项目说明文档

数据结构课程设计

——修理牧场(HuffmanTree)

作 者 姓 名： 王星洲

学 号： 1652977

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](#_Toc531715473)

[1.1 背景分析 1](#_Toc531715474)

[1.2 功能分析 1](#_Toc531715475)

[2 设计 1](#_Toc531715476)

[2.1 数据结构设计 1](#_Toc531715477)

[2.2 优先级队列的使用 1](#_Toc531715478)

[2.3 流程设计 2](#_Toc531715479)

[3 实现 2](#_Toc531715480)

[3.1优先级队列的创建 2](#_Toc531715481)

[3.1.1 主要代码 2](#_Toc531715482)

[3.1.2 注意事项 2](#_Toc531715483)

[3.2 接收用户的输入 3](#_Toc531715484)

[3.2.1用户输入过程流程图 3](#_Toc531715485)

[3.2.2核心代码 4](#_Toc531715486)

[3.2.3 人机交互截屏示例 5](#_Toc531715487)

[3.2.4 人机交互优势及注意事项 6](#_Toc531715488)

[3.3 求最优解功能的实现 8](#_Toc531715489)

[3.3.1 求最优解功能流程图 8](#_Toc531715490)

[3.3.2 求最优解功能核心代码 9](#_Toc531715491)

[3.3.3 求最优解功能亮点及注意事项 9](#_Toc531715492)

[4 测试 10](#_Toc531715493)

[4.1 功能测试 10](#_Toc531715494)

[4.1.1 一般功能测试 10](#_Toc531715495)

[4.1.2 数字全部相同功能测试 10](#_Toc531715496)

[4.1.3 题例测试 10](#_Toc531715497)

[4.2 边界测试 11](#_Toc531715498)

[4.2.1 最小N测试 11](#_Toc531715499)

[4.3 出错测试 11](#_Toc531715500)

[4.3.1 输入N非法或不在范围内 11](#_Toc531715501)

[4.3.2 输入length非法或不在范围内 11](#_Toc531715502)

[4.3.3 输入超过规定 12](#_Toc531715503)

[5 亮点 12](#_Toc531715504)

[5.1 寻找最优解算法亮点 12](#_Toc531715505)

[5.2 人机交互亮点 12](#_Toc531715506)

[5.3 优先级序列创建时的注意事项及难点 12](#_Toc531715507)

# 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 功能分析

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。

# 2 设计

## 2.1 数据结构设计

经过分析，该问题在考查霍夫曼树，每一次提取出数列之中最小的两个值作和作为新的元素加入数列，并且移去这两个元素，最后把所有移去元素求和得到结果。因为要取数列中的最小值，我们考虑使用堆或者优先级队列来解决，本题我采用优先级队列priority-queue解决问题。建立一个小顶优先级队列q(greater priority-queue)储存用户输入的长度，接下来使用队列操作求出最后的结果。

## 2.2 优先级队列的使用

采用#include<queue>使用STL库提供的标准队列，其中包含优先级队列，值得注意的是小顶优先级队列并非默认，使用时需要greater<T>模板，使用#include <functional>可以获得这个模板，以此来使用小顶优先级队列。在使用时，我们需要用到top()函数获得队列顶的值，本题中即为最小值，再使用pop()函数将其移出队列。

## 2.3 流程设计

本程序首先创建一个空的小顶优先级队列q，接收用户输入的数量N和长度length，在输入过程中进行合法性检测，保证程序的健壮性，接下来将length存入队列q，接下来对q进行操作，当q的元素超过1个时，将最小的两个元素相加，结果作为队列的新元素，并将其结果与最终的结果sum相加(sum初始值为0)，操作结束后，将这两个值移出队列，循环结束后，sum的值即为所求。

# 3 实现

## 3.1优先级队列的创建

### 3.1.1 主要代码

#include <iostream>

//为了使用greater模板

#include <functional>

#include <queue>

using namespace std;

//定义优先级队列，greater<int>表示小顶

priority\_queue<int, vector<int>, greater<int> > q;

