**离散数学课程设计**

**项目说明文档**

**最优2元树在通信编码中的应用**

作 者 姓 名： 苏家铭

学 号： 2151299

指 导 教 师： 唐剑锋

学院、 专业： 软件学院 软件工程

同济大学

**Tongji University**



**目录**

[1 项目分析 1](#_Toc2414)

[1.1 项目背景 1](#_Toc3717)

[1.2项目功能要求 1](#_Toc11625)

[1.2 项目实例 1](#_Toc18163)

[2 项目设计及实现 1](#_Toc21921)

[2.1 数据结构设计思路 1](#_Toc7652)

[2.2 类设计 2](#_Toc5543)

[2.2.1 堆 2](#_Toc29807)

[2.2.1 哈夫曼树 3](#_Toc7889)

[2.3 项目算法 4](#_Toc16552)

[2.3.1 实现思路 4](#_Toc22683)

[2.3.2 代码实现 4](#_Toc14092)

[3 项目测试 5](#_Toc3278)

[4 算法性能分析 5](#_Toc19571)

[4.1 正确性 5](#_Toc20021)

[4.2 可使用性 6](#_Toc15499)

[4.3 可读性 6](#_Toc13737)

[4.4 健壮性 6](#_Toc24865)

[5 实验感想 6](#_Toc12466)

# 1 项目分析

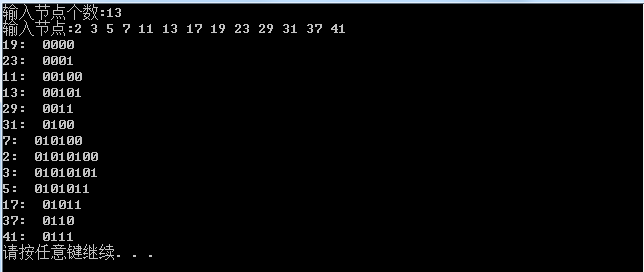
## 项目背景

哈夫曼树(Huffman Tree)是由n个带权叶子结点构成的所有二叉树中带权路径长度最短的二叉树，又称最优二叉树/最优二元树。哈夫曼编码是哈夫曼树的一个应用。在通信工程中，通常需要将传送的文字转换成由二进制字符组成的二进制串，这一过程被称为编码。不同字符出现的概率不一样，在传送电报时，总是希望电报尽可能短，那么采用哈夫曼编码构造每个字符的最佳2元前缀码写成的电报的总长将会是最短的。

## 1.2项目功能要求

输入一组通信符号的使用频率，求各通信符号对应的前缀码。

### 项目实例



# 2 **项目设计及实现**

## 2.1 **数据结构设计思路**

根据项目要求，需要在得到哈夫曼编码，那么自然联想到使用哈夫曼树这一数据结构来实现。而建树的过程需要每次选出一系列边中最小的两条边，如果仅仅使用一半的方法遍历选最小边，时间复杂度会达到O(n)，本项目为了优化这一过程，使用了堆来优化，使得时间复杂度降为O(log2n)。

## 2.2 **类设计**

### 2.2.1 堆

堆(heap):若有一个关键码的集合K = {k0，k1，k2，…，kn-1}，把它的所有元素按[完全二叉树](https://so.csdn.net/so/search?q=%E5%AE%8C%E5%85%A8%E4%BA%8C%E5%8F%89%E6%A0%91&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/zgdlxs/article/details/_blank)的顺序存储方式存储在一个一维数组中，并满足：Ki <= K2i+1 且 Ki <= K2i+2，则称为小堆(或大堆)。将根节点最大的堆叫做最大堆或大根堆，根节点最小的堆叫做最小堆或小根堆。

