**数据结构课程设计**

**项目说明文档**

**修理牧场**

作 者 姓 名： 苏家铭

学 号： 2151299

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 项目分析 1](#_Toc32634)

[1.1 项目背景 1](#_Toc6933)

[1.2项目要求 1](#_Toc1603)

[1.2.1 功能要求 1](#_Toc24808)

[1.2.2 输入要求 1](#_Toc14809)

[1.2.3 输出要求 2](#_Toc31771)

[1.2.4 项目实例 2](#_Toc30467)

[2 项目设计及实现 2](#_Toc1646)

[2.1 数据结构设计思路 2](#_Toc18489)

[2.2 类设计 2](#_Toc8547)

[2.2.1 堆 2](#_Toc11824)

[2.2.1 优先级队列 4](#_Toc8984)

[2.3 项目算法 5](#_Toc1511)

[2.3.1 实现思路 5](#_Toc7832)

[2.3.2 代码实现 7](#_Toc16408)

[3 项目测试 8](#_Toc19470)

[3.1 一般情况 8](#_Toc32058)

[3.2 花费各不相同的情况 8](#_Toc14147)

[3.3 花费全部相同的情况 8](#_Toc31342)

[3.4 只有两个木块的情况 9](#_Toc3912)

[3.5 只有一个木块的情况 9](#_Toc24344)

[3.6 输入N为非法字符的情况（健壮性） 9](#_Toc2113)

[3.7 输入木块长度为非法字符的情况（健壮性） 10](#_Toc14317)

[4 算法性能分析 10](#_Toc2212)

[4.1 正确性 10](#_Toc5051)

[4.2 可使用性 10](#_Toc25971)

[4.3 可读性 10](#_Toc5950)

[4.4 效率 11](#_Toc19765)

[4.5 健壮性 11](#_Toc18613)

# 1 项目分析

## 项目背景

哈夫曼树(Huffman Tree)是由 n 个带权叶子结点构成的所有二叉树中带权路径长度最短的二叉树，又称最优二叉树。哈夫曼编码是哈夫曼树的一个应用。在通信工程中，通常需要将传送的文字转换成由二进制字符组成的二进制串，这一过程被称为编码。不同字符出现的概率不一样，在传送电报时，总是希望电报尽可能短，那么采用哈夫曼编码构造每个字符的最佳2元前缀码写成的电报的总长将会是最短的。

农夫修理牧场问题则是构造哈夫曼树算法中一道经典的问题，在计算最小花费的同时运用了哈夫曼树的构造思想。完成本项目对于哈夫曼树的理解和构造哈夫曼树有着不一般的意义。

## 1.2项目要求

### 1.2.1 功能要求

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）。

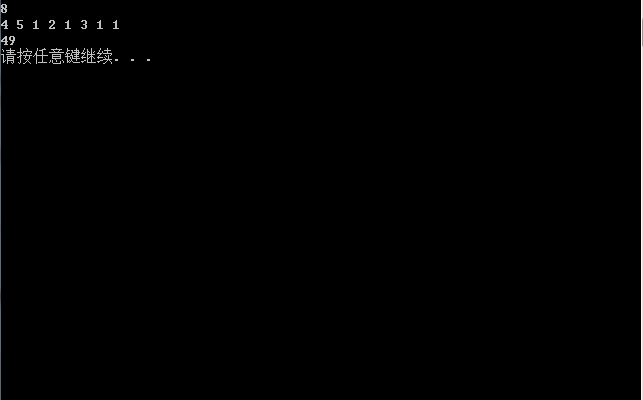
### 1.2.2 输入要求

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。

### 1.2.3 **输出要求**

输出一个整数，即将木头锯成N块的最小花费。

### 1.2.4 项目实例



# 2 **项目设计及实现**

## 2.1 **数据结构设计思路**

根据项目要求，需要在一众元素中选取最小权值的元素，那么自然联想到最小优先级队列这一数据结构。最小优先级队列的排序可以用普通的链表进行，此方法的时间复杂度为O(n)，为此，本项目基于最小堆实现了最小优先级队列，使选取最小元素的算法时间复杂度降为O(log2n).

## 2.2 **类设计**

### 2.2.1 堆

堆(heap):若有一个关键码的集合K = {k0，k1，k2，…，kn-1}，把它的所有元素按[完全二叉树](https://so.csdn.net/so/search?q=%E5%AE%8C%E5%85%A8%E4%BA%8C%E5%8F%89%E6%A0%91&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/zgdlxs/article/details/_blank)的顺序存储方式存储在一个一维数组中，并满足：Ki <= K2i+1 且 Ki <= K2i+2，则称为小堆(或大堆)。将根节点最大的堆叫做最大堆或大根堆，根节点最小的堆叫做最小堆或小根堆。

