项目说明文档

数据结构课程设计

——修理牧场

作 者 姓 名： 伊啸

学 号： 1951220

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc60150252)

[1.1 项目简介 1](#_Toc60150253)

[2 设计 1](#_Toc60150254)

[2.1 数据结构设计 1](#_Toc60150255)

[2.2 类结构设计 1](#_Toc60150256)

[2.3 成员与操作设计 1](#_Toc60150257)

[2.4 系统设计 2](#_Toc60150258)

[3 实现 2](#_Toc60150259)

[3.1 链表中功能的实现 2](#_Toc60150260)

[3.2 构建霍夫曼树的实现 3](#_Toc60150261)

[3.2.1 构建霍夫曼树流程图 3](#_Toc60150262)

[3.2.2 构建霍夫曼树核心代码 5](#_Toc60150263)

[3.3 求最小长度的实现 5](#_Toc60150264)

[3.3.1 求最小长度流程图 5](#_Toc60150265)

[3.3.2 求最小长度核心代码 6](#_Toc60150266)

[3.3.3 求最小长度截屏示例 7](#_Toc60150267)

[3.4 总体系统的实现 8](#_Toc60150268)

[3.4.1 总体系统流程图 8](#_Toc60150269)

[3.4.2 总体系统核心代码 10](#_Toc60150270)

[3.4.3 总体系统截屏示例 10](#_Toc60150271)

[4 测试 10](#_Toc60150272)

[4.1 边界测试 10](#_Toc60150273)

[4.1.1 初始化无输入数据 10](#_Toc60150274)

[4.2.2 只有一个数据 11](#_Toc60150275)

[4.2 出错测试 11](#_Toc60150276)

[4.2.1 木头数量错误 11](#_Toc60150277)

[4.2.2 木头长度存在错误 12](#_Toc60150278)

# 1 分析

## 1.1 项目简介

农夫要修理牧场的一段栅栏，他测量了栅栏，发现需要N块木头，每块木头长度为整数*Li*个长度单位，于是他购买了一个很长的，能锯成N块的木头，即该木头的长度是*Li*的总和。

但是农夫自己没有锯子，请人锯木的酬金跟这段木头的长度成正比。为简单起见，不妨就设酬金等于所锯木头的长度。例如，要将长度为20的木头锯成长度为8，7和5的三段，第一次锯木头将木头锯成12和8，花费20；第二次锯木头将长度为12的木头锯成7和5花费12，总花费32元。如果第一次将木头锯成15和5，则第二次将木头锯成7和8，那么总的花费是35（大于32）.

# 2 设计

## 2.1 数据结构设计

容易想到用霍夫曼树来实现该项目的功能，同时，建立霍夫曼树的过程中需要频繁的取数据，因此用辅助链表来存数据。对霍夫曼树进行遍历时还要设置一个栈。

## 2.2 类结构设计

综上所述，类结构包括链表节点、链表、霍夫曼树的节点、霍夫曼树、存储霍夫曼树节点的栈等。因为链表只是辅助，且要便于取其中的数据，因此链表设置为struct类型的。

## 2.3 成员与操作设计

**链表结点类（LinkNode）**

**公有成员：**

  int data;//存储的数据

    LinkNode\* link;//指向下一节点的指针

**公有操作：**

 LinkNode()

        : data(0)

        , link(NULL)

    {

    }*//构造函数*

    LinkNode(int elem)

        : data(elem)

        , link(NULL)

    {

    }*//含默认参数的构造函数*

    ~LinkNode() {};*//析构函数*

**链表类（LinkList）**

**公有成员：**

 LinkNode\* head;*//链表头结点*

**公有操作：**

LinkList();*//构造函数*

~LinkList() {};*//析构函数*

bool Initial(int);*//初始化函数，输入数据存储在链表中*

 void Insert(const int& x);//将值为x的节点插入链表中

void Delete(int& x);*//删除值为x的节点*

void Search(int& x);

*//搜索整个链表让x成为最小的元素*

## 2.4 系统设计

系统首先实现对屏幕的初始化，完成对链表的创建和输入数据工作，然后每次从链表中取出最小元素构造霍夫曼树，最后用霍夫曼树实现功能

# 3 实现

## 3.1 链表中功能的实现

链表中实现的功能如将数据插入链表、从链表中删除指定元素、查找某个元素的操作之前的项目已经做过很多次，因此在这里不再赘述。

## 3.2 构建霍夫曼树的实现

### 3.2.1 构建霍夫曼树流程图



### 3.2.2 构建霍夫曼树核心代码

*for* (int i = 0; i < n - 1; i++) {

        list.Search(first);

        list.Delete(first);

        list.Search(second);

        list.Delete(second);

