项目说明文档

数据结构课程设计

——修理牧场

作 者 姓 名： 何慧琳

学 号： 2152343

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 分析

## 1.1 背景分析

农夫要请人将一块很长的木头锯成不同长度的几段，每锯一次的花费与锯完后锯成的两段木头的长度和成正比，即使锯成相同一组长度，不同的切割顺序也会产生不同的开销，当切割段数很多时人工列举计算出最小开销显得有些困难，用计算机程序来计算不仅快速而且准确，于是该项目具有十分重要的意义。

## 1.2 功能分析

该项目首先能输入要将木头锯成的块数和每块木头的长度，然后就是计算出将木头锯成N块的最小花费并输出。

# 2 设计

## 2.1 数据结构设计

经分析可以发现如果将每块被锯成后的木头作为树的结点，每锯一次后原来的一块木头（父节点）变成两块木头（左右子女）,最终每块被锯出来的木头都是一个叶结点，它在第几层代表第几次的时候被锯出来，它的长度就是它的开销也是结点的权值。于是总开销就是树的带权路径长度，计算最小开销的过程就是建立霍夫曼树的过程。

根据以上分析，为了建立霍夫曼树，需要定义一个扩充二叉树类（ExtBinTree）用于两棵树的合并。建立霍夫曼树要不断选择根权值最小的树，于是要用到最小堆（MinHeap）。

## 2.2 类结构设计

**最小堆类（MinHeap）**

最小堆是基于完全二叉树的数组表示定义的，因此它包含一个参数模板类型的一维空间，存各根节点。同时它应能实现插入结点、删除堆顶（最小结点）等功能，具体成员设计见下

**私有成员及操作：**

Type\* heap;

int CurrentSize;//堆当前大小

int MaxHeapSize;//堆最大大小

void FilterDown(int i, int m);//向下调整

void FilterUp(int i); //向上调整

**公有操作：**

MinHeap(int maxsize=10);

MinHeap(Type arr[], int n=10);

~MinHeap() { delete[] heap; }

int insert(const Type& x);//插入结点

int DeleteMin(Type& x);//在x堆中删除最小结点

int IsEmpty() const { return CurrentSize == 0; }//判断堆是否为空

int IsFull() const{return CurrentSize == MaxHeapSize;}//判断堆是否满

void MakeEmpty() { CurrentSize = 0; }//置空堆

**扩充二叉树类（ExtBinTree）**

扩充二叉树是建立霍夫曼树的辅助数据结构，其主要功能是将一棵树合并到另一棵树上，它的下层结构由另一个类（Element）来表示，既然是霍夫曼树的辅助结构在Element类中定义了私有成员key表示一棵树根节点的权值，还定义了左孩子右孩子指针以为了在ExtBinTree类中完成合并树的操作。

**私有成员：**

Element\* root=new Element();//扩充二叉树的根

**公有操作：**

ExtBinTree(ExtBinTree& bt1, ExtBinTree& bt2)

{//将bt2与bt1合并

root->leftchild = bt1.root;

root->rightchild = bt2.root;

root->key = bt1.root->key + bt2.root->key;

}

ExtBinTree() { root = new Element(); }

ExtBinTree(ExtBinTree\* bt)

{

root = bt->root;

## }

## 2.4 系统设计

系统首先调用main()函数完成对木头块数和每块木头长度的输入数据工作，创建按一个扩充二叉树对象用于存储目标二叉树，然后调用霍夫曼树生成函数在完成霍夫曼树的建立后将输出最小开销。

# 3 实现

## 求最小开销功能的实现

### 核心代码

void HuffmanTree(Type\* fr, int n, ExtBinTree& newtree)

{//建立霍夫曼树

ExtBinTree first, second;

ExtBinTree \*Node=new ExtBinTree[n];

int ans=0;//最小权重和

for (int i = 0; i < n; i++) {

Node[i].root->key = fr[i];

Node[i].root->leftchild =Node[i].root->rightchild = NULL;

} //传送初始权值

MinHeap < ExtBinTree> hp(Node, n); //最小堆

for (int i = 0; i < n - 1; i++) {//建立霍夫曼树的过程，做n-1趟

hp.DeleteMin(first); //选根权值最小的树

hp.DeleteMin(second); //选根权值次小的树

newtree = new ExtBinTree(first, second); //建新的根结点

ans += newtree.root->key;

hp.insert(newtree); //形成新树插入

}

cout << ans << endl;

}

# 4 测试

## 4.1 功能测试

**测试用例**：

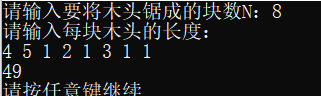
8

4 5 1 2 1 3 1 1

**预期结果**：

49

**实验结果**



## 4.2 边界测试

### 4.2.1 输入最小块数

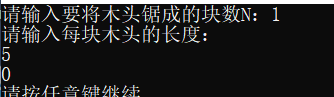
**测试用例：**

1

5

**预期结果：**0（不用锯，所以开销是0）

**实验结果：**



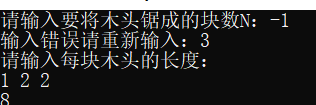
## 4.3 出错测试

### 4.3.1 输入木头块数错误

**测试用例：**输入木头块数为负数

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

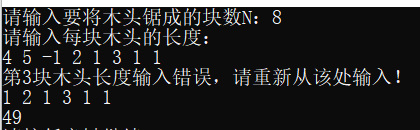
****

### 4.3.2 木头长度输入错误

**测试用例：**木头长度输入负数

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

****