本项目中堆基于一维数值来实现，通过定义建堆、调整堆来完成了一个堆。

最小堆类数据成员：

template <class T>

T\* pheap;//存放小根堆中元素的数组

int current\_size;//小根堆中当前元素个数

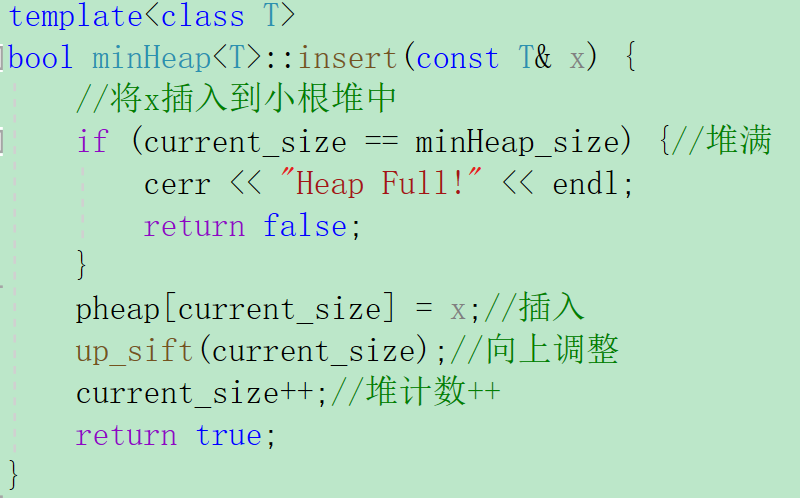
int minHeap\_size;//小根堆最多允许元素个数

该最小堆已经实现8种功能：向上调整、向下调整、插入元素、删除堆顶元素、判空、判满、置空堆、堆排序。

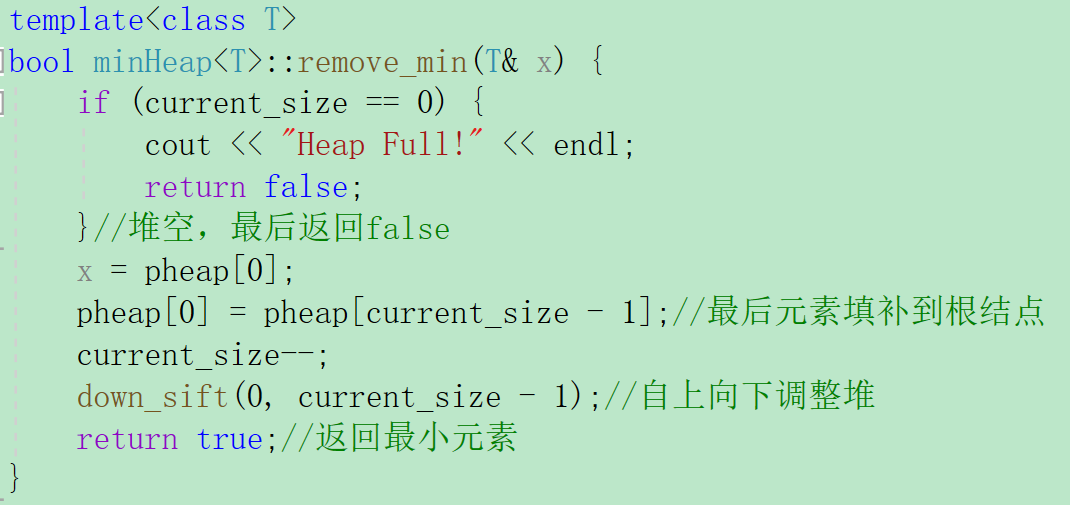
具体函数声明为：

* void down\_sift(int start, int m);//自上向下调整
* void up\_sift(int start);//自下向上调整
* void makeEmpty();//置空
* bool insert(const T& x);//将x插入到小根堆中
* bool remove\_min(T& x);//删除小根堆上的最小元素
* bool empty()const ;//判断堆是否为空
* bool full()const ;//判断堆是否达到最大限制个数
* void make\_empty();//置空堆
* void rank\_heap(T rank[]);//堆排序

插入算法代码实现：



删除算法代码实现：



主要算法的时间复杂度：

* 插入：O(log2n)
* 删除：O(log2n)
* 置空：O(n)
* 判断空否：O(1)

在最后为了能在优先级队列中访问堆的数据成员和成员函数，将优先级队列声明为友元函数：

//友元类

template <class E>

friend class minPQueue;

### 2.2.1 优先级队列

一般的队列是一种特征为FIFO的特殊线性表，但是在实际情况中通常需要选取优先级最高的元素先出队列。为此，本项目实现了一种每次从队列中取出的应该是具有最高优先权的元素的队列，也即优先级队列(Priority Queue)

本项目中最小优先级队列基于小根堆来实现，每次出队则返回选取小根堆的堆顶元素并删除调整，完成了优先级队列的出队过程。

最小优先级队列类数据成员：

template <class T>

minHeap<T> heap;

