项目说明文档

数据结构课程设计

——修理牧场

作 者 姓 名: 沈星宇

学　　号: 1951576

指 导 教 师: 张颖

学院、 专业: 软件学院 软件工程

同济大学

Tongji University

一、 分析

（1） 应用背景

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，第一次将木头锯成 12 和 8，花费 20；第二次将长度为 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）。

（2） 项目功能要求

（1） 输入格式：输入第一行给出正整数 N（N < 10^4），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。

二、 设计

（1） 数据结构设计

```
class MinHeap{
private:
    int* heap;
    int MaxHeapSize;
    int CurrentSize;
    void FilterDown(int start, int endOfHeap);
    void FilterUp(int i);
public:
    MinHeap() { ... }
    MinHeap(int arr[],int size) { ... }
    ~MinHeap() { ... }
    bool Insert(const int& x);
    bool RemoveMin(int& x);
    int isEmpty()const { ... }
    int isFull()const { ... }
    void MakeEmpty() { ... }
};
```

1、 MinHeap 是最小堆，本项目通过构建哈夫曼树的思想，用最小堆来实现。

2、 函数 `void FilterDown(int start, int endOfHeap);` 是用于自上而下将其调整为最小堆；`void FilterUp(int i);` 是将插入新节点后的结构从下往上调整为最小堆。

3、 函数 `bool Insert(const int& x);`是往树里插入新的节点，而函数 `bool RemoveMin(int& x);`是将根节点的最小值取出。

（2） 程序流程设计

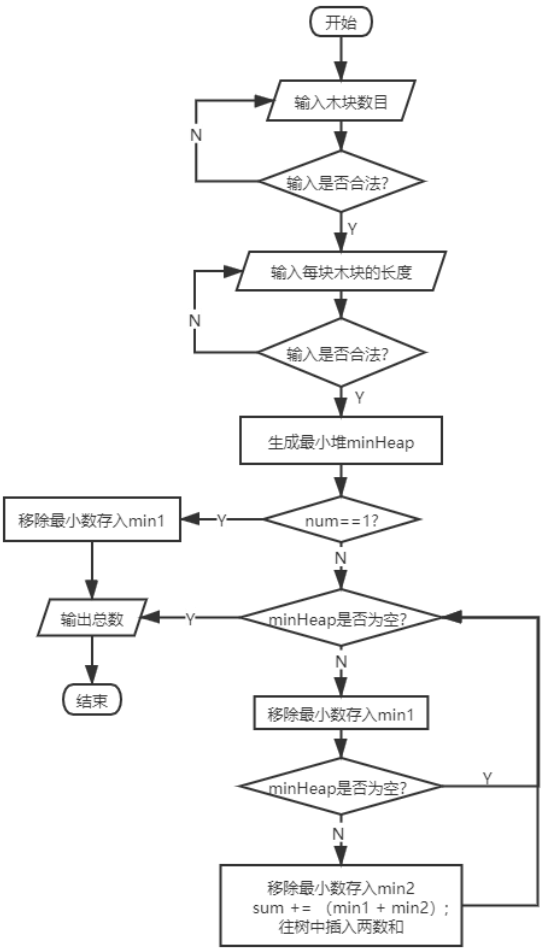
1、 首先木块的数目，并进行合法性检测

2、 然后输入每块木块的长度，并进行合法性检测

3、 生成最小堆 minHeap。

4、循环调用 MinHeap::RemoveMin()函数拿出最小的数字存入 min1，若最小堆非空，则再拿出新生成的最小堆的最小数存入 min2。将两数相加存入 sum，并将两数和重新放回到堆当中。

5、输出最后的结果 sum。



三、 实现

1、 main()的部分实现

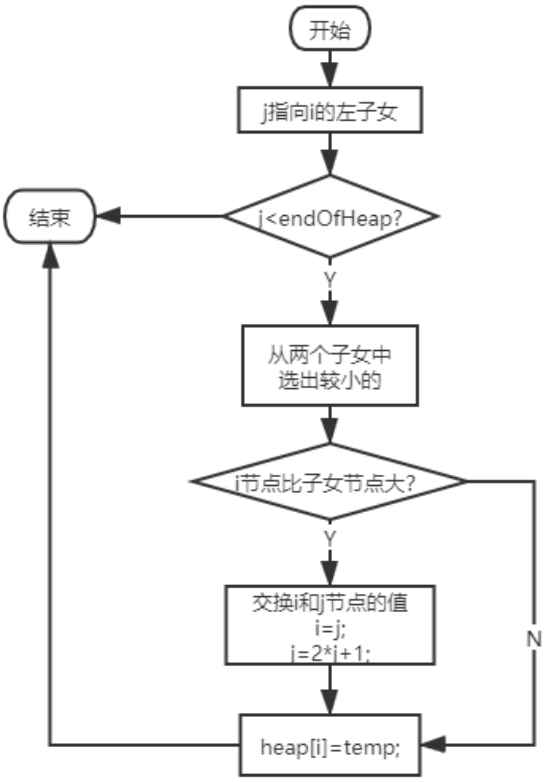
```
MinHeap minHeap(arr, num);
int sum = 0;
int tempNum;
int min1, min2;
if(num != 1) {
    while (!minHeap.isEmpty()) {
        minHeap.RemoveMin(min1);
        if (!minHeap.isEmpty()) {
            minHeap.RemoveMin(min2);
            tempNum = min1 + min2;
            sum += tempNum;
            minHeap.Insert(tempNum);
        }
    }
}
std::cout << "最小开销为:" << sum << std::endl;
free(arr);
```

