《数据结构》课程设计总结

指导教师：柳先辉

学　　号：1652286

姓　　名：李辉

专　　业：计算机科学与技术

2020 年 8 月

目　录

第一部分 算法实现设计说明.....1
1.1 题目.....1
1.2 软件功能.....1
1.3 设计思想.....2
1.4 逻辑结构与物理结构.....3
1.5 开发平台.....5
1.6 系统的运行结果分析说明.....6
1.7 操作说明.....6
第二部分 综合应用设计说明.....10
2.1 题目.....10
2.2 软件功能.....11
2.3 设计思想.....12
2.4 逻辑结构与物理结构.....14
2.5 开发平台.....16
2.6 系统的运行结果分析说明.....16
2.7 操作说明.....17
第三部分 实践总结.....21
3.1.所做的工作.....21
3.2.总结与收获.....21
第四部分 参考

第一部分 算法实现设计说明

1.1 题目

6. 哈夫曼树、编码：

给出一组关键值，建立哈夫曼树，显示该哈夫曼树，并给出每个关键值的哈夫曼编码。

说明：关键值的获得可以选择通过 以下途径：

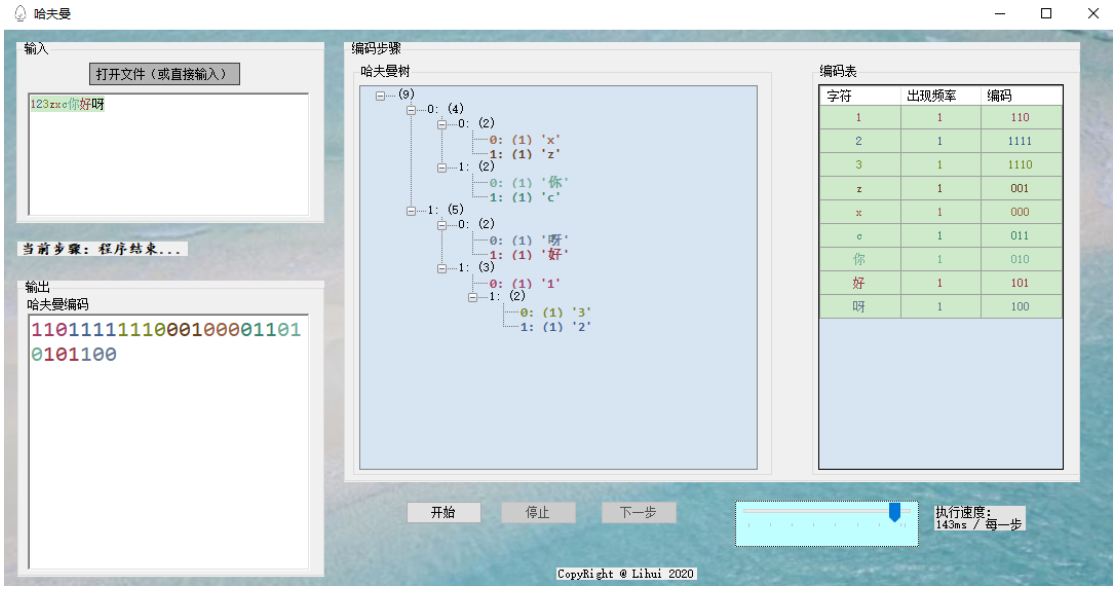
（1）给定的一组关键值；

（2）给定的一个文本；

（3）随机输入的一段文本。

1.2 软件功能

首先给出主页面显示：



该软件可实现如下功能：

1. 用户可以通过点击“打开文件”按钮打开本地计算机中任意一个文本文件，点击确认将其文本导入到程序输入文本框中，也可以直接在文本框中通过键盘键入文本。

2. 文本支持 UTF-8 格式，可以随意输入英文、数字和中文交给程序运行处理。

3. 点击“开始”按钮可以开始程序，点击后该按钮变成“终止”按钮，点击即可终止程序。程序执行过程中，首先将字符逐个添加到右侧的编码表中，新添加一栏或者频率加一，正在处理的字符在文本框和编码表内都会有高亮颜色突出显示，不同的字符也会随机赋予不同的颜色显示，进行有效区分。

4. 会按照字符出现频率将字符从上到下排序，出现频率即对应权重，首先将所有字符输入到树结构中当做初始节点，然后按照构建哈夫曼树的算法逐渐把节点按权重升序添加到哈夫曼树中，中间的树视图会将这一过程进行动态展示，过程中会对当前添加节点进行高亮颜色突出显示。