本项目中堆基于一维数值来实现，通过定义建堆、调整堆来完成了一个堆。

最小堆类数据成员：

template <class T>

T\* pheap;//存放小根堆中元素的数组

int current\_size;//小根堆中当前元素个数

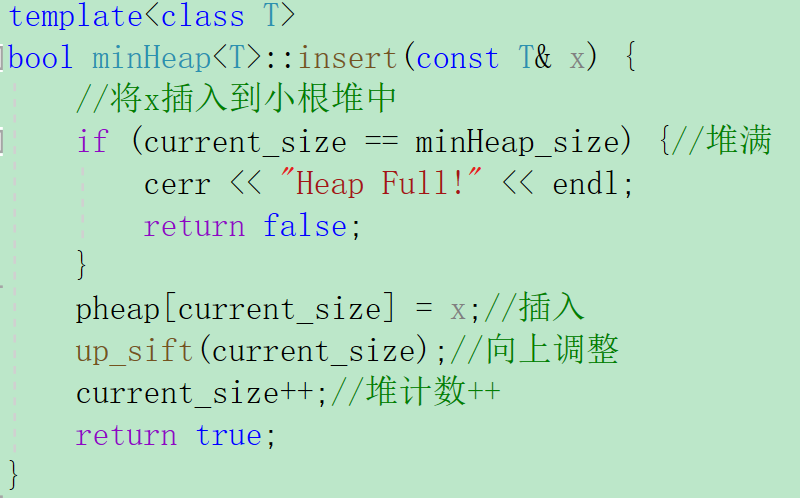
int minHeap\_size;//小根堆最多允许元素个数

该最小堆已经实现8种功能：向上调整、向下调整、插入元素、删除堆顶元素、判空、判满、置空堆、堆排序。

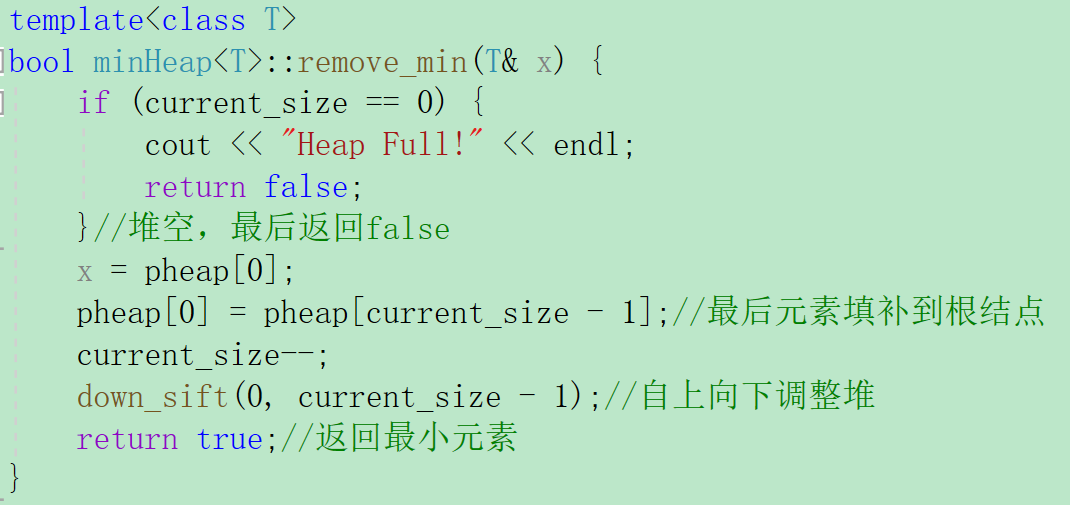
具体函数声明为：

* void down\_sift(int start, int m);//自上向下调整
* void up\_sift(int start);//自下向上调整
* void makeEmpty();//置空
* bool insert(const T& x);//将x插入到小根堆中
* bool remove\_min(T& x);//删除小根堆上的最小元素
* bool empty()const ;//判断堆是否为空
* bool full()const ;//判断堆是否达到最大限制个数
* void make\_empty();//置空堆
* void rank\_heap(T rank[]);//堆排序

插入算法代码实现：



删除算法代码实现：



主要算法的时间复杂度：

* 插入：O(log2n)
* 删除：O(log2n)
* 置空：O(n)
* 判断空否：O(1)

在最后为了能在优先级队列中访问堆的数据成员和成员函数，将优先级队列声明为友元函数：

//友元类

template <class E>

friend class minPQueue;

