****

**Tongji University**

**数据结构课程设计实验报告**

**007-修理牧场**

专 业： 软 件 工 程

指导教师： 张 颖

学 号： 2 1 5 3 0 6 1

姓 名： 谢 嘉 麒

1. **项目概览**
   1. **项目背景**

价格最优问题主要内容就是将分段价格最小化，而这一点自然将问题引向了最优二元树。

最优二元树则是在树的基础上考虑带权路径长度（Weight Path Length,WPL）的特殊处理。带有全职的叶结点叫做扩充二叉树的外结点，其余不带权值的分支结点叫做内结点。直观来看，带权路径长度最小的二叉树应是权值大的外结点离根结点最近的扩充二叉树，即Huffman树。

在本项目中，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。设酬金等于所锯木头的长度。

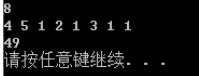
* 1. **项目要求**
     1. **项目输入要求**

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。

* + 1. **项目输出要求**

输出格式：输出一个整数，即将木头锯成 N 块的最小花费。

* + 1. **项目示例**

****

* 1. **需求分析**

修理牧场问题主要考虑项目的执行效率，时间和空间开销以及程序的正确性和严谨性问题。

* + 1. **系统功能的完善性**

修理牧场项目需要实现求价格最小化的功能。完善起见需要对用户的错误输入进行处理提示。

* + 1. **程序的正确性**

修理牧场涉及公民财产，因此程序应该以严谨无误为要求，保证数据存储的正确性，避免数据丢失等错误。

* + 1. **程序的健壮性**

程序需要在用户输入错误数据时及时提示，防止客户在求解最优价格时出现错误数据。

1. **项目设计与功能**
   1. **数据结构的设计与实用**

本次项目借鉴了Huffman树的求解方法，本质上是每次挑选出两条最小边（两段最小的木材）加入生成树。因此，本次项目设计了最小堆作为项目的数据结构，每次从最小堆中取出两个元素，将其和加入总价格，将其重新加入最小堆。

堆作为优先级队列中最高效的一种数据结构，有着很广泛的应用。其主要思想是将较小元素上浮至堆顶（即队列队首）

这一过程使用了建立Huffman树的思想。

1. 最小堆的数据元素：

在最小堆中设定三个元素，即存放堆的数组指针，堆元素个数和堆最大容纳量。

1. //存放堆中元素的数组

Type\* heap;

1. //最小堆中当前元素个数

int currentSize;

1. //最小堆最多允许元素个数

int maxHeapSize;

1. 最小堆的成员函数：

在最小堆中设定成员函数，完成堆的建立，堆元素上浮，堆元素插入删除置空判空等操作。

1. //从start到m下滑调整成为最小堆

void siftDown(int start, int m);

1. //从start到0上滑调整成为最小堆

void siftUp(int start);

1. //空堆构造函数

minHeap(int sz);

1. //利用数组建立堆

minHeap(Type arr[], int n);

1. //析构函数

~minHeap()

1. //将元素x插入最小堆中

bool insert(const Type& x);

1. //删除堆顶上的最小元素并返回其值

Type delMin();

1. //判断堆是否空

bool isEmpty()const

1. //判断堆是否满

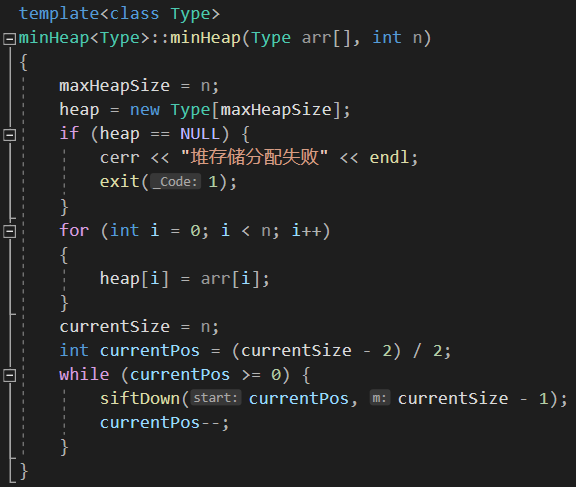
bool isFull()const

1. //置空堆

void makeEmpty()

