项目7说明文档

数据结构课程设计

——修理牧场

作 者 姓 名： 谭欢秘

学 号： 1954185

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](#_Toc60248564)

[1.1 背景分析 1](#_Toc60248565)

[1.2 功能分析 1](#_Toc60248566)

[2 设计 1](#_Toc60248567)

[2.1 数据结构设计 1](#_Toc60248568)

[2.2 类结构设计 1](#_Toc60248569)

[2.3 成员与操作设计 2](#_Toc60248570)

[3.1.1 hmfNode类成员： 2](#_Toc60248571)

[3.1.2 HuffmanTree类成员： 2](#_Toc60248572)

[2.4 主函数设计 3](#_Toc60248573)

[3 实现 3](#_Toc60248574)

[3.1 前序遍历树功能的实现 3](#_Toc60248575)

[3.1.1 前序遍历树功能流程图 3](#_Toc60248576)

[3.1.2 前序遍历树功能核心代码 4](#_Toc60248577)

[3.1.3 preOrder()函数实现时需要注意的细节 4](#_Toc60248578)

[3.2 对还未加入树中节点权值排序功能的实现 4](#_Toc60248579)

[3.2.1 权值排序功能核心代码 4](#_Toc60248580)

[3.2.2 sort(int len)函数实现时需要注意的细节 5](#_Toc60248581)

[3.3 构造霍夫曼树功能的实现 5](#_Toc60248582)

[3.3.1 构造霍夫曼树功能流程图 5](#_Toc60248583)

[3.3.2 构造霍夫曼树功能核心代码 6](#_Toc60248584)

[3.4初始化功能的实现 6](#_Toc60248585)

[3.4.1 初始化功能流程图 6](#_Toc60248586)

[3.4.2 初始化功能核心代码 7](#_Toc60248587)

[3.4.3 initialize()函数实现时需要注意的细节 7](#_Toc60248588)

[3.5 主函数的实现 7](#_Toc60248589)

[3.5.1 主函数代码 7](#_Toc60248590)

[3.5.2 总体系统截屏示例 8](#_Toc60248591)

[4 测试 8](#_Toc60248592)

[4.1 功能测试 8](#_Toc60248593)

[4.1.1 初始化功能测试 8](#_Toc60248594)

[4.1.2森林排序、子树合并功能测试 8](#_Toc60248595)

[4.1.3 生成霍夫曼树功能测试 9](#_Toc60248596)

[4.2 边界测试 10](#_Toc60248597)

[4.2.1 输入木头总块数不合法 10](#_Toc60248598)

[4.2.2输入木头长度不合法 11](#_Toc60248599)

[5 总结 12](#_Toc60248600)

[5.1 遇到的错误 12](#_Toc60248601)

[5.1.1 xx函数中 12](#_Toc60248602)

[5.2 改进与优化 12](#_Toc60248603)

[5.3 项目心得 12](#_Toc60248604)

# 1 分析

## 背景分析

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，请人锯木的酬金等于所锯木头的长度。本项目企图找到一个最省钱的分法，就是要尽量让长度较长的木头被锯的次数尽量少。这样思考就会发现本题目很像字符编码时为了简便，需要将出现频率高的字符编码设置为最短；进一步抽象就是，让权值较大的外节点尽可能地靠近根节点。由此可发现，解决本题的核心就是霍夫曼树，实质就是构建带权路径最小的二叉树。

## 1.2 功能分析

求解砍木头最优惠的方法，需要能够正确接受输入的木头总块数、每块木头的长度并将其存储起来。然后根据构造霍夫曼树的方法构造一颗开销最小的二叉树，最后根节点的权值即使最小花费。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，本题的解决方法是构建一棵霍夫曼树，所以应选择一对多的树结构，且应是二叉树。