### 2.2.1 哈夫曼树

哈夫曼树的类设计利用结构体作为树的结点，其中包括数据域和两个指针域在树类内用root指针指向第一个结点的地址，实现树形结构。

在构造哈夫曼树的算法中使用了一个最小堆，利用它组织森林并从中选择根结点权值最小和和次小的两棵树。

哈夫曼树树结点数据成员：

T data;//每个结点的数据，可能是结构体

huffmanNode<T>\* left, \* right;//每个结点的左右指针

哈夫曼树类数据成员：

huffmanNode<T>\* root;//树的根结点指针

该哈夫曼树实现了2种基本功能：合并两棵子树，输出哈夫曼编码。

具体函数声明为：

* void merge(huffmanNode<T>\* l,huffmanNode<T>\* r, huffmanNode<T>\*& father);//合并两棵二叉树
* void preorder(huffmanNode<T>\* p,int h);//深度优先搜索输出编码主要算法的时间复杂度：
* 合并：O(1)
* 上输出：O(n)

## 2.3 **项目算法**

### 2.3.1 实现思路

构造哈夫曼树的算法步骤：

① 初始化：用给定的 n 个权值{w1,w2,…,wn}构造 n 棵二叉树并构成的森林F={T1,T2,…,Tn}，其中每一棵二叉树Ti（1<=i<=n）都只有一个权值为 wi 的根结点，其左、右子树为空。

② 找最小树：在森林 F 中选择两棵根结点权值最小的二叉树，作为一棵新二叉树的左、右子树，标记新二叉树的根结点权值为其左、右子树的根结点权值之和。

③ 删除与加入：从 F 中删除被选中的那两棵二叉树，同时把新构成的二叉树加入到森林 F 中。

④ 判断：重复②、③操作，直到森林中只含有一棵二叉树为止，此时得到的这棵二叉树就是哈夫曼树。

简单的说就是先选择权小的，所有权小的结点被放置在树的较深层，而权较大的离根较近，这样一来所构成的哈夫曼树就具有最小带权路径长度。

### 2.3.2 代**码实**现

#include<iostream>

#include"huffman\_tree.h"

using namespace std;

#include"min\_heap.h"

int main(){

int n, \*arr;

cout << "输入结点个数：";

while (1) {//健壮性

cin >> n;

if (cin.fail() || n < 1) {

cout << "输入非法，请重新输入!" << endl;

cin.clear();

cin.ignore(65535, '\n');

}

break;

}

arr = new int[n];

cout << "输入结点：" << endl;

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

cin >> arr[i];

}

if (n == 1) {

cout << arr[0] << "：" << "0";

delete[]arr;

return 0;