### 3.1.2 注意事项

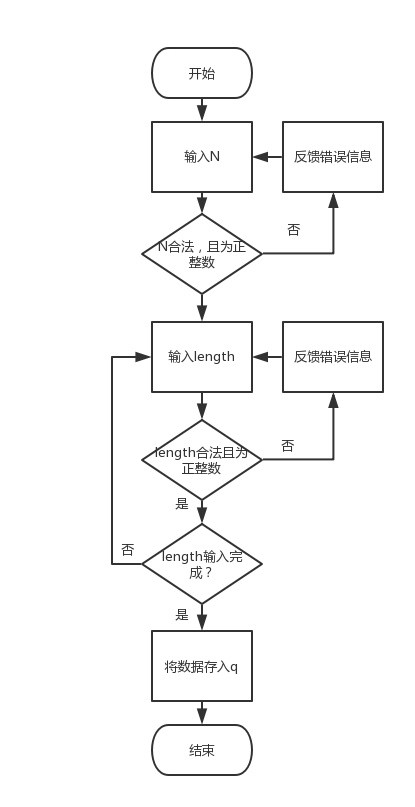
1.注意包含头文件<queue>

2.为了使用greater<int>模板需要头文件<functional>，因为优先级队列默认为大顶，而本题需要小顶优先级队列。

3.优先级队列定义要注意< int, vector<T>, function<T> >默认为< T, vector<T>, less<T> >，当我们确认了类型之后，如果我们需要使用最小堆，需要自行使用greater<T>模板。

## 3.2 接收用户的输入

### 3.2.1用户输入过程流程图



### 3.2.2核心代码

//N姐搜用户输入的数量，length接收输入的长度

int N, length;

//对于N的输入

while (1)

{

//N输入失败

if (!(cin >> N))

{

cout << "输入非法，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

continue;

}

//N不是正整数

else if (N <= 0)

{

cout << "请输入一个正整数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

continue;

}

//N输入正确

else

{

break;

}

}

//接收length

for (int i = 0; i < N; i++)

{

//length输入失败

while (!(cin >> length))

{

cout << "输入非法，已知您要输入" << N << "个长度，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

//清空q队列

while (!q.empty())

{

q.pop();

}

//i置回0

i = 0;

}

if (length <= 0)

{

cout << "长度必须为正整数，已知您要输入" << N << "个长度，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

//清空q队列

while (!q.empty())

{

q.pop();

}

//将i置为-1，再次进入循环时i变回0

i = -1;

}

else

{

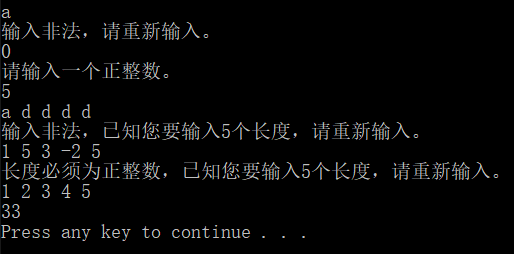
//输入正确，将length加入q队列

q.push(length);

}

}

### 3.2.3 人机交互截屏示例



### 3.2.4 人机交互优势及注意事项

1.对于N有合法检测和非正整数检测

//N输入失败

if (!(cin >> N))

{

cout << "输入非法，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

continue;

}

//N不是正整数

else if (N <= 0)

{

cout << "请输入一个正整数。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

continue;

}

//N输入正确

else

{

break;

}

因为N是int类型，所以输入不合法的值会进入if语句，提示用户重新输入，同时清空输入缓存区，更改cin标志，为下次输入做准备，这样做大大提升了用户的体验，防止程序出现问题。

2.对于length同样有合法检测和非正整数检测

for (int i = 0; i < N; i++)

{

//length输入失败

while (!(cin >> length))

{

cout << "输入非法，已知您要输入" << N << "个长度，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

//清空q队列

while (!q.empty())

{

q.pop();

}

//i置回0

i = 0;

}

if (length <= 0)

{

cout << "长度必须为正整数，已知您要输入" << N << "个长度，请重新输入。" << endl;

cin.clear();

while (cin.get() != '\n')

{

continue;

}

//清空q队列

while (!q.empty())

{

q.pop();

}

//将i置为-1，再次进入循环时i变回0

i = -1;

}

else

{

//输入正确，将length加入q队列

q.push(length);