## 2.2 类结构设计

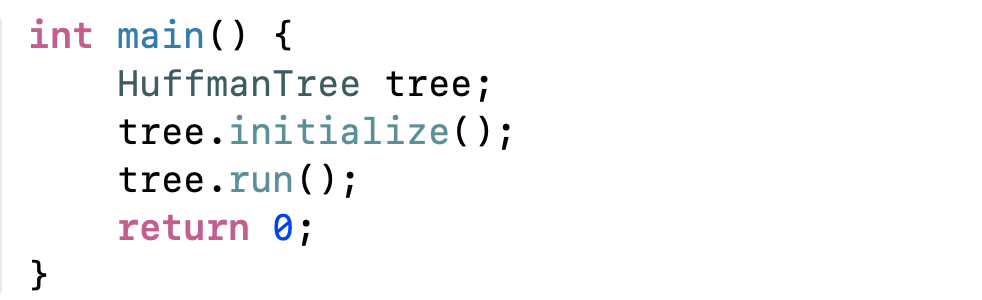
树一般包括两个抽象数据类型（ADT）——树结点类（hmfNode）与树类（HuffmanTree），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统将树类HuffmanTree类作为节点类hfmNode的友元，这样便于HuffmanTree类访问结点hmfNode类的私有成员。

## 2.3 成员与操作设计

### 3.1.1 hmfNode类成员：

### 3.1.2 HuffmanTree类成员：

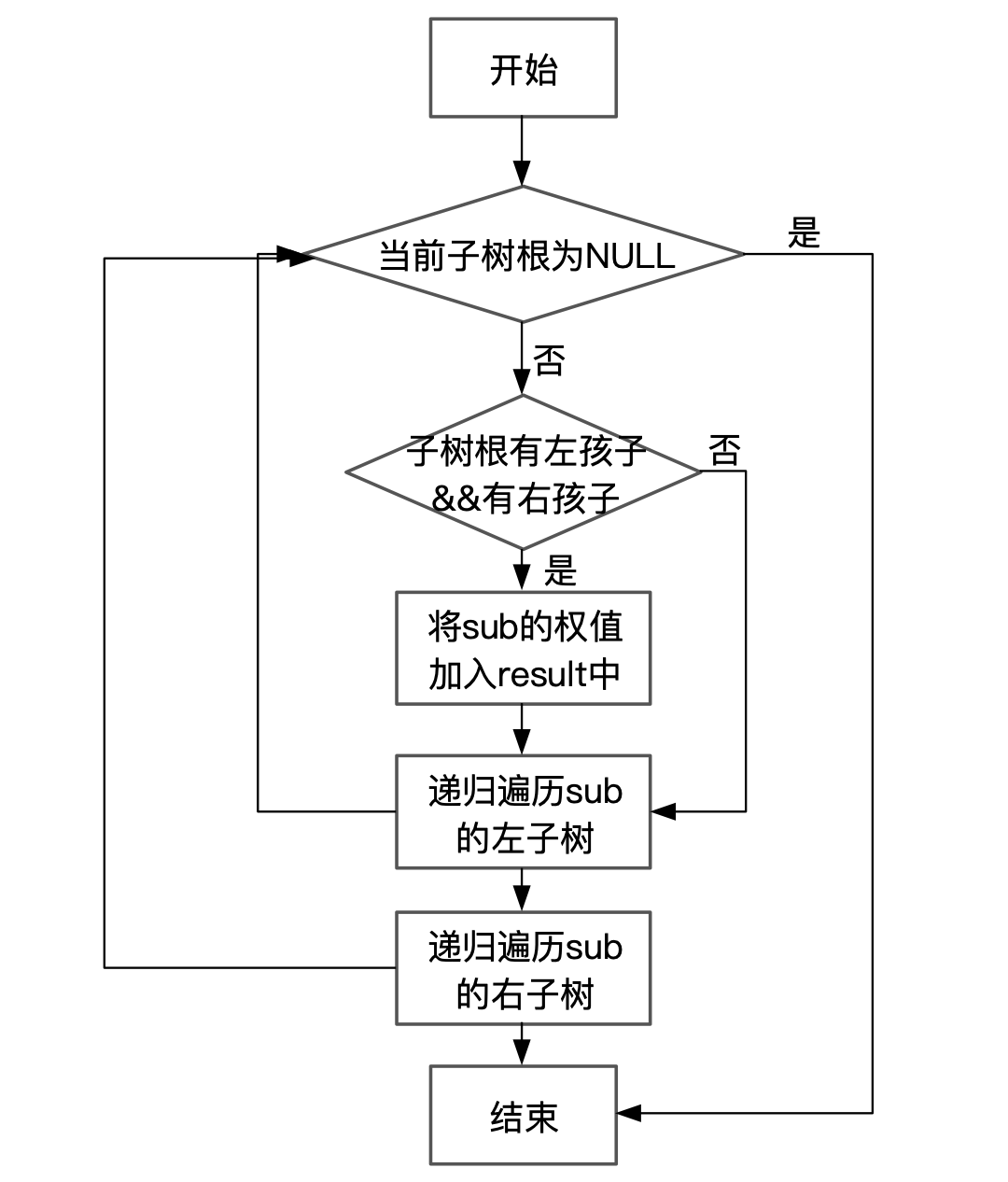
## 2.4 主函数设计

声明一个HuffmanTree类的对象tree。首先调用tree.initialize()函数实现初始化、输入木头块数长度的信息。然后调用tree.run()，构造霍夫曼树、输出最小开销。

# 3 实现

## 3.1 前序遍历树功能的实现

### 3.1.1 前序遍历树功能流程图



### 3.1.2 前序遍历树功能核心代码

### 3.1.3 preOrder()函数实现时需要注意的细节

根据霍夫曼树的特性，只有内节点是既有左孩子又有右孩子的。所以在遍历过程中检查一个节点是否有左右孩子就可判断它是不是内节点。如果是内节点，就将其权值加入result中。遍历结束就得到了霍夫曼树。

## 3.2 对还未加入树中节点权值排序功能的实现

### 3.2.1 权值排序功能核心代码



### 3.2.2 sort(int len)函数实现时需要注意的细节

按照根节点权值对森林里剩余的树进行冒泡排序，森林中共剩余len棵树，所以注意forest数组的范围是0～len-1。交换时直接交换forest中的元素（这些元素是每棵树根节点的指针，forest是一个指针的一维数组）。

## 3.3 构造霍夫曼树功能的实现

### 3.3.1 构造霍夫曼树功能流程图

### 3.3.2 构造霍夫曼树功能核心代码

## 3.4初始化功能的实现

### 3.4.1 初始化功能流程图

### 3.4.2 初始化功能核心代码

### 3.4.3 initialize()函数实现时需要注意的细节

木头总块数N设为double变量，取整后与N相减若不为零则表示输入的不是整数，需要提示用户重新输入。同样的，weight也需是double的一维数组，支持输入浮点型权值，但也要检查weight[i]是正数。