与一般队列类似，该最小优先级队列也实现6种功能：入队、出队、置空、判空、返回队首元素、返回队列长度。

具体函数声明为：

* void push(const T& val);//入队
* void pop();//出队
* void makeEmpty();//置空
* bool empty()const;//判断空否
* int size()const;//返回队列大小
* const T& front();//返回队首元素

主要算法的时间复杂度：

* 入队：O(log2n)
* 出队：O(log2n)
* 置空：O(n)
* 判断空否：O(1)
* 返回队首元素：O(1)
* 返回队列大小：O(1)

## 2.3 **项目算法**

### 2.3.1 实现思路

构造哈夫曼树的算法步骤：

① 初始化：用给定的 n 个权值{w1,w2,…,wn}构造 n 棵二叉树并构成的森林F={T1,T2,…,Tn}，其中每一棵二叉树Ti（1<=i<=n）都只有一个权值为 wi 的根结点，其左、右子树为空。

② 找最小树：在森林 F 中选择两棵根结点权值最小的二叉树，作为一棵新二叉树的左、右子树，标记新二叉树的根结点权值为其左、右子树的根结点权值之和。

③ 删除与加入：从 F 中删除被选中的那两棵二叉树，同时把新构成的二叉树加入到森林 F 中。

④ 判断：重复②、③操作，直到森林中只含有一棵二叉树为止，此时得到的这棵二叉树就是哈夫曼树。

