修理牧场

作 者 姓 名： 王凌

学 号： 1951504

指 导 教 师： 张颖 \_\_

学院、 专业： 软件工程 \_\_

同济大学

Tongji University

目录

[1 分析 1](#_Toc10649)

[1.1 项目简介 1](#_Toc12723)

[1.2 解题分析 1](#_Toc32257)

[2 设计 2](#_Toc5945)

[2.1 小根堆设计 2](#_Toc3420)

[2.2 主程序设计 2](#_Toc28927)

[3 实现 3](#_Toc4515)

[3.1 小根堆的实现 3](#_Toc13313)

[3.1.1 小根堆MinHeap类的成员和成员函数 3](#_Toc19258)

[3.1.2 自定义构造函数 3](#_Toc2572)

[3.1.3 push(int)压堆函数的实现 4](#_Toc2499)

[3.1.4 pop()出堆函数的实现 4](#_Toc11320)

[3.1.5 top()函数实现 4](#_Toc381)

[3.2 主函数的实现 5](#_Toc10099)

[4 测试 5](#_Toc28973)

# 1 分析

## 1.1 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）.

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。

输出格式：输出一个整数，即将木头锯成N块的最小花费。

## 1.2 解题分析

给了要将木头锯成N块,与每块木头的长度。最后需要输出最小的花费。

如果从锯木头的思路来考虑，很难找出一种较为简单的策略来锯木头，那么我们只能搜索所有锯木头的方法来得到最小的花费。但是这种思路会耗费大量的空间与时间，我们不妨反向思维，从拼木头来考虑。为了保证总花费最小，我们每次都要让较短的两块木头拼起来，每一次的花费最小，那么最终的花费也必然最小，是一个成功的贪心策略。那么我们应该如何实现呢？根据课内的知识，很显然地，我们可以用哈夫曼树来做。但是实际上我们并没有必要完全建立一个完整的哈夫曼树，因为我们只需要记录每次拼木头的花费，与此同时，如果建立一个完整的哈夫曼树需要较大的空间和较大的代码量。我们只需要借用哈夫曼树的思想，然后用堆排序来做。为什么用堆排序做呢？借用哈夫曼树的思想，我们要做的是，每次要选取两个最小的结点值，然后加起来之后再进行排序。如果用快排等排序方法做，每一次排序的时间复杂度是O(nlogn)，而堆排序每一次压进堆然后维护堆的时间复杂度是O(logn)。又因为每一个结点要排一次序，所以用快排等其它排序做总时间复杂度O(n2logn)，而用堆排序做时间复杂度为O(nlogn)。因为N<=104，nlogn总运算次数少于108次，故一般能够在一秒内完成运算。

# 2 设计

## 2.1 小根堆设计

最小优先级队列每次出队列的是优先权最高的最小元素，即是小根堆。设计小根堆的思路为，用数列存储元素，每一次往小根堆加入元素时，都把元素放在数列末尾，然后让这个元素上升到比父节点元素小的位置，这样的话，堆顶永远是最小元素。每次弹出的都是堆顶的最小元素，记录堆顶元素，并将数组末尾的元素放置到堆顶，此时堆的当前长度(CurrentSize)减1，堆顶元素一直下沉到比子节点元素大的位置。总结的说，就是每次加入或弹出结点都要维护一次堆，使得堆的性质一直不变。

## 2.2 主程序设计

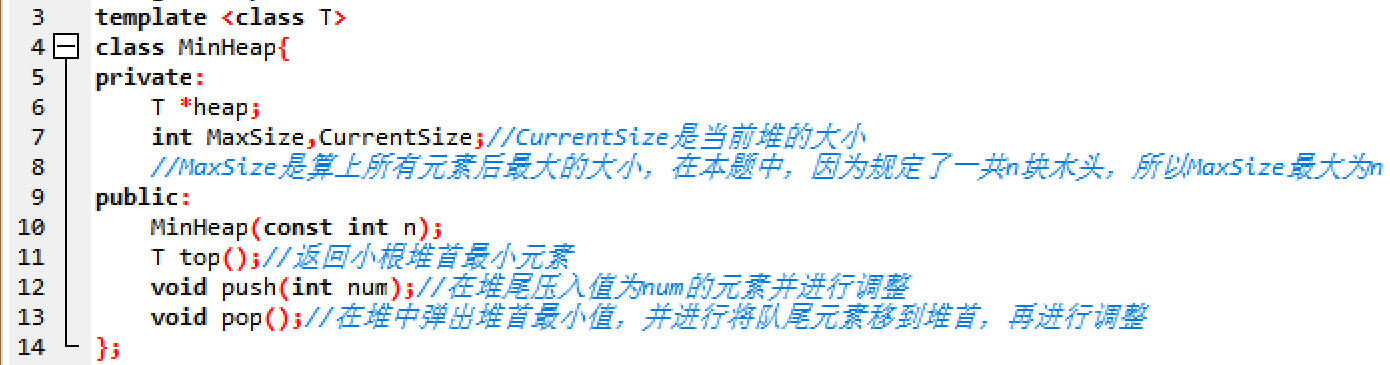
首先用循环将输入的N个正整数逐个压入堆。之后再次循环N-1次，每次都从堆中记录并弹出两个最小的元素，让总的花费ans加上这两个的值，并且把两个加起来的值继续压入堆中。