## 3.5 主函数的实现

### 3.5.1 主函数代码

### 3.5.2 总体系统截屏示例

# 4 测试

## 4.1 功能测试

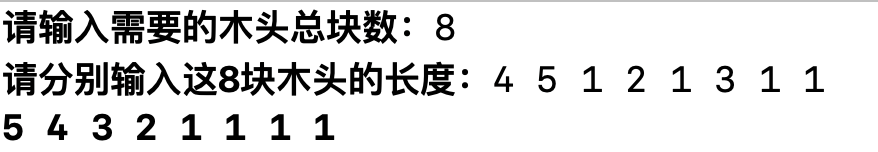
### 4.1.1 初始化功能测试

**测试用例**：

8块 长度：4 5 1 2 1 3 1 1

**预期结果**：

初始化、排序后森林中有8棵树，每棵树按根节点权值大小排列

**实验结果：**

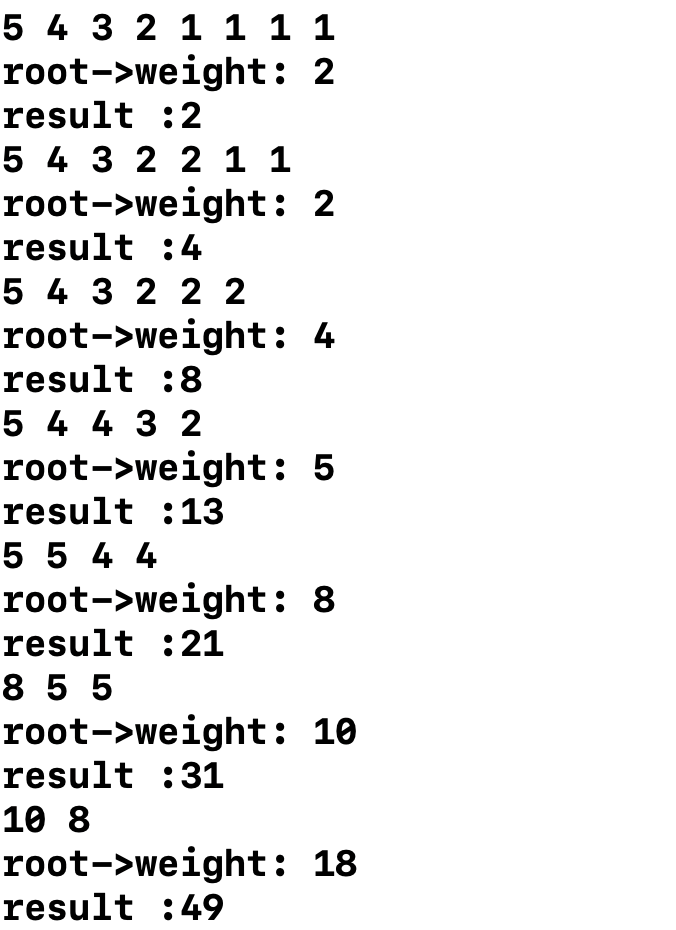
### 4.1.2森林排序、子树合并功能测试

**测试用例：**

8块 长度：4 5 1 2 1 3 1 1

**预期结果：**

每次最小的两棵树合并后从森林中删除，新的内部节点作为霍夫曼树的根加入森林

**实验结果：**

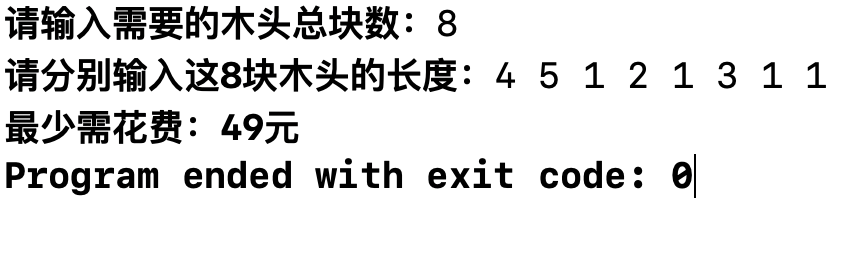
### 4.1.3 生成霍夫曼树功能测试

**测试用例1：**

8块 长度：4 5 1 2 1 3 1 1

**预期结果1：**

49

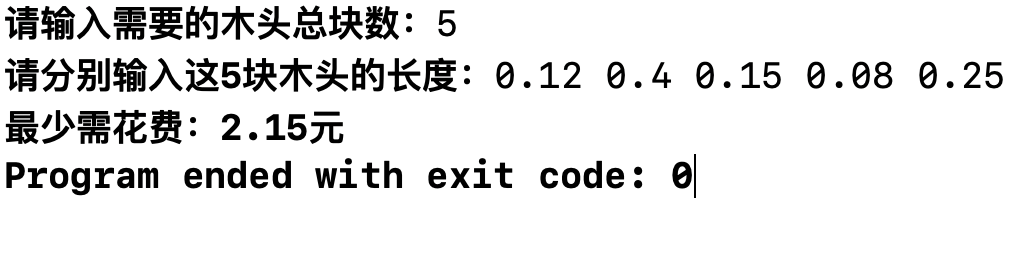