1. int getCurrentSize()
   1. **算法的设计与函数功能的实现**
      1. **堆建立函数的实现**

建立堆首先需要给堆指针动态申请内存，若无特殊需要则按默认大小进行申请。随后通过循环反复调用堆元素下滑函数siftDown进行排序，使元素下滑变成最小堆。

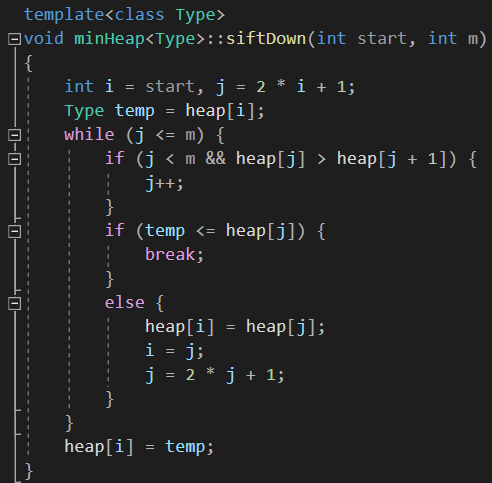


* + 1. **下滑/上浮调整最小堆函数的实现**

堆的下滑调整算法基本思想是：对有m个记录的集合R，将它置为完全二叉树的顺序存储。若结点i左子女关键码小于右子女关键码，则沿结点左分支调整，否则沿右分支调整，j指示参加调整的子女。

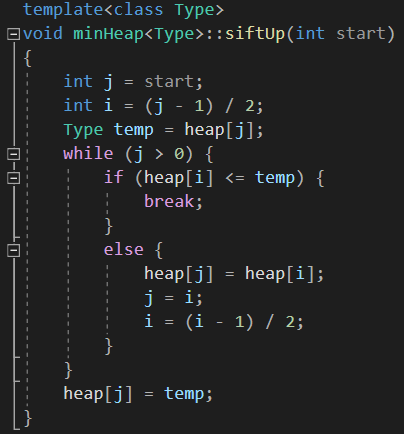
比较i和j结点的关键码，令较小者上浮，令i=j，j=2i+1，继续向下一层比较；若关键码i小于等于j，则不对调，算法终止。