简单的说就是先选择权小的，所有权小的结点被放置在树的较深层，而权较大的离根较近，这样一来所构成的哈夫曼树就具有最小带权路径长度。

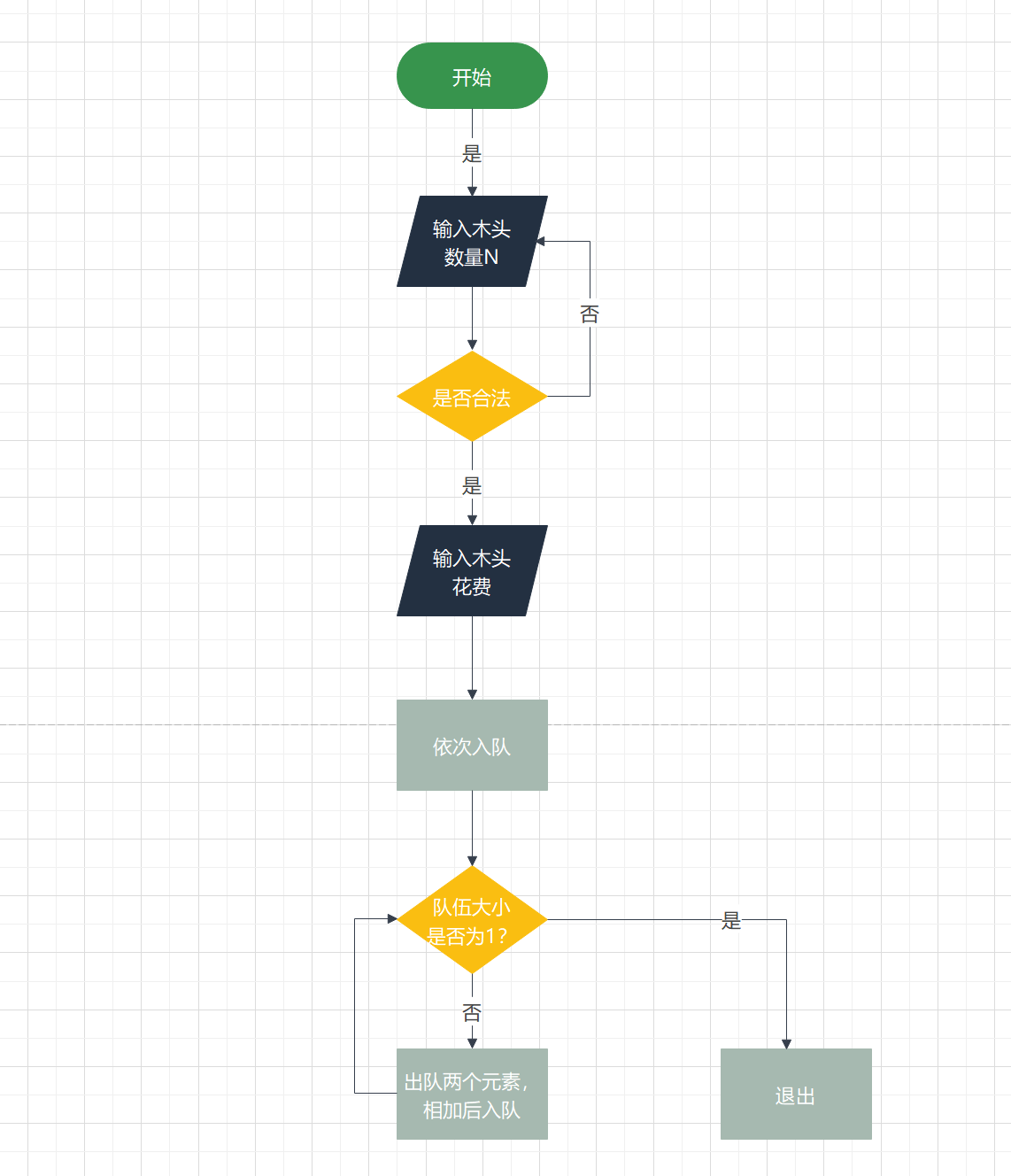
修理牧场问题的求解则类似于构造哈夫曼树算法：

1. 初始化：将每一个木头的花费权值加入最小优先级队列

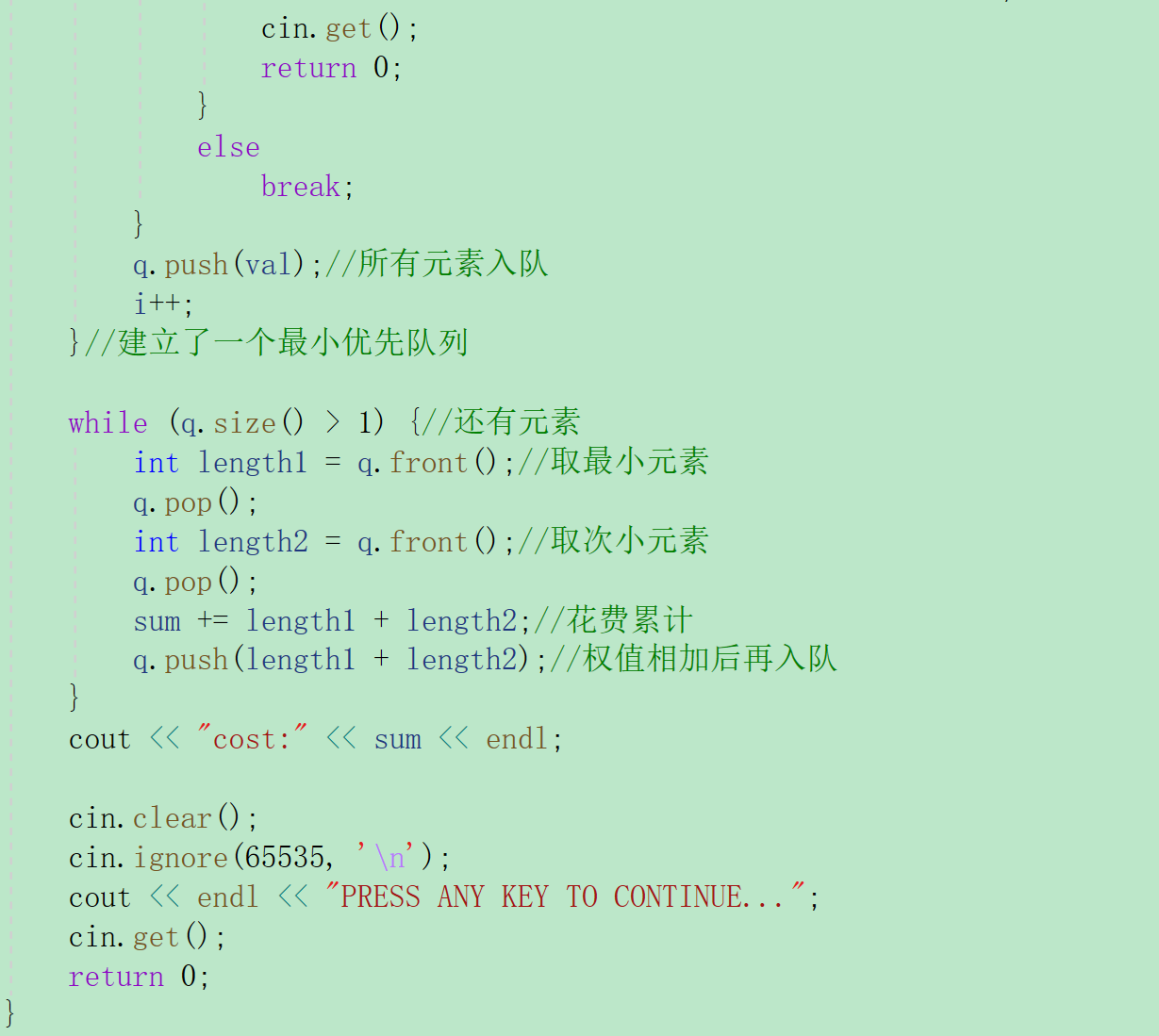
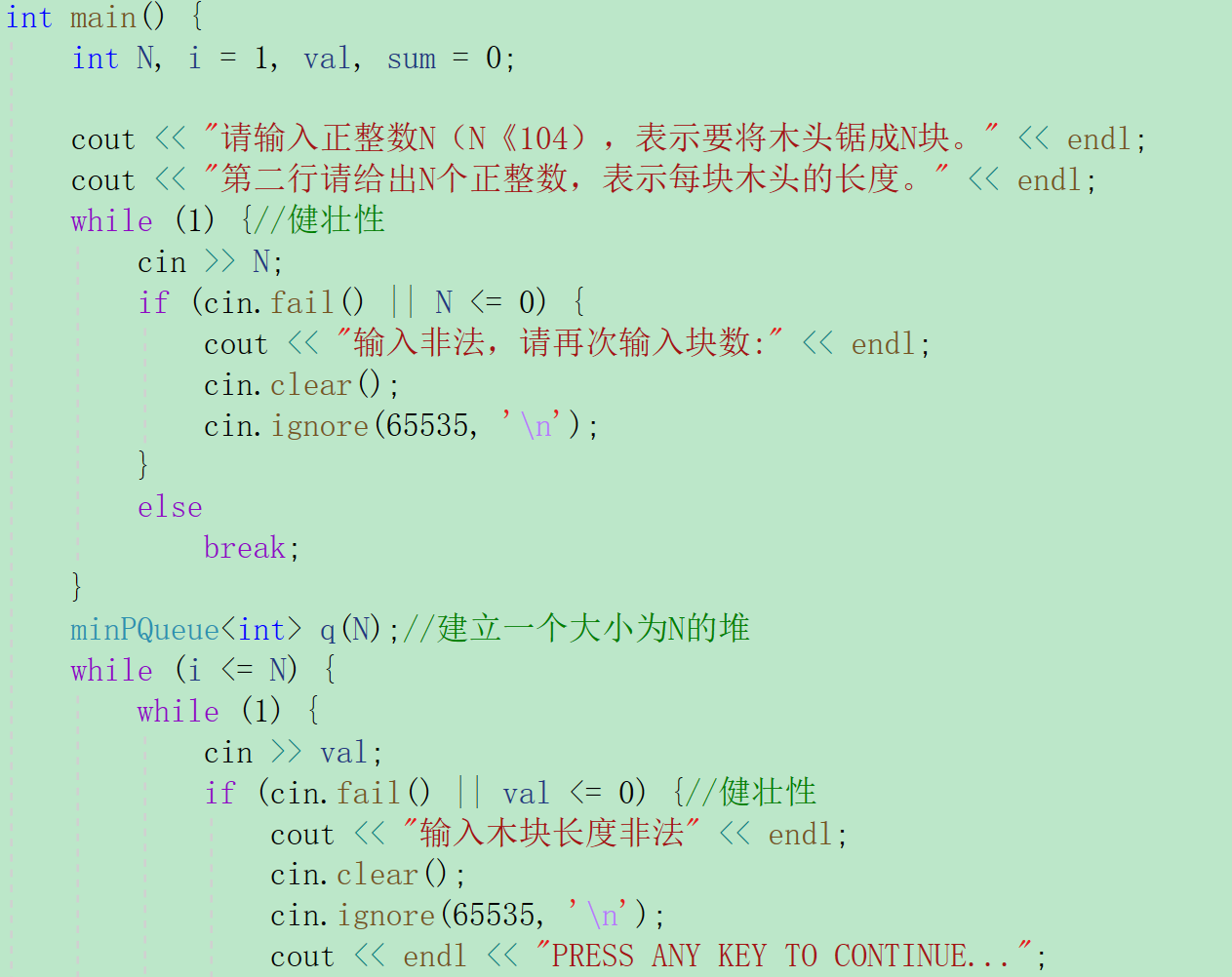
② 找最小树：在队列中选择两个权值最小的元素，作为一个新的元素。该元素的权值为前两个元素权值之和。