**实验结果1：**

**测试用例2：**

5块 长度：0.12 0.4 0.15 0.08 0.25

**预期结果2：**

2.15

**实验结果2：**

## 4.2 边界测试

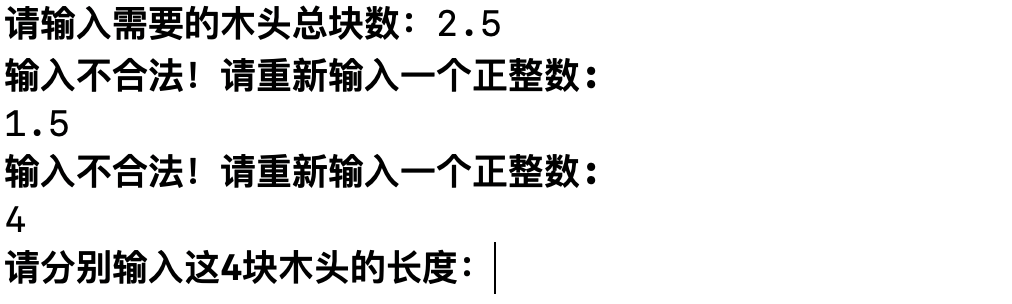
### 4.2.1 输入木头总块数不合法

**测试用例1：**

尝试输入若干非整数，如2.5、1.5

**预期结果1：**

立即给出错误提示，并要求用户重新输入一个正整数，程序运行正常不崩溃。

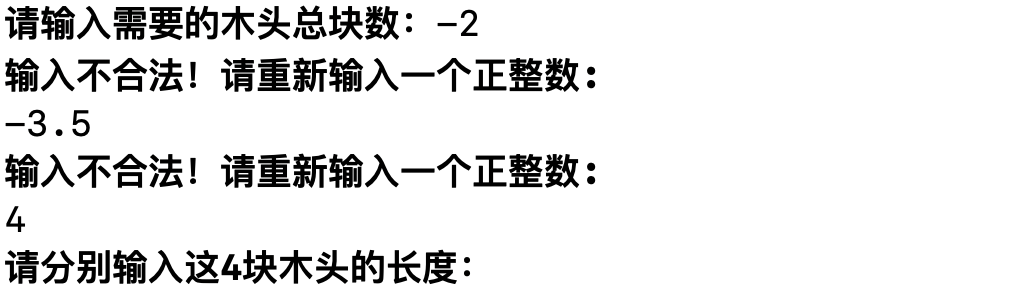
**实验结果1：**

**测试用例2：**

尝试输入若干负数，如-2、-3.5

**预期结果2：**

立即给出错误提示，并要求用户重新输入一个正整数，程序运行正常不崩溃。

**实验结果2：**

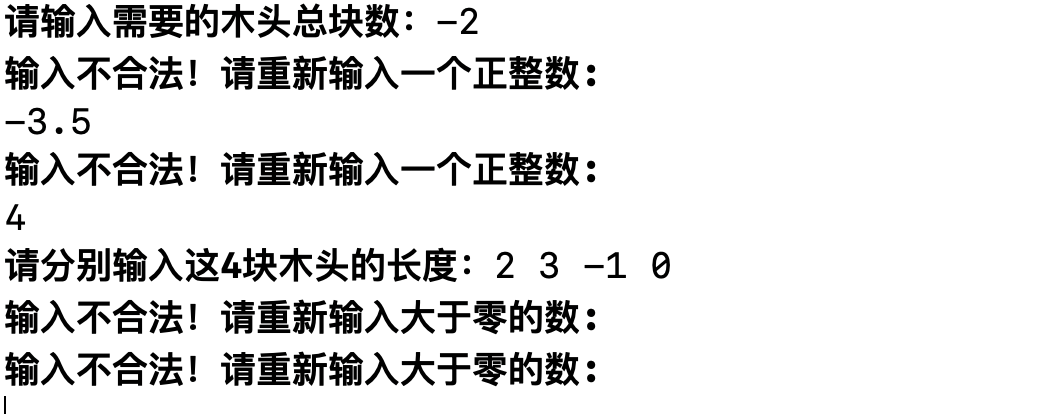
### 4.2.2输入木头长度不合法

**测试用例：**

尝试输入负数

**预期结果：**

立即给出错误提示，并要求用户重新输入正树，程序运行正常不崩溃。

**实验结果：**

# 5 总结

## 5.1 遇到的错误

### 5.1.1 xx函数中

## 5.2 改进与优化

## 5.3 项目心得