# 3 实现

## 3.1 小根堆的实现

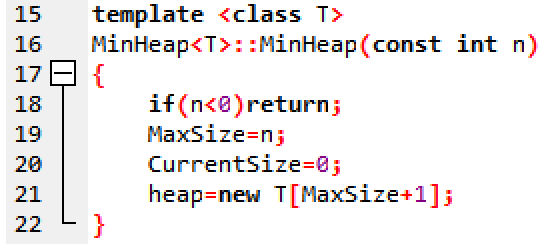
### 3.1.1 小根堆MinHeap类的成员和成员函数

### 私有类包括heap数组，MaxSize和CurrentSize分别为堆的数组最大可存长度和当前堆的大小。公共类包括四个成员函数，分别是自定义构造函数，返回堆顶元素的top()函数，压堆的push(int)函数和出堆的pop()函数。堆的向上与向下调整被整合在push(int)与pop()函数之中。



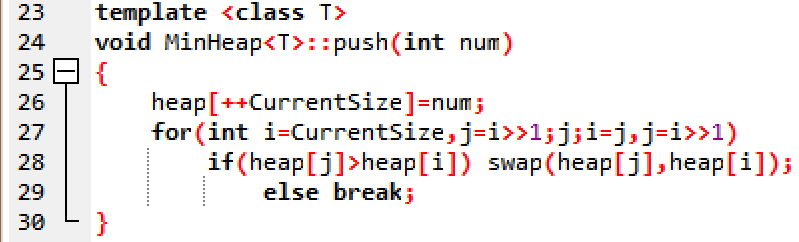
### 3.1.2 自定义构造函数

输入数组长度n，如果长度为负数则返回不做。如果长度输入正确，则MaxSize=n ，CurrentSize=0，堆的数组最大长度为n，当前长度为0。同时，开辟一个n+1长度的数组heap。



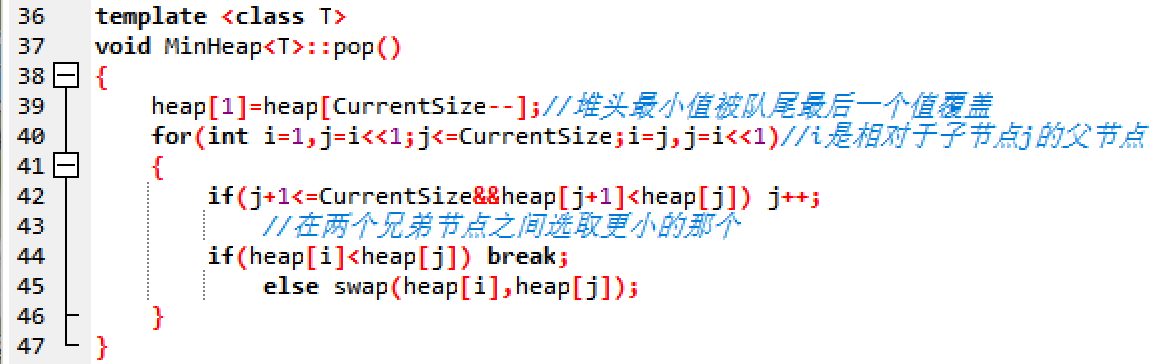
### 3.1.3 push(int)压堆函数的实现

把元素存放在数组末尾，i记录的是num的结点位置，j是i结点的父节点位置，不断比较，如果父节点比子节点的值大，就交换，一直交换到父节点比子节点值小为止。此时小根堆重新被维护完毕。



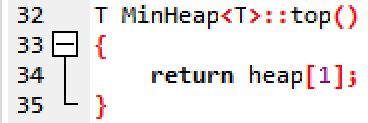
### 3.1.4 pop()出堆函数的实现

堆头最小值被队尾最后一个值覆盖，当前长度CurrentSize减一。i记录的是相对于子节点j的父节点位置，j为i父节点下两个子节点中较小的那个。如果父节点比两个子节点中最小的那个大则和它交换。一直交换到不需要交换为止。这样，小根堆又再次被维护完成。



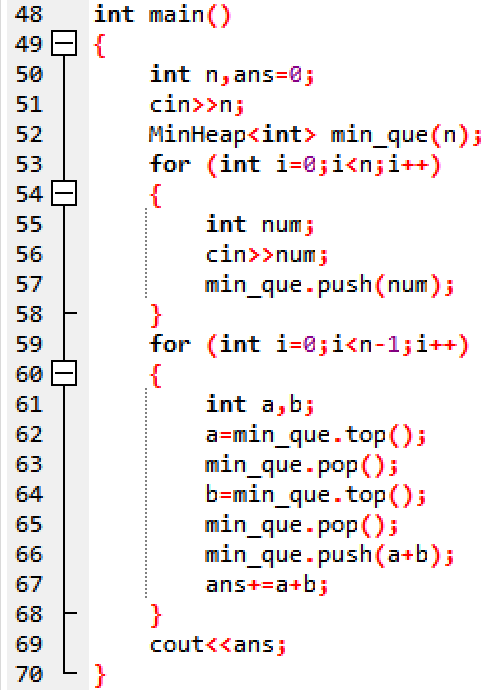
### 3.1.5 top()函数实现

堆顶元素即小根堆的最小元素。



## 3.2 主函数的实现

通过循环把每一个数字压入堆中并维护来构造小根堆。之后每次选取两个最小的结点，然后把他们加起来的和继续压入堆中，循环n-1次，此时堆还剩一个元素，即拼成了一块木头。



# 4 测试

