str[100];**

**char str1[100][50];**

**/\*\* 加密后的密码 \*/**

**char encryptionstr[200];**

**/\*\* 解密后的原码 \*/**

**char decodestr[100];**

**vector<int>show;**

**int choice;**

**pNode treeRoot = (pNode)malloc(sizeof(Node)) ;**

**treeRoot->left = 0; treeRoot->right = 0;**

**ReadingFile(vec\_node ,passworld , str);**

**CeateHalfTree(treeRoot,vec\_node);**

**TravelHalfTree(treeRoot,show,str1,i,k);**

**StoryHalfTree(str1,element,i);**

**while(1)**

**{**

**ShowMenu();**

**deep = 0;**

**cin >> choice ;**

**system("cls");**

**if(choice == -1){**

**SetPosition(0 , deep);**

**SetColor( 4 , 0);**

**cout<< "谢谢使用本程序！";**

**Sleep(1000);**

**SetColor( 10 , 0);**

**break;**

**}**

**switch(choice)**

**{**

**case 1:**

**{**

**PrintHalfTree(element);**

**break;**

**}**

**case 2:**

**{**

**EncryptionString(element ,str ,encryptionstr);**

**break;**

**}**

**case 3:**

**{**

**Decodestring(treeRoot, passworld ,decodestr);**

**break;**

**}**

**default :**

**{**

**SetPosition(0 , deep);SetColor( 4 , 0);**

**cout<< "请输入有效的选项序号！";**

**SetColor( 10 , 0);**

**break;**

**}**

**}**

**}**

**OutFile(element ,encryptionstr , decodestr);**

**return 0;**

**}**

**void CeateHalfTree(Node\* TreeRoot,vector<Node\*> &vec\_node)**

**{**

**while(!vec\_node.empty())**

**{**

**int elemnum = 0;**

**pNode newnode1,newnode2;**

**pNode newnode = (pNode)malloc(sizeof(Node));**

**newnode->left = 0;newnode->right = 0;**

**sort(vec\_node.begin() , vec\_node.end() , mycompare);**

**newnode1 = vec\_node.back();**

**elemnum++;**

**vec\_node.pop\_back();**

**if(!vec\_node.empty())**

**{**

**newnode2 = vec\_node.back();**

**elemnum++;**

**vec\_node.pop\_back();**

**}**

**if(elemnum == 2)**

**{**

**if(newnode1->weight < newnode2->weight)**

**{**

**newnode->left = newnode1;**

**newnode->right = newnode2;**

**}**

**else{**

**newnode->left = newnode2;**

**newnode->right = newnode1;**

**}**

**newnode->weight = newnode1->weight + newnode2->weight ;**

**newnode->str = '\*';**

**}**

**else if (elemnum == 1)**

**{**

**newnode->left = newnode1;**

**newnode->right = 0;**

**newnode->weight = newnode1->weight;**

**newnode->str = '\*';**

**}**

**if(vec\_node.empty())**

**{**

**TreeRoot->weight = newnode->weight;**

**TreeRoot->left = newnode->left;**

**TreeRoot->right = newnode->right;**

**TreeRoot->str = newnode->str;**

**break;**

**}**

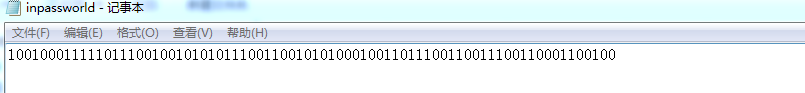
**else**

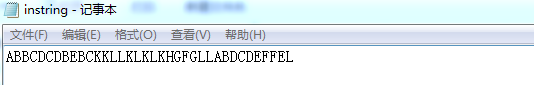
**vec\_node.push\_back(newnode);**

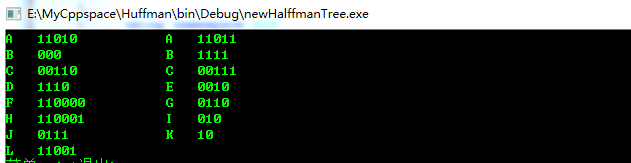
**}**

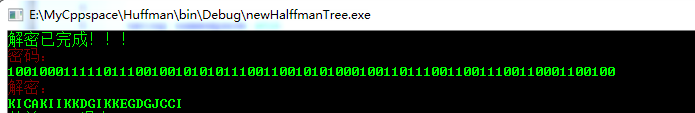
**实验截图：**

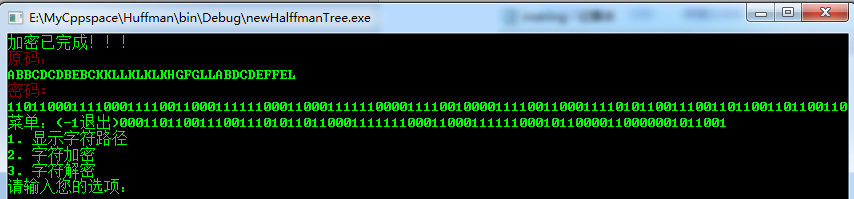












**四、实验结果&总结：**

给定n个权值作为n个[叶子结点](https://baike.baidu.com/item/%E5%8F%B6%E5%AD%90%E7%BB%93%E7%82%B9/3620239)，构造一棵二叉树，若该树的带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树(Huffman Tree)。哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。

假设有n个权值，则构造出的哈夫曼树有n个叶子结点。 n个权值分别设为 w1、w2、…、wn，则哈夫曼树的构造规则为：

(1) 将w1、w2、…，wn看成是有n 棵树的森林(每棵树仅有一个结点)；

(2) 在森林中选出两个根结点的权值最小的树合并，作为一棵新树的左、右子树，且新树的根结点权值为其左、右子树根结点权值之和；

(3)从森林中删除选取的两棵树，并将新树加入森林；

(4)重复(2)、(3)步，直到森林中只剩一棵树为止，该树即为所求得的哈夫曼树。