③ 删除与加入：从队列中删除被选中的那个元素，同时把新构成的元素加入到队列中。

④ 判断：重复②、③操作，直到为止队列中只剩下一个元素为止，此时得到的这个元素的权值就是最小花费。



### 2.3.2 代**码实**现



# 3 项目测试

## 3.1 **一般情况**

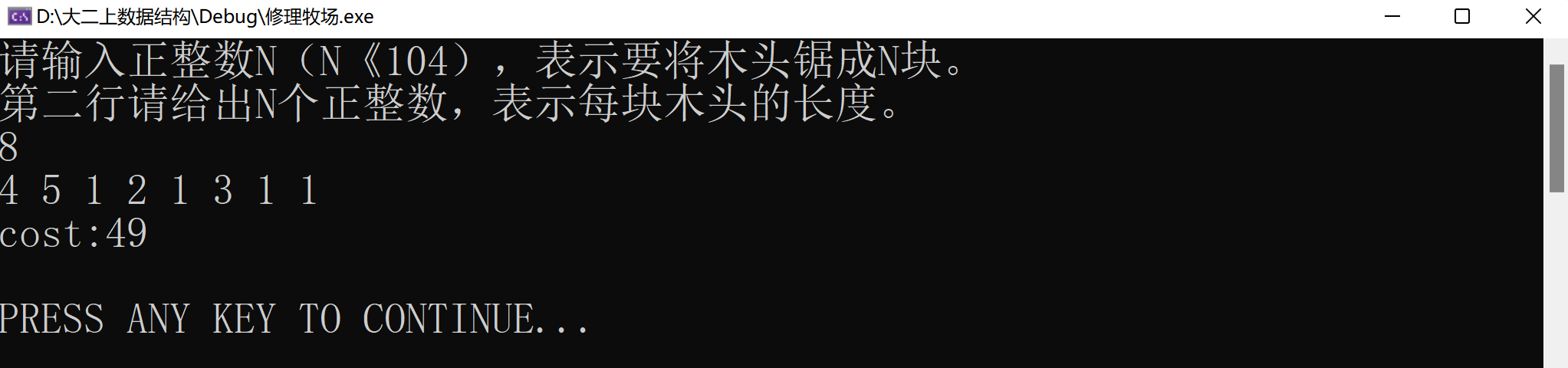
测试输入：

8

4 5 1 2 1 3 1 1

预期输出：49

实验结果：



## 3.2 花费各不相同的情况

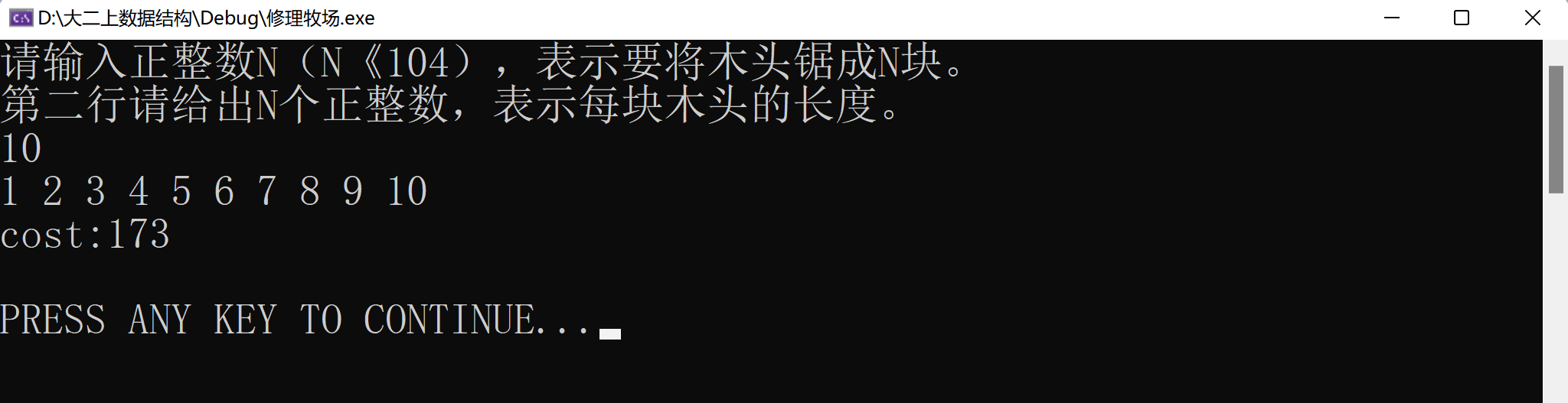
测试输入：

10

1 2 3 4 5 6 7 8 9 10

预期输出：173

实验结果：



## 3.3 花费全部相同的情况

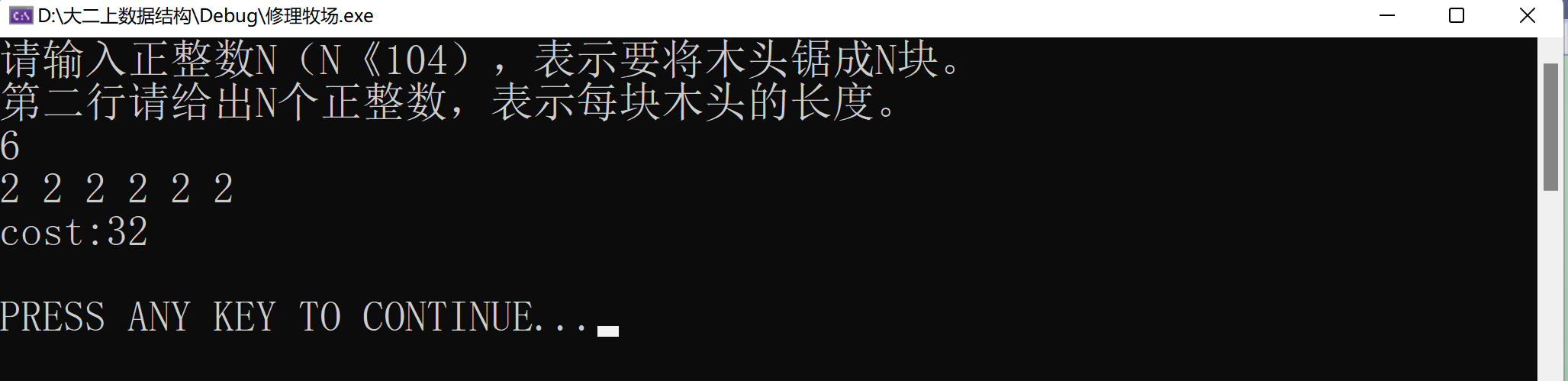
测试输入：

6

2 2 2 2 2 2

预期输出：32

实验结果：



## 3.4 只有两个木块的情况

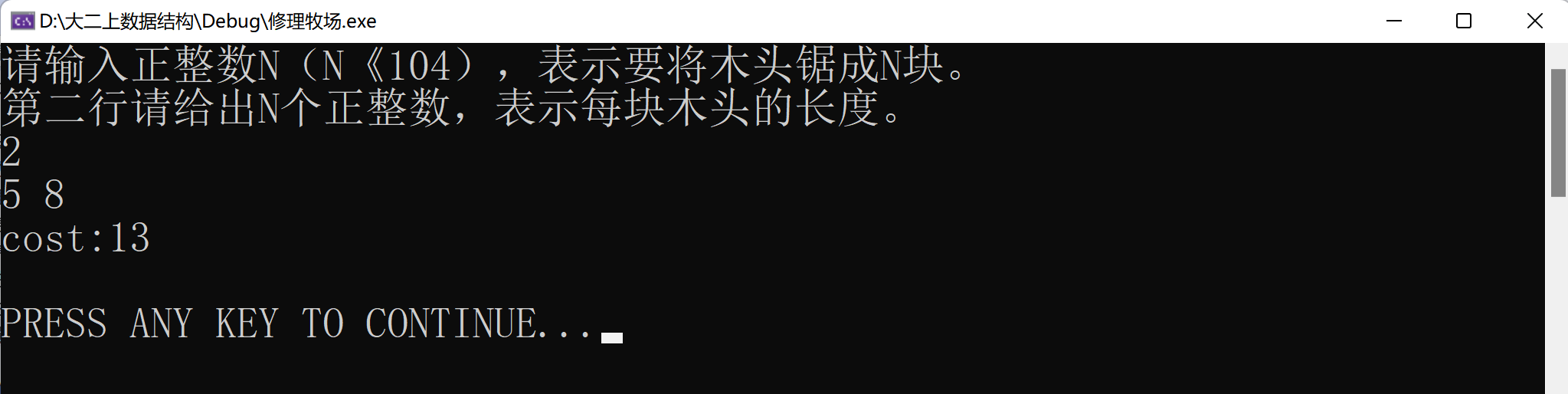