        HuffmanNode\* firstNode = new HuffmanNode(first);

        HuffmanNode\* secondNode = new HuffmanNode(second);

*//找到两个最小值，然后构造两个霍夫曼树节点*

        mergeTree(firstNode, secondNode, parent);

        arr[i] = \*parent;

        list.Insert(first + second);

    }

*for* (int i = 0; i < n - 1; i++) {

*for* (int j = i + 1; j < n - 1; j++) {

*if* (arr[i].data == arr[j].left->data) {

                arr[i].parent = &arr[j];

                arr[j].left = &arr[i];

            } *else* *if* (arr[i].data == arr[j].right->data) {

                arr[i].parent = &arr[j];

                arr[j].right = &arr[i];

            }

        }

    }*//将所有节点连接到霍夫曼树*

## 3.3 求最小长度的实现

### 3.3.1 求最小长度流程图

基本思路是求所有内部节点的和，采用前序遍历。



### 3.3.2 求最小长度核心代码

stack st(size);

    int sum = 0;

    HuffmanNode\* p = root;

    HuffmanNode\* q = NULL;

    st.Push(q);

*while* (p != q) {//前序遍历

*if* (p->right != NULL) {

            st.Push(p->right);

        }

*if* (p->left != NULL) {

            sum += p->data;

            p = p->left;

        } *else*

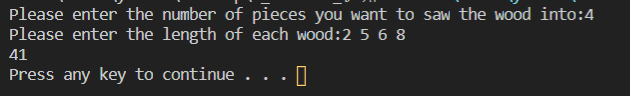
            st.Pop(p);

    }*//计算所有内部节点的和*

*return* sum;

### 3.3.3 求最小长度截屏示例

C:\Users\yixiao\AppData\Roaming\Tencent\Users\846507684\QQ\WinTemp\RichOle\MIRXT(RQYPJ3OD$`RGJE[]U.png



## 3.4 总体系统的实现

### 3.4.1 总体系统流程图



### 3.4.2 总体系统核心代码

LinkList list;

*while* (1) {

*if* (list.Initial(size)) {

*if* (size == 1) {

                cout << list.head->link->data << endl;

*break*;

            } *else* {

                HuffmanTree HT(list, size);

                minlength = HT.Minlength(size);//计算最小长

                cout << minlength << endl;

*break*;

            }

        } *else* {

            cout << "The data you entered is illegal. Please enter positive integers!" << endl;

            cout << "Please enter the length of each wood:";

*while* (getchar() != '\n')//清空输入

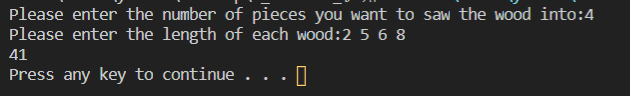
                ;

            cin.clear();

        }

### 3.4.3 总体系统截屏示例

C:\Users\yixiao\AppData\Roaming\Tencent\Users\846507684\QQ\WinTemp\RichOle\MIRXT(RQYPJ3OD$`RGJE[]U.png



# 4 测试

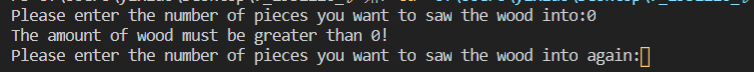
## 4.1 边界测试

### 4.1.1 初始化无输入数据

**测试用例：**初始无输入数据

**预期结果：**给出错误提示，程序运行正常不崩溃。

**实验结果：**

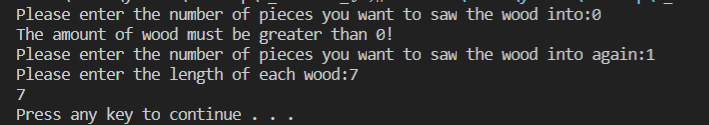


### 4.2.2 只有一个数据

**测试用例：**size为1

**预期结果：**程序正常运行，不崩溃。

**实验结果：**



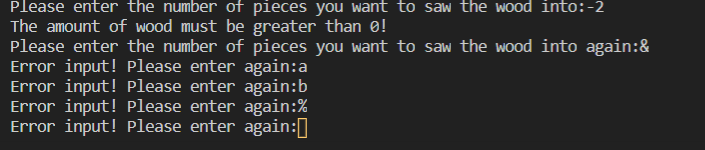
## 4.2 出错测试

### 4.2.1 木头数量错误

**测试用例：**输入木头数量为负数，abc，%&等字符

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**



### 4.2.2 木头长度存在错误

**测试用例：**木头长度错误

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

