同濟大學
TONGJI UNIVERSITY

项目说明文档

修理牧场

指导教师：张颖

1751984 王舸飞

1.分析

1.1 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，请人锯木的酬金等于该木头的长度，现在要求给出一种方案，计算出锯木所用的最小花费。

1.2 功能分析

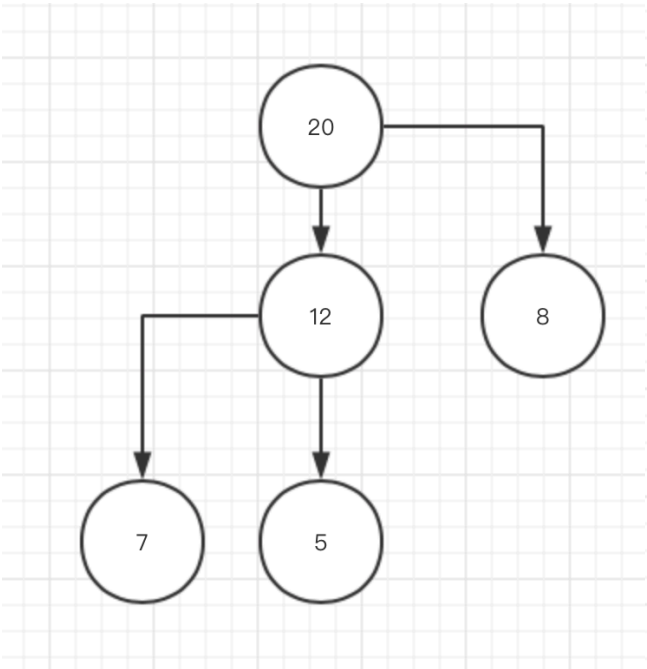
本次设计的输入分为两行，第一行会给出所需要锯成木头的节数，之后一行给出的就是各段木头的长度，为完成最基本的题目需求，需要输出他们所需的最小花费。

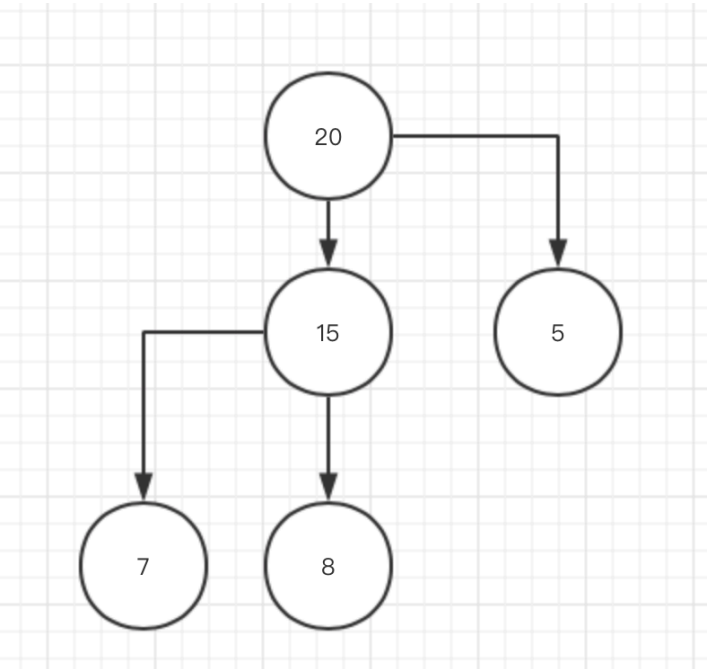
此外为了方便调试和观察，本次输出还包含了每次锯木的结果。

2.设计

2.1 算法设计

首先可以考虑题目要求给出的较为基础的锯木情况，对于案例中的要将长度为20的木头锯成8、7、5三段的情况，可以列出这两种锯法：





其中左侧的锯法花费为32，右侧为35，在计算时我们不难发现，最下层的两个节点被加了两次，上一层的节点被加了一次，这个结论可以推广到多层时的情况，即一个数被加的次数正好是它的层数。

因此，这种体系结构恰满足带权路径长度(Weighted Path Length,WPL)的所有要求，而求解最短带权路径长度可以使用构造Huffman树的方法。这种方法的核心思想是：将降尽量小的数放到尽量深的层从而实现路径和最短。

2.2 数据结构设计

本次Huffman树的所要完成的功能仅仅为输出一个最小的值而不需要对结果树进行任何操作，故本次设计数据结构时仅仅运用了Huffman最短路径的算法原理而并为组织起一个存储用的树：即开辟一个数组，每次选出最小两数合并产生新数组后循环操作。这种仅仅用一个数组的组织方法会遗漏结果信息，但与之相对的，在能完成要求的前提下结构简单，编译时间短，代码可读性较高。