测试输入：

2

5 8

预期输出：13

实验结果：



## 3.5 只有一个木块的情况

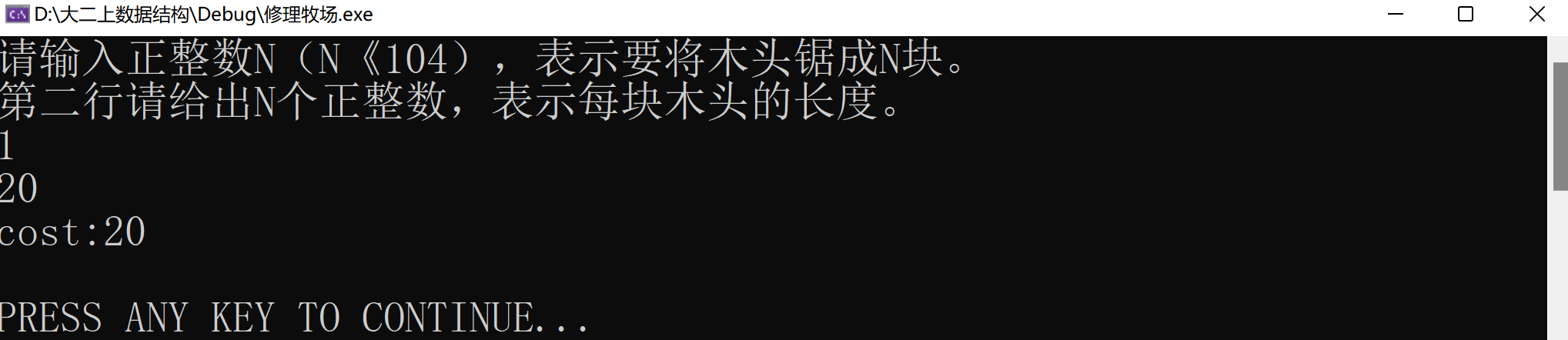
测试输入：

1

20

预期输出：20

实验结果：



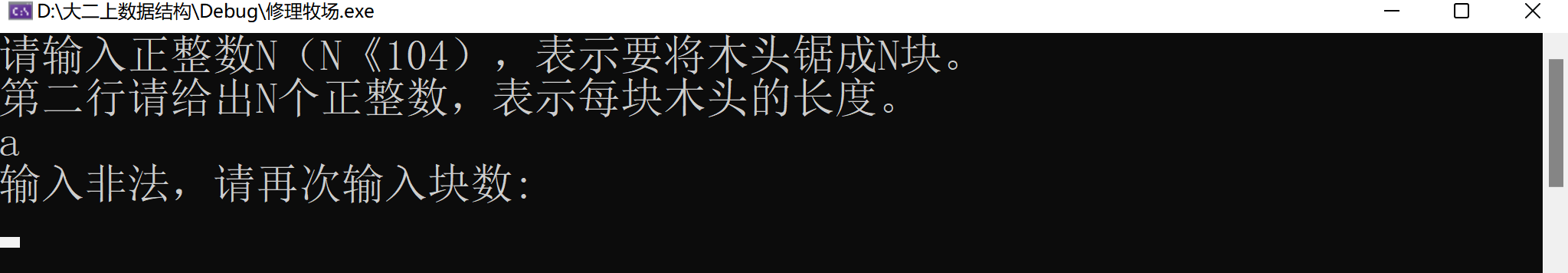
## 3.6 输入N为非法字符的情况（健壮性）

测试输入：

a

预期输出：错误提示+重新输入

实验结果：



## 3.7 输入木块长度为非法字符的情况（健壮性）

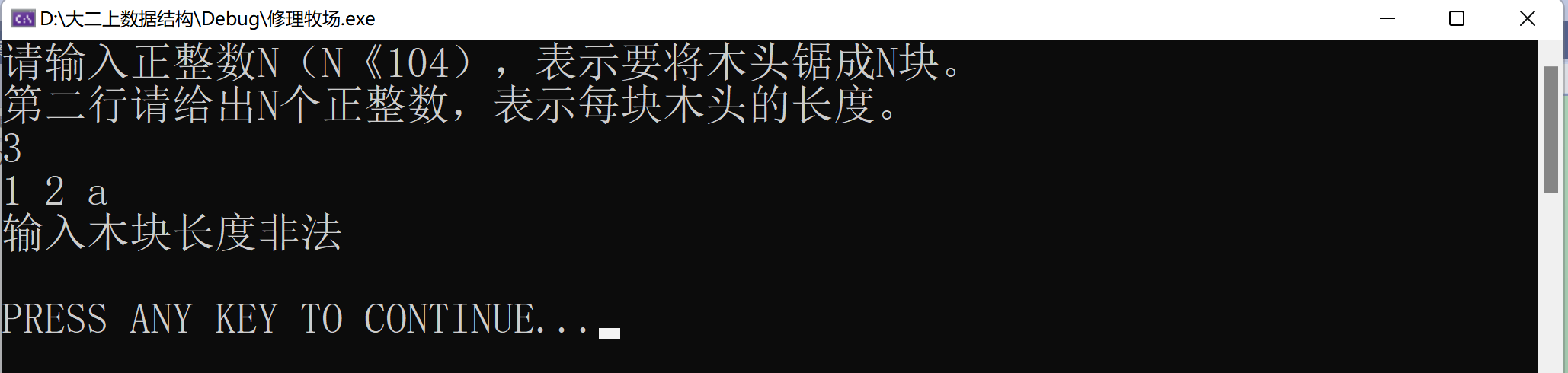
测试输入：

3

1 2 a

预期输出：错误提示+退出

实验结果：



# 4 算法性能分析

## 4.1 正确性

本算法能正确地执行预定的功能和性能要求，由大量的实验结果和人工计算的结果相匹配后都满足一致，满足正确性。

## 4.2 可使用性

本算法可以很方便地使用，求最小花费只需要用户输入木块总数和各个木块长度即可求解。并且该算法有良好的界面和完备的用户文档。没有使用公用变量或全局变量。

## 4.3 可读性

本算法逻辑清晰、简单、且结构化，所有命名与函数名都具有实际含义，让人见名知义。且算法中包含了大量注释，简要说明了算法功能、输入与输出参数的使用规则、重要数据的作用、算法中各程序段完成的功能。

## 4.4 效率

哈夫曼树的构造可以采取从数组里面取最小值，但是这样做的话，每次插入元素到数组中，因为要维持升序，它会花费很多时间在维护上，时间复杂度为O(n)。于是我们可以采用最小堆/最小优先级队列，堆的插入操作时间复杂度为O(log2n),比起数组还是快很多的。

哈夫曼树的总体时间复杂度为：创建最小堆(O(n))+插入n-1个数进入堆中(O(nlog2n))+从堆中要删除2n-1个结点(O(nlog2n))=O(nlog2(n))。

而本项目中修理牧场的问题也近似于构造哈夫曼树问题，时间复杂度优化为O(nlog2(n))。

## 4.5 健壮性

本算法对于边界条件，诸如：只有两快木头、只有一块木头都能有正确的计算，并且对于堆中新结点的申请失败、输入木块数量非法的情况也有相应的错误提示。