**项目说明文档**

**数据结构课程设计**

**——修理牧场**

作者姓名：香宁雨

学号：1954098

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.题目要求及功能分析 2](#_Toc58776417)

[2.设计 2](#_Toc58776418)

[2.1数据结构设计 2](#_Toc58776419)

[2.2类结构设计 3](#_Toc58776420)

[2.3类的成员与方法 3](#_Toc58776421)

[2.3.1node结构体 3](#_Toc58776422)

[2.3.2SortedList类 4](#_Toc58776423)

[2.4测试函数 4](#_Toc58776424)

[3.实现 5](#_Toc58776425)

[3.1SortedList实现 5](#_Toc58776426)

[3.1.1构造函数SortedList() 5](#_Toc58776427)

[3.1.2析构函数~SortedList() 5](#_Toc58776428)

[3.1.3插入方法push() 5](#_Toc58776429)

[3.1.4empty() 6](#_Toc58776430)

[3.1.5front() 7](#_Toc58776431)

[3.1.6pop\_front() 7](#_Toc58776432)

[3.2核心函数calculateMinCost() 7](#_Toc58776433)

[3.2.1函数介绍 7](#_Toc58776434)

[4.输出测试 8](#_Toc58776435)

[4.1题目输出 8](#_Toc58776436)

[4.2样例输出 8](#_Toc58776437)

[4.3错误输入 8](#_Toc58776438)

[4.3.1n值错误 8](#_Toc58776439)

[4.3.2length错误 9](#_Toc58776440)

# 1.题目要求及功能分析

**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）.

**项目功能要求**：（1）输入格式：输入第一行给出正整数N（N《104），表示要将木头锯成N块。第二行给出N个正整数，表示每块木头的长度。

（2）输出格式：输出一个整数，即将木头锯成N块的最小花费。

**功能分析**：我们需要找到一个数据结构使得结点的带权路径长度最小，结点的带权路径长度指的是从根节点到该节点的路径长度与该节点的路径长度与该结点的权的乘积。所以我们需要采用构建哈夫曼树的思路进行数据结构的设计。

# 2.设计

## 2.1数据结构设计

为得到结点的带权路径长度和最小，我们采用哈夫曼树的实现思路进行项目的实现。哈夫曼树有多种实现方法，最常见的一种为，从n个权值中选出两个最小的权值，对应的两个结点组成一个新的二叉树，且新二叉树的根节点的权值为左右两个孩子权值的和，在原有的n个权值中删除刚才已经插入树中的两个最小权值，同时将新的权值加入到n-2的权值行列中，以此类推最终形成一个树状结构。

因为多次形成树和多次进行搜索会有较高的时间复杂度，所以本项目通过采用有序链表这种数据结构在插入时就进行排序，并模拟哈夫曼树的形成过程进行数据的处理，可以在使用更简单数据结构的基础上实现更低的时间复杂度。

## 2.2类结构设计

有序链表也是链表的一种，区别在于他在插入时是有序的，新插入的结点会与各个节点进行比较并插入其顺序位置上，为了便于进行插入和删除操作，本项目采用双向链表进行数据的储存。

## 2.3类的成员与方法

### 2.3.1node结构体



### 2.3.2SortedList类



2.4测试函数



# 3.实现

## 3.1SortedList实现

### 3.1.1构造函数SortedList()



### 3.1.2析构函数~SortedList()



### 3.1.3插入方法push()

为了实现SortedList的有序排列，我们采用在插入时进行比较的方法来生成顺序链表。



### 3.1.4empty()

SortedList用empty()方法判断当前链表是否为空。



### 3.1.5front()

通过调用front()方法获取当前链表的最小值，并将值返回进行处理。



### 3.1.6pop\_front()

将有序链表的头节点删除，即将链表中最小值删除。



## 3.2核心函数calculateMinCost()

### 3.2.1函数介绍

将链表中最小的两个值弹出，进行相加，再将相加的值放入有序链表中，并且使当前计算的ans值加上当前所得到的值，当链表只有一个元素时当前值即为所得值。



# 4.输出测试

## 4.1题目输出



## 4.2样例输出



## 4.3错误输入

### 4.3.1n值错误



### 4.3.2length错误