}

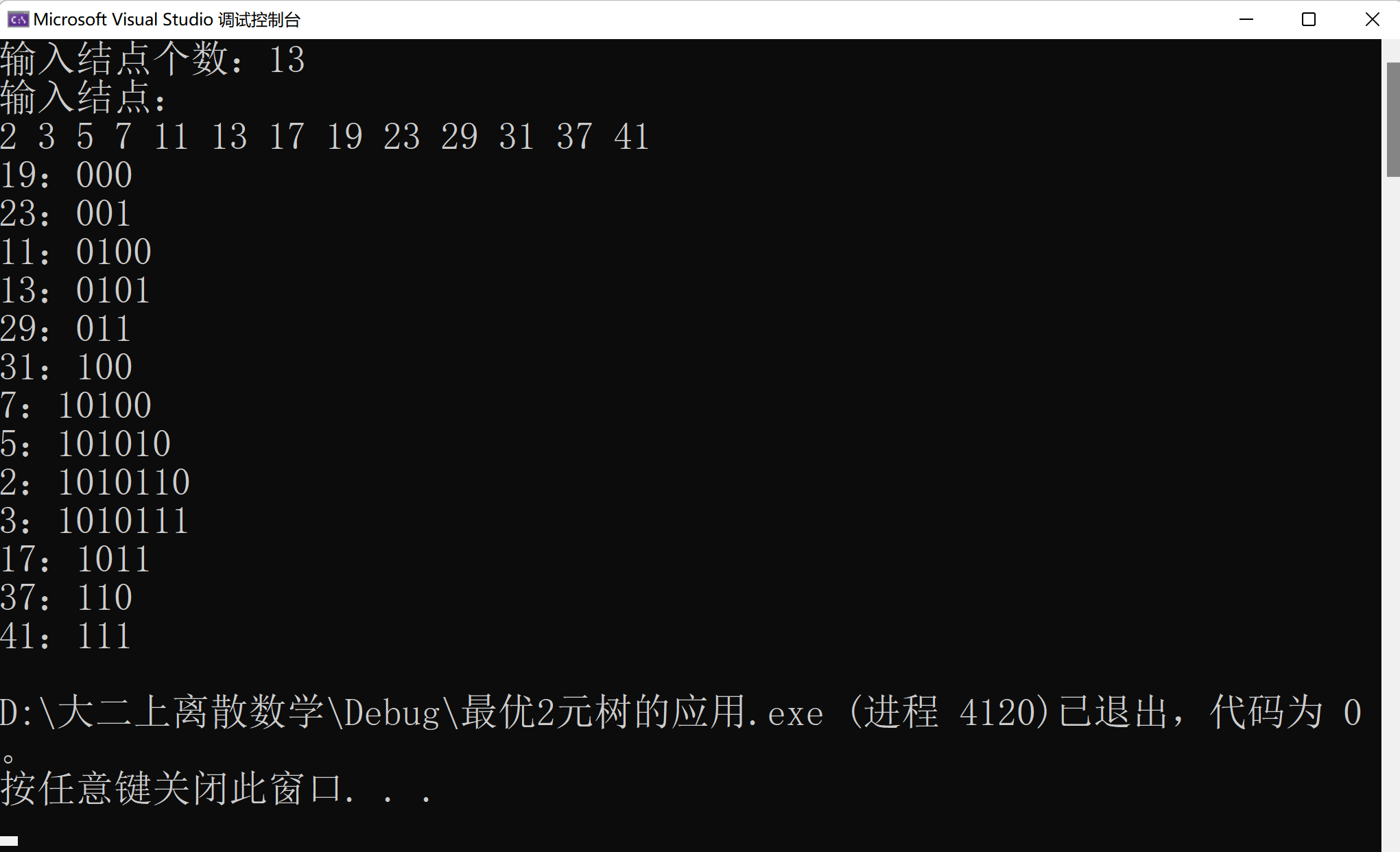
huffmanTree<int> tree(arr,n);//生成哈夫曼树并输出

delete[]arr;

return 0;

}

# 3 项目测试



# 4 算法性能分析

## 4.1 正确性

本算法能正确地执行预定的功能和性能要求，由以上版块证明本项目与例题的情况比较，并和大量数据验证后满足正确性。

## 4.2 可使用性

本算法可以很方便地使用，生成哈夫曼树和深度优先输出哈夫曼编码两个函数嵌套使用，只需要一个权值数组即可得出结果。并且该算法有良好的界面和完备的用户文档。

## 4.3 可读性

本算法逻辑清晰、简单、且结构化，所有命名与函数名都具有实际含义，让人见名知义。且算法中包含了大量注释，简要说明了算法功能、输入与输出参数的使用规则、重要数据的作用、算法中各程序段完成的功能。

## 4.4 健壮性

本算法对于边界条件，诸如输入负数个、0个数据或非法字符有错误提示，并且对于n=1的情况也有特定输出。

# 5 实验感想

哈夫曼编码是具有时代性突破的成果，在旧时代欧洲的战争中，人们直接传输信息往往使用电报作为媒体，而时常会出现电报还未翻译完毕或传输完毕就被敌军侵占的情况，并且电报的长度也关乎着电报制作的成本和翻译时间，这更关乎着一个战争的走势。为此当时人们也意识到了，使用较短的编码来表示常用的字符，而使用较长的编码来表示不常用的字符，这样可以使得电报的长度相对简短，而当时的摩斯密码也只是尽可能采用这种方式，并没有一套理论基础来支撑。直到哈夫曼编码的问世，人们得到一套最为简洁的编码方式，至此电报传输的效率也得到了极大的提高。

哈夫曼编码的思想就是把力气用到刀刃上，这也启示着我如果想取得成功，要把时间精力用在有意义的事情上，对于相对而言没那么重要的事情，可以先放一放，先完成更重要的事情。

本次实验设计也将纸面上的离散数学知识升华到软件工程专业最重要的编程能力上，让我们在学习知识的同时还能应用于实际，把理论和实际相结合，使学习更高效。