5. 哈夫曼树建好之后，根据各个节点在树上的位置，对各个字符进行编码，显示到右侧的编码表中。

6. 获得了字符编码后，即可对字符串的哈夫曼编码进行动态输出，输出在左下角的富文本框中，同样的，输出过程会有当前字符及其对应编码的高亮显示。

7. 点击“停止”按钮可在程序执行过程中暂停程序，点击后按钮变为“继续”，再次点击即可继续执行。

8. 当上一步点击“停止”后，可以点击“下一步”按钮，一步步的进行程序运行，仔细观察程序运行的每一步。

9. 配有速度拖动条可以动态调节程序执行速度，每一步 1ms-1800ms 的变化区间，非常灵活，是通过控制 timer 控件的间隔来实现的。

10. 程序会显示当前的运行步骤，主要有五个阶段（终止，读入，建树，编码，输出）

11. 随机生成颜色种子，不同的字符会有不同的颜色显示，包括其对应的编码。

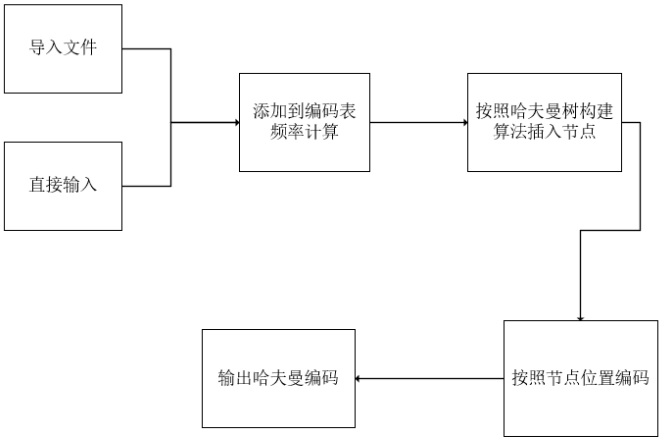
12. 通过 ToolTip 控件，当程序执行完后，鼠标落在树节点上会显示树节点的路径权重编码信息，直观明了。

1.3 设计思想

可视化图形界面采用 C# .net framwork 编写，初步接触窗体程序设计，感觉和网页页面的前端后端类似，“前端”即窗体程序显示界面，通过.net framework 框架工具箱里的各个组件灵活进行搭配，至于“后端”，即各个模块对应的处理程序代码及其相应的算法。题目要求通过多种方式读入字符串，我考虑通过调用计算机本地文本文件或者直接输入两种方式输入字符串，这就需要一个按钮调用 openFileDialog 组件和一个文本输入框，题目要求给出哈夫曼树的构建显示过程并输出哈夫曼编码，没有涉及到解码过程，这就简单多了，哈弗曼编码的过程简单说就是先对所有字符排序，把它们当做初始节点，然后按照权重由小到大一个一个往哈夫曼树里增添。最后根据字符在哈夫曼树中的位置，进行相应的 0、1 编码，这样每个字符都会获得自己的一个独有的编码，最后按照字符串中字符的顺序依次输出编码，初始想法是这样。因此需要输入模块、控制模块、执行模块、输出模块，每个模块都会有不同的属性和功能。因此，首先通过 VS 可视化操作将整体页面设计完善，然后进入代码页面赋予对应模块相应的功能，最后达到程序正常运转。之后考虑到调整程序速度或者逐步

执行以便详细观察哈夫曼编码过程，考虑添加 timer 计数器模块，通过控制其计时间隔控制程序执行速度，开始或者暂停。

程序按照以下步骤展开：



运行过程可分为以下几个过程：

1. 将字符逐个添加到编码表中，计算频率并进行排序

2. 按照各个字符出现频率（权重）有小到大插入到哈夫曼树中

3. 根据字符在书中节点位置给出对应 0、1 编码

4. 根据字符编码输出字符串哈夫曼编码

1.4 逻辑结构与物理结构

存储数据用的.net framework 表格组件 DataGridView，实质是二维数组，属于逻辑结构。

```
public class DataGridView : Control, ISupportInitialize
public DataGridViewCell this[string columnName, int rowIndex] { get; set; }
public DataGridViewCell this[int columnIndex, int rowIndex] { get; set; }
```