具体代码实现如下：



堆上浮调整算法的基本思想和下滑调整算法类似，但其从父结点出发，逐渐与父母结点比较，使结点上浮，得到最小堆。

具体代码实现如下：



* + 1. **堆插入和删除算法**

堆的插入和删除算法只需要插入/删除数据，然后分别调用siftUp和siftDown函数重新将堆整理为最小堆即可。

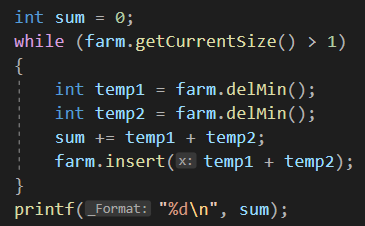
* + 1. **最小价格求解算法**

最小价格求解只需要设定总价变量，规定在最小堆内容不为0时进行循环。

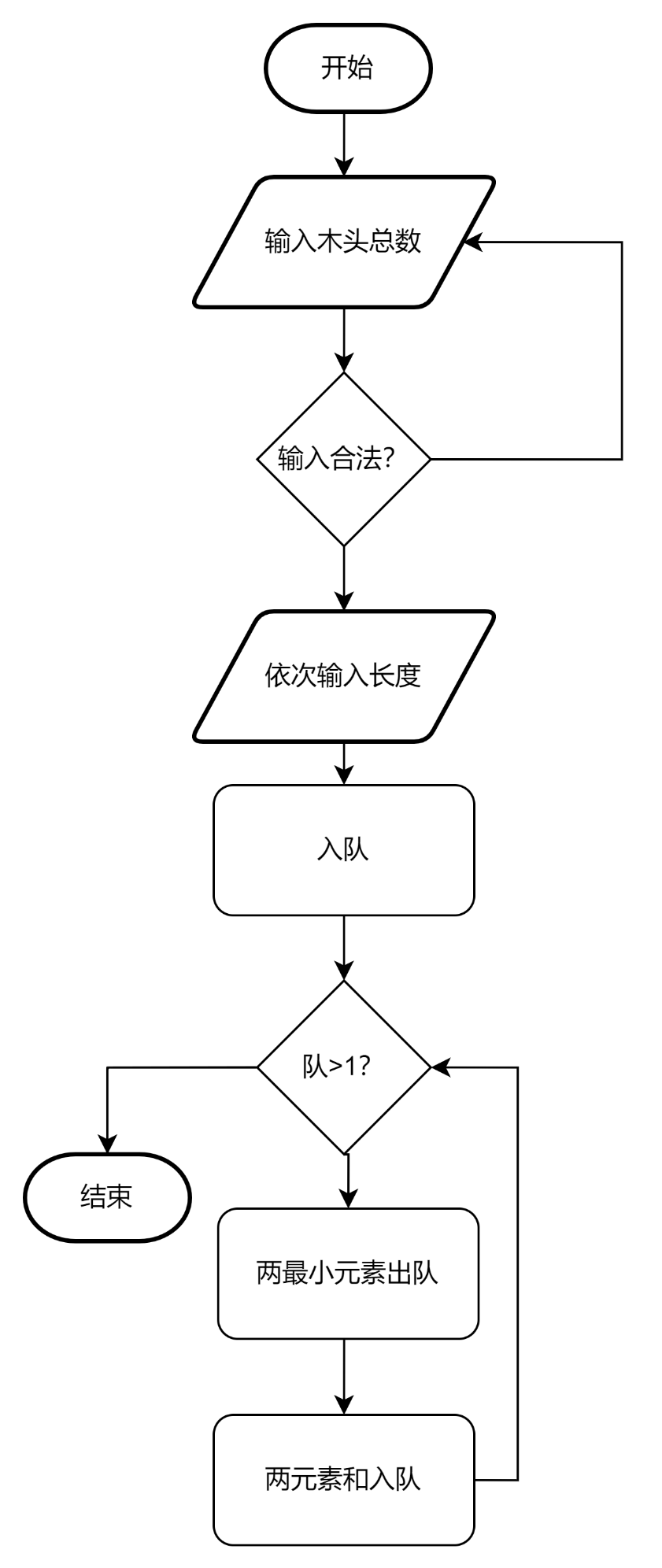
设定两个中间变量，分别取出最小堆的最小和第二小元素将其输入的中间变量中。将两个中间变量求和置入总价中，将其和同样加入最小堆。

循环完成后，打印总价即可。

此过程运用求解Huffman树的思想



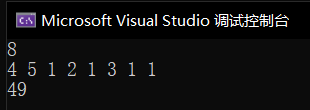
1. **程序流程图图解**

****

1. **程序测试**
   1. **样例测试**

测试数据:8 4 5 1 2 1 3 1 1

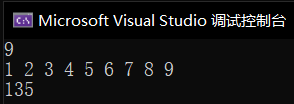
预期结果：49

****

* 1. **木头长度不同的情况**

测试数据:9 1 2 3 4 5 6 7 8 9

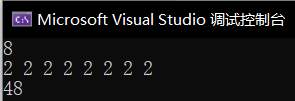
预期结果：135

****

* 1. **木头长度相同的情况**

测试数据:8 2 2 2 2 2 2 2 2

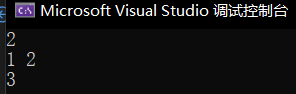
预期结果：48

****

* 1. **边界情况**
     1. **只有两个输入**

测试数据:2 1 2

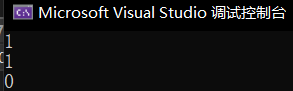
预期结果：3

****

* + 1. **只有一个输入**

测试数据:1 1

预期结果：0

****