

数据结构课程设计

——修理牧场项目

项目说明文档

作 者 姓 名： 韦世贸

学 号： 2351131

指 导 老 师： 软件学院

专 业： 软件工程

同济大学

Tongji University

1 介绍

1.1 背景介绍

农夫要修理牧场的一段栅栏，他测量栅栏，得出需要的总木头数N，每块木头长度不同均为整数，于是他需要将一块很长的能锯成N块的木头交给有锯子的人锯开，每次收费为锯木头的长度，请你动动脑筋，将锯木头的总费用算出

1.2 功能介绍

输入格式：输入第一行，给出正整数N，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。

输出格式：需要通过计算得出锯出N块木头所需要的最小的花费。

2 设计

2.1 数据结构设计

本项目中的数据结构使用了优先级队列进行实现，使用动态分配内存的数组来实现优先级队列，数据结构比较简单。

2.2 类结构设计

本项目的实质就是对哈夫曼树的一种实现，求哈夫曼树中所有非叶子节点的权值之和。此题中运用了使用了一个优先级队列的模板类。

2.3 成员与操作设计

优先级队列类

template<class T>

class PQueue

公有操作

bool isFull()

bool isEmpty()

//插入元素

bool Insert(T x)

//删除最小的元素并取得它

bool RemoveMin(T& x)

//取得队列长度

int getLength()

私有成员和操作

T\* pqelements;

int count;

int maxsize;

//调整所有元素的排列顺序，使其从小到大排列

void adjust()

3 实现

3.1 调整排序功能的实现

3.1.1 调整排序功能流程简介

调整排序功能先将数组中元素不按顺序进行输入数组建立一个由数组表示的二叉树中，第一个元素为数组中下标为0的元素，再对数组进行调用向下调整函数进行排序，最终得到最小权值优先队列

3.1.2 调整排序功能核心代码

//调整所有元素的排列顺序，使其从小到大排列

void adjust()

{

T temp = pqelements[count - 1];

//从队尾开始排序

int j;

for (j = count - 2; j >= 0; j--) {

//小于等于temp跳出循环

if (pqelements[j] <= temp)

break;

//比temp大则往后移一位

else

pqelements[j + 1] = pqelements[j];

}

pqelements[j + 1] = temp; //temp插入适当位置

}

3.2 入队列功能的实现

3.2.1 入队列功能流程简介

入队列功能包括进入队列以及队列元素位置向上的调整，入队列默认从优先级队列队尾开始进入，然后调用调整排序函数进行调整。

//插入元素

bool Insert(T x)

{

//若优先级队列不满，则将元素插入队尾，再排序，否则出错

if (isFull())

return false;

pqelements[count] = x; //插入队尾

count++;

adjust(); //调整排序

return true;

}

3.3 出队列功能的实现

3.3.1 出队列功能流程简介

先判断队列是否为空，如果为空则进行错误处理，如果队列不为空，则通过引用取得队列头（即优先级队列中位于权值最大的元素），并且将最后一个数据放入队首再调用自上而下的调整进行调整。

//删除最小的元素并取得它

bool RemoveMin(T& x)

{

if (isEmpty()) //队列为空，无法删除元素

return false;

x = pqelements[0];

int min = 1;

//取得最小元素的位置

for (int i = 2; i < count; i++)

if (pqelements[i] < pqelements[min])

min = i;

//更改最小元素的位置

pqelements[0] = pqelements[min];

pqelements[min] = pqelements[count - 1];

count--; //优先级队列元素个数-1

return true;

}

3.4 总体功能的实现

3.4.1 总体功能流程介绍

总体输出一个最后所需花费的数目，进行调用函数以及逐个相加最终得到所需费用。求哈夫曼树中所有非叶子节点的权值之和的算法如下，只要优先级队列中元素数量大于1，就取出两个最小的元素x和y，计算合并花费并累加到total\_cost，然后将合并后的长度x+y重新插入优先级队列。

3.4.2 总体功能核心代码

//计算花费

void Calculate()

{

while (Cost.getLength() > 1) {

int x, y;

Cost.RemoveMin(x);

Cost.RemoveMin(y);

total\_cost += (x + y);

Cost.Insert(x + y);

}

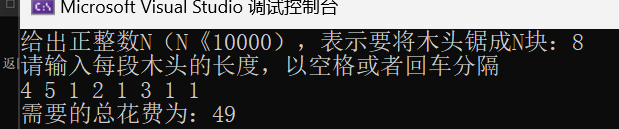
cout << "需要的总花费为：" << total\_cost << endl;

return;

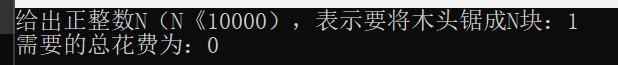
}

4.总体功能实现截屏

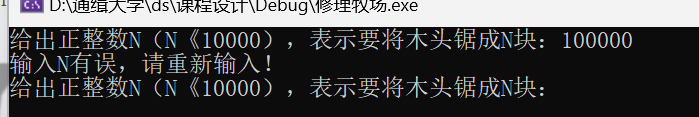
测试1：



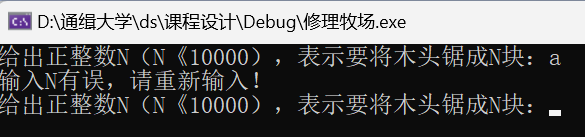
测试2：N=1



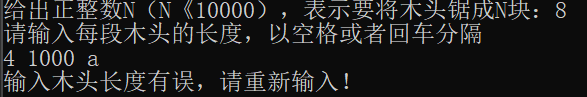
测试3：N超出范围



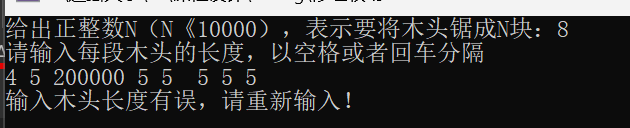
测试4：N类型不正确



测试5：x类型不正确



测试6：x范围不正确



5.总结

通过这个项目的设计，我熟悉了优先级队列的各种操作，也学会了使用动态分配内存的数组来实现优先级队列，还复习了有关哈夫曼树的知识。但之后我发现，可以使用最小堆来优化时间复杂度。在之后的学习中，如果涉及到最小值问题，我会多使用最小堆来减少时间复杂度。

在这个项目中，我还增加了输入输出的提示和错误处理，在总体功能实现中有展示。