树结构用的.net framework 树视图组件 TreeView，其节点表示为

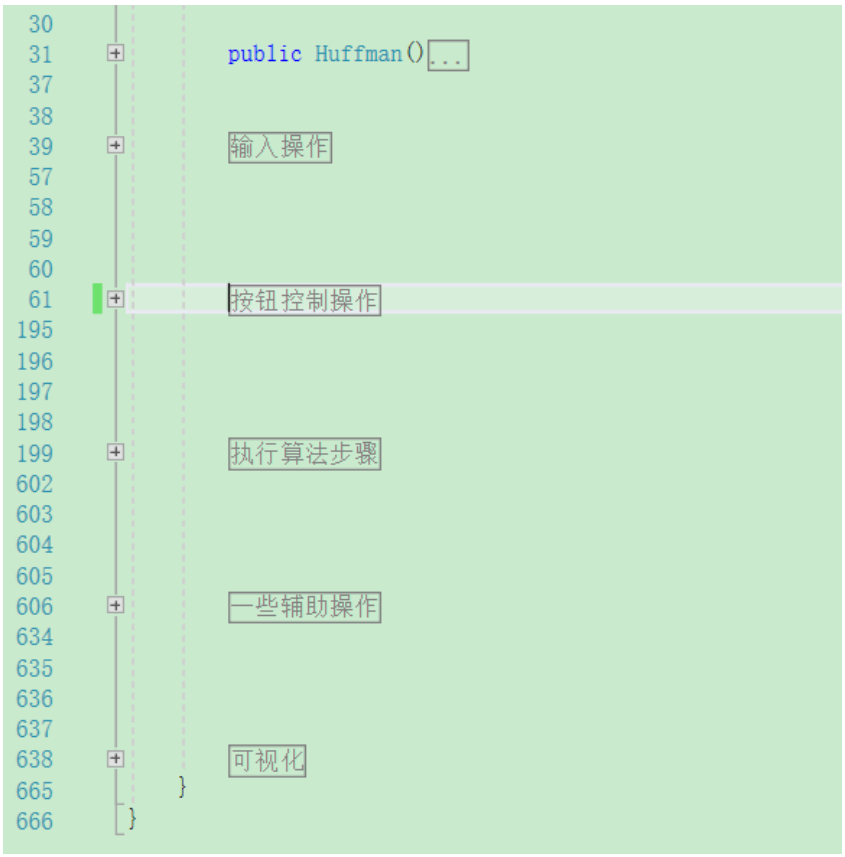
```
public class TreeNode : MarshalByRefObject, ICloneable, ISerializable
public TreeNode(string text, TreeNode[] children)
```

本软件实例中其 children 数一直为 2，即

```
public TreeNode(string text, TreeNode *left,TreeNode *right)
```

由于 C#没有指针，所以采用连续的数组表示，树结构属于链式存储，是物理结构。

重要函数（方法）：



输入操作：

private void button1_Click(object sender, EventArgs e)//导入文本文件

按钮控制操作：

private void richTextBox_input_Enter(object sender, EventArgs e)//获取焦点

private void richTextBox_input_Leave(object sender, EventArgs e)//失去焦点

private void button2_Click(object sender, EventArgs e) //点击开始按钮

private void trackBar1_Scroll(object sender, EventArgs e) //拖动速度条

private void button3_Click(object sender, EventArgs e) //点击暂停按钮

private void button4_Click(object sender, EventArgs e) //点击下一步

程序执行：

private void timer1_Tick(object sender, EventArgs e)//计时器

private void Next_Algrithm_State()//下一个状态

```
private void Read()  //读入
private void BuildingTree()   //构建哈夫曼树
private void GenerateCode()  //编码
private void Walking(string PathWay, TreeNode No)//朝左，朝右
private void GenerateOutput()//输出，到DataGridView
private void Painting(string PathWay, TreeNode No, bool CleaningUp)
//richTextBox_output输出
private void richTextBox_output_TextChanged(object sender, EventArgs e)//文本框
内容改变
private void dataGridView1_RowStateChanged_1(object sender,
DataGridViewRowStateChangedEventArgs e)//DataGridView编码表状态改变
```