(1) 主要利用哈夫曼树的思想，即在构造哈夫曼树的过程中，是将整个序列中最小的两个数字拿出，然后将两数和重新添加到序列当中，那么构造出的哈夫曼树遍历就能得到最小的 WPL。此处是用了最小堆来找到整个序列当中最小的数字。

(2) 始终要保证再最小堆非空时从里面取数字，循环调用 MinHeap::RemoveMin() 函数拿出最小的数字存入 min1，若最小堆非空，则再拿出新生成的最小堆的最小数存入 min2。将两数相加存入 sum，并将两数和重新放回到堆当中。

2、 MinHeap::FilterDown()的实现

```
void MinHeap::FilterDown(int start, int endOfHeap) {
    int i = start, j = 2 * i + 1;
    int temp = heap[i];
    while (j <= endOfHeap) {
        //两子女中选小的
        if (j<endOfHeap && heap[j]>heap[j + 1]) {
            j++;
        }
        if (temp <= heap[j]) {
            break;
        }
        else {
            heap[i] = heap[j];
            i = j;
            j = 2 * j + 1;
        }
        heap[i] = temp;
    }
}
```



(1) 对于一个 i 来说，j=2*i+1 就是 i 这个节点的左子女（由堆的性质可得）
(2) 当 j 到达堆的最后一个节点时结束循环
(3) 从 i 的左右子女（如果由右子女的话）当中选出最小的那个，与 i 节点的值进行比较，如果 i 节点已经是最小的了，则结束循环；如果 i 的子女节点比 i 节点更小，那么交换两个节点的值，同时继续向下搜索（即 i 指向 j，j 指向下一个左子女）

3、 MinHeap::FilterUp()的实现

(1) 本函数是针对新加入的节点，从下往上调整它的位置
(2) j 指向当前节点，i 是 j 的父母节点，循环判断新的节点的值与其父母节点的大小关系，如果新的节点比其父母节点更小，则往上调整其位置；其他的情况则可以退出循环。
(3) 基本上就是 MinHeap::FilterDown()的逆过程

```
void MinHeap::FilterUp(int start) {
    int j = start, i = (j - 1) / 2;
    int temp = heap[j];
    while (j > 0) {
        if (heap[i] <= temp) {
            break;
        }
        else {
            heap[j] = heap[i];
            j = i;
            i = (i - 1) / 2;
        }
        heap[j] = temp;
    }
}
```

4、 MinHeap::Insert()的实现

（1） 将插入的值放在数组的最末端，然后调用 FilterUp()函数将其调整为最小堆，同时调整最小堆的大小。

```
bool MinHeap::Insert(const int& x) {
    if (CurrentSize == MaxHeapSize) {
        std::cout << "堆已满" << std::endl;
        return false;
    }
    heap[CurrentSize] = x;
    FilterUp(CurrentSize);
    CurrentSize++;
    return true;
}
```

5、 MinHeap::RemoveMin()的实现

(1) 移除根节点最小的那个数字，通过引用 x 来将其值传出

(2) 同时对于 heap[]数组来说，要做的就是把最前面的数字给放到最后，同时让 CurrentSize--,确保该值不会再被访问。

(3) 调用 FilterDown()函数，调整其重新成为最小堆。

```
bool MinHeap::RemoveMin(int& x) {
    if (!CurrentSize) {
        std::cout << "堆已空" << std::endl;
        return false;
    }
    x = heap[0];
    heap[0] = heap[CurrentSize - 1];
    CurrentSize--;
    FilterDown(0, CurrentSize - 1);
    return true;
}
```

四、 测试

1、合法性检测

（1）输入的木块数目不合法（即非正整数）,会要求重新输入。

（2）输入的木块长度不合法（即非正整数）,会要求重新输入。

```
D:\VS文件\数据结构课程设计\Project7\Release\Project7.exe
请输入要将木头锯成几块：a
输入不合法，请输入正整数
请输入要将木头锯成几块：0
输入不合法，请输入正整数
请输入要将木头锯成几块：2
请输入每一段木块的长度：
1 a
输入不合法，请重新输入正整数序列
请输入每一段木块的长度：
1 -1
输入不合法，请重新输入正整数序列
```

2、一般情况的功能实现

（1）一般的情况会计算出最小开销

（2）特殊情况要将木头锯成 1 块（相当于没有锯）那么开销为 0

```
D:\VS文件\数据结构课程设计\Project7\Release\Project7.exe
请输入要将木头锯成几块：8
请输入每一段木块的长度：
4 5 1 2 1 3 1 1
最小开销为:49
```

```
D:\VS文件\数据结构课程设计\Project7\Release\Project7.exe
--------------------------------
请输入要将木头锯成几块：1
请输入每一段木块的长度：
10
最小开销为:0
```