}

}

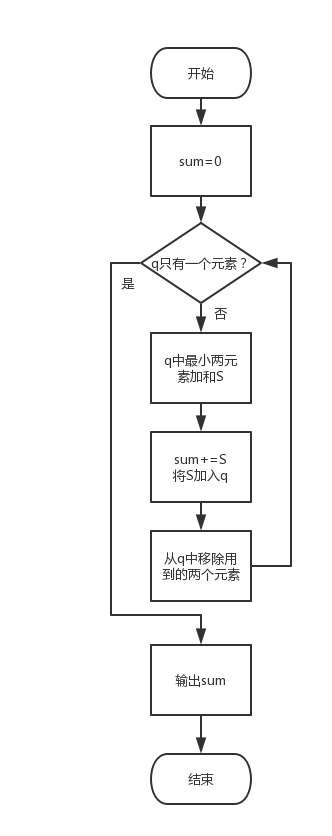
其中有关合法性判断以及清空输入流的操作与N的检测相同，特别的是，当length出错，需要同时清空队列，并且对于循环指示变量i进行操作，保证输入正常进行。

3.注意，检测合法代码中需要将输入流清空，否则将会在输出语句出现死循环

4.length出错务必将队列清空。否则会让之前的错误数据污染队列。

## 3.3 求最优解功能的实现

### 3.3.1 求最优解功能流程图



### 3.3.2 求最优解功能核心代码

//结果

int sum = 0;

//当q的元素个数大于1

while (q.size() > 1) {

//取出q中最小的两个值与sum相加作为sum，把结果加入队列，再将这两个值移出队列

int first = q.top();

q.pop();

int second = q.top();

q.pop();

sum += first + second;

q.push(first + second);

}

//输出sum

cout << sum << endl;

### 3.3.3 求最优解功能亮点及注意事项

1.循环结束的条件是q.size()==1，因为只有一个元素时，说明木块已经按要求切完了，而最后的那个元素值与本题无关，同时如果N=1，直接输出0也符合题意。

2.优先级序列不可以使用front(),back()函数，但可以使用top()函数获得最小(大)值，在于最开始定义时使用less还是greater模板。

3.在计算完两个最小值和之后，要把这两个值的和作为新的元素加入到队列中，使用q.push(first+second)实现。

# 4 测试

## 4.1 功能测试

### 4.1.1 一般功能测试

**测试用例**：

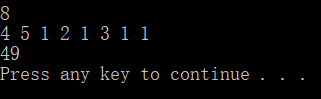
输入 8

输入 4 5 1 2 1 3 1 1

**预期结果**：

输出49

**实验结果**



### 4.1.2 数字全部相同功能测试

**测试用例：**

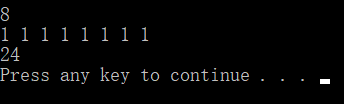
输入 8

输入 1 1 1 1 1 1 1 1

**预期结果：**

输出 24

**实验结果：**



### 4.1.3 题例测试

**测试用例：**

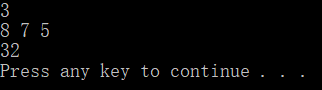
输入 3

输入 8 7 5

**预期结果：**

输出 32

**实验结果：**



## 4.2 边界测试

### 4.2.1 最小N测试

**测试用例：**

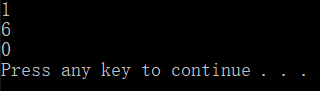
输入 1

输入 6

**预期结果：**

0

**实验结果：**



## 4.3 出错测试

### 4.3.1 输入N非法或不在范围内

**测试用例：**输入N非法，输入N为0

**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：**



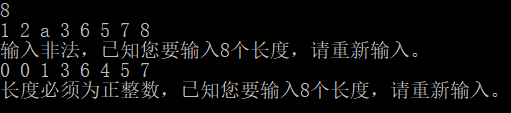


### 4.3.2 输入length非法或不在范围内

**测试用例：**输入length时非法或不在范围内

**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：**

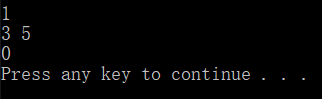


### 4.3.3 输入超过规定

**测试用例：**N=1，length输入3 5

**预期结果：**根据题目要求，只读取3，舍弃5，程序正常运行输出0

**实验结果：**



# 5 亮点

### 5.1 寻找最优解算法亮点

参考3.3.3

### 5.2 人机交互亮点

参考3.2.4

### 5.3 优先级序列创建时的注意事项及难点

参考2.2， 3.1.2