1.5 开发平台

CPU:AMD Ryzen 7 4800H with Radeon Graphics 2.90GHz

系统类型：Windows10 64位，基于X64

开发环境：Visual Studio 2019 professional

Visual C#

.Net Framework

调用的系统命名空间：

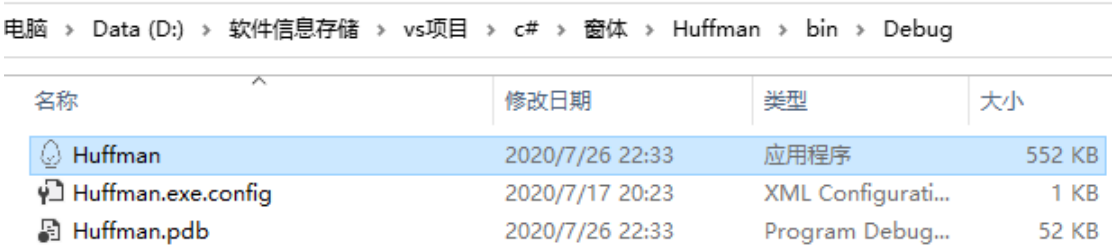
```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.IO;
using System.Text;
using System.Windows.Forms;
```

1.6 系统的运行结果分析说明

利用 Visual Studio C# .Net Framework 集成开发环境，先设计好整体页面，将各组件布置到位，然后依次完成输入、控制、运行（数据插入数组，排序，插入树节点，输出哈弗曼编码）以及其他的一些操作的逻辑构建，代码编写，逐步调试，如果出现错误，则通过加入断点，判断出错位置，进行修正。经过验证，本软件运行正确，与手写推倒的结果一致，运行过程稳定，并且可以在运行中选择暂停、继续、加速、减速等操作，稳定性良好。考虑了没有输入文本就开始运行等状况，容错性良好。基本上达到了题目的要求并且进行了一些功能的优化和改进。虽然题目没要求进行解码，但我认为解码也是很有必要的，目前没有加入这一模块，这是一个缺点和遗憾，另外，界面美观度,UI 设计也有待改善。