2.3 成员与操作设计

整型数组：

Logs[10001]; //代表最终要锯成的结果，最多10000段

整型变量：

first; // 代表一次遍历中最短的木头的长度

firstid; //代表一次遍历中最短木头长度的序列码

second;//代表一次遍历中第二短的木头的长度

secondid; //代表一次遍历中第二短的木头的序列码

N; //代表当前剩余木头的数量，每次合并后减一

i; //循环变量

sum;//用于存储每次合并时两段木头的长度之和用于输出

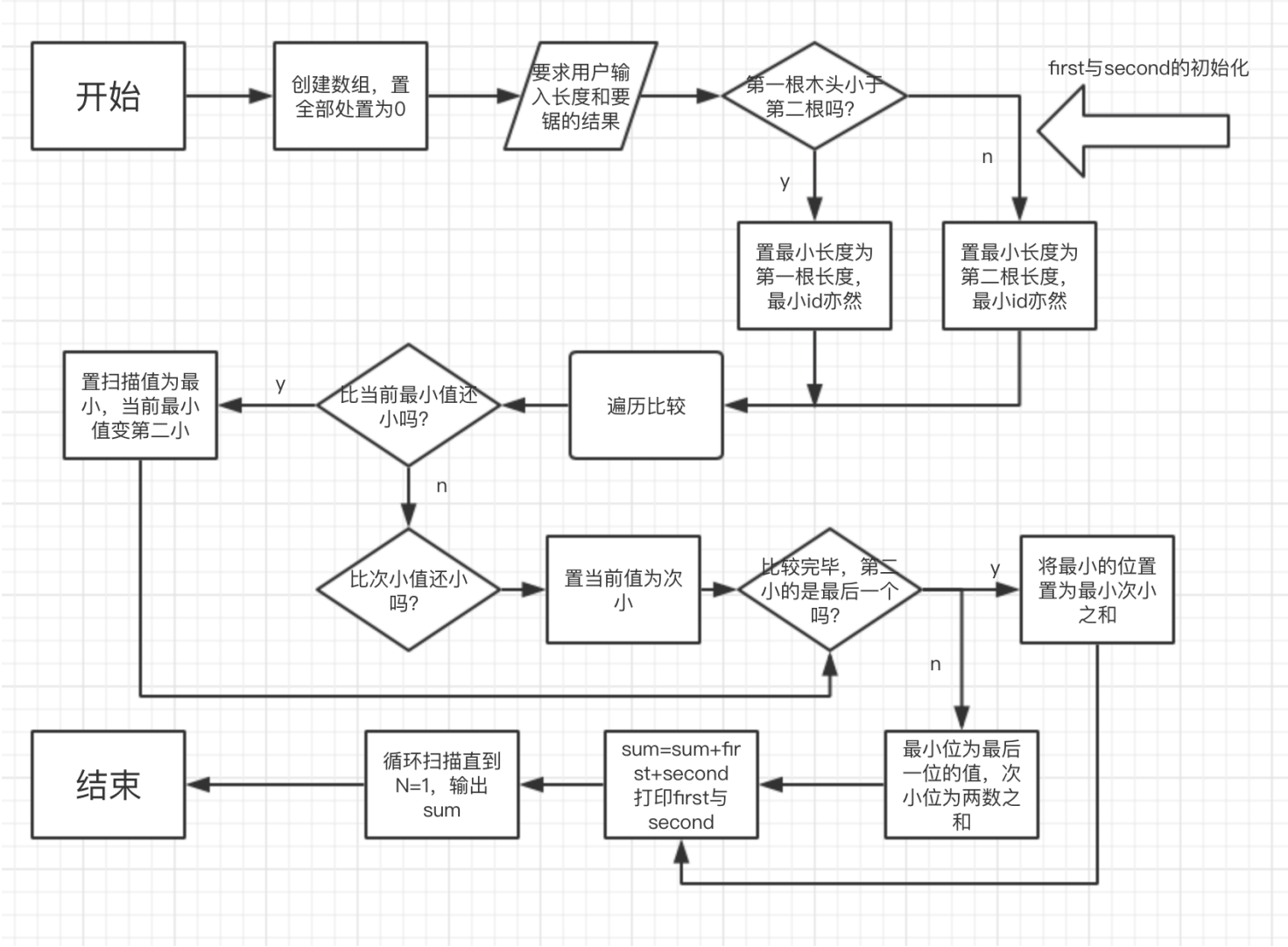
2.4 系统设计

程序运行之后，系统会创建一个用于存储木头长度的数组并要求用户依次输入木头长度，之后会对存储长度的数组进行遍历找出两最小值后合并放入数组，直到最后剩下一根木头后进行输出。

3.实现

3.1 总体功能的实现

3.1.1 总体功能流程图



3.1.2 总体功能核心代码

```
cin>>N;getchar();

for(int i=0;i<N;i++)cin>>logs[i];

while(N!=1){first=logs[0]<logs[1]?logs[0]:logs[1];
    firstid=logs[0]<logs[1]?0:1;
    second=logs[0]<logs[1]?logs[1]:logs[0];
    secondid=logs[0]<logs[1]?1:0;

    for(i=2;i<N;i++)
    {
        if(logs[i]<first){second=first;secondid=firstid;first=logs[i];firstid=i;}
        else if(logs[i]<second){second=logs[i];secondid=i;}

    }


    if(secondid==N-1){logs[firstid]=first+second;}
    else{logs[firstid]=logs[N-1];
        logs[secondid]=first+second;}
    cout<<first<<" "<<second<<endl;
    sum=first+second+sum;
    N--;}



cout<<sum<<endl;
```

4.测试

4.1 功能测试

4.1.1 一般情况测试1

测试用例：3
8 7 5

实验结果：

```
3
8 7 5
5 7
8 12
32
Program ended with exit code: 0
```

4.1.2 一般情况测试2

测试用例：4
2 4 5 7

本例为书p241页用例

实验结果：

```
4
2 4 5 7
2 4
5 6
7 11
35
Program ended with exit code: 0
```

4.1.3 一般情况测试3

测试用例：8
4 5 1 2 1 3 1 1

本例为要求测试样例

实验结果：

```
8
4 5 1 2 1 3 1 1
1 1
1 1
2 2
2 3
4 4
5 5
8 10
49
Program ended with exit code: 0
```

4.1.4 两位数测试

测试用例：10
8 5 6 4 3 22 5 4 1 3 1

实验结果：

```
10
8 5 6 4 3 22 5 4 1 3 1
1 3
3 4
4 4
5 5
6 7
8 8
10 13
16 22
23 38
180
Program ended with exit code: 0
```