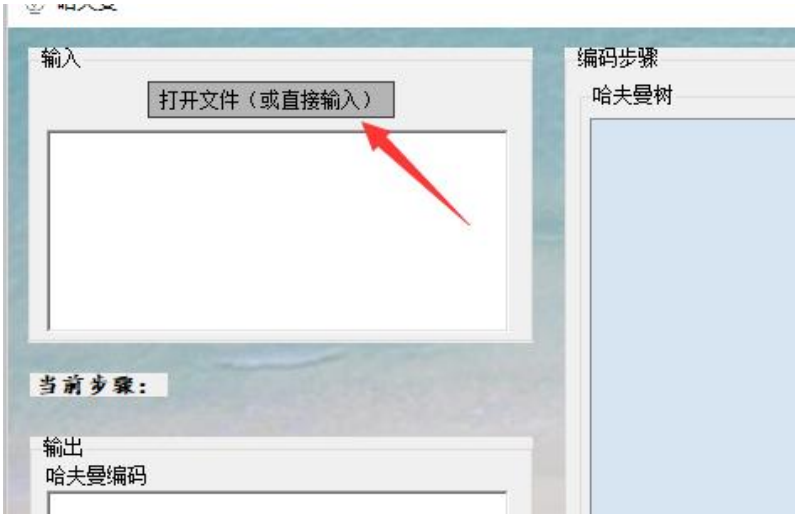
1.7 操作说明

1、打开可执行文件

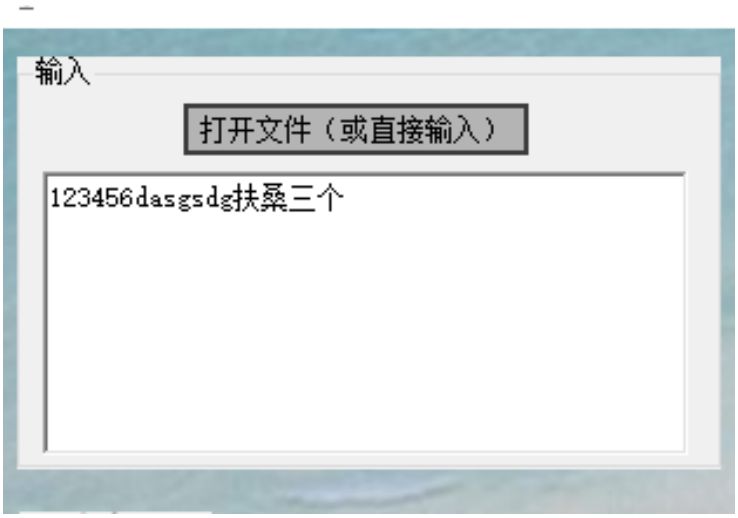


程序界面：

2、点击“打开文件”，导入一个有内容的 txt 文本文件



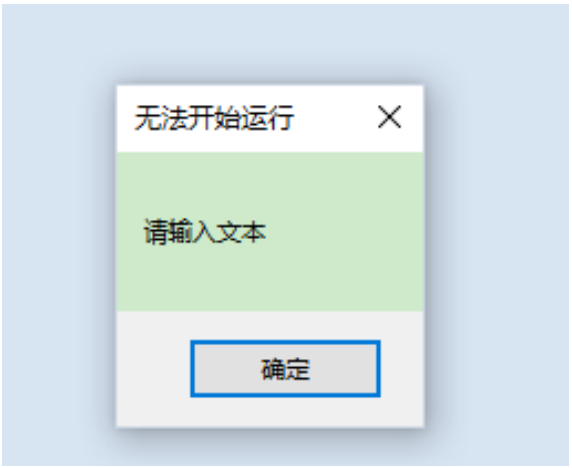
会在下面文本框显示导入文件内容（UTF-8）



或者直接在文本框键入文本：

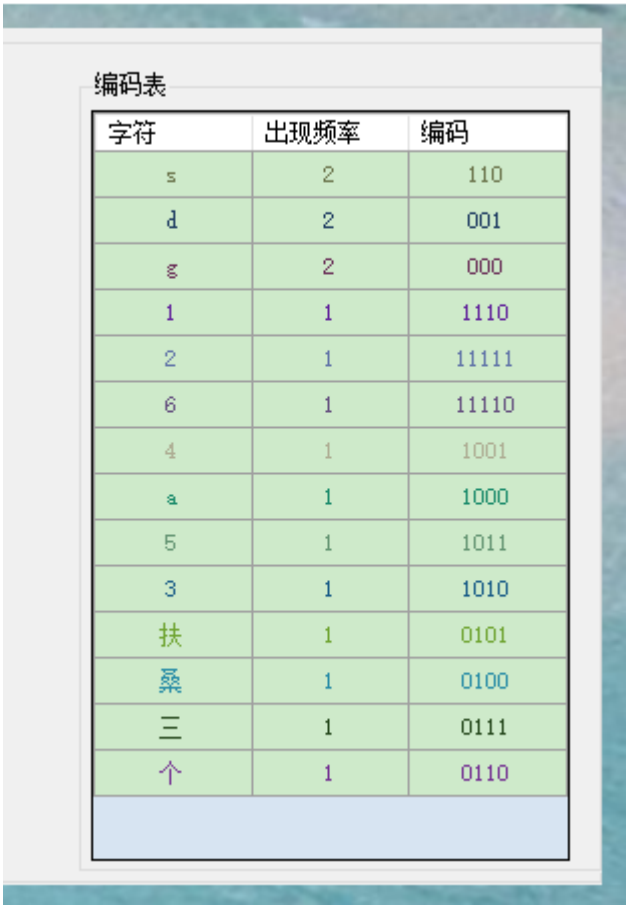
3、点击“开始”按钮，在程序执行之前另外两个按钮是灰色的

若没有文本输入，会报错提示：



4、途中可以选择“终止”或者“暂停”或者“下一步”操作，依据需要选择

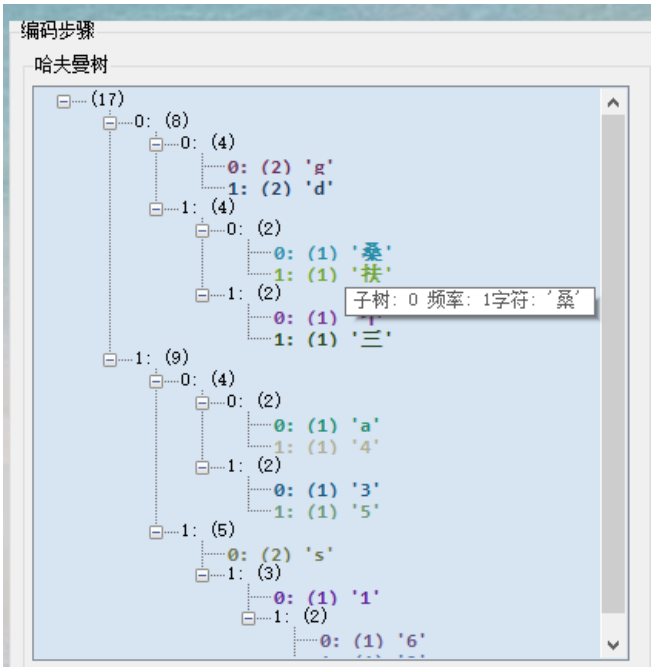
5、编码表将字符存储并排序，出现频率对应哈夫曼树中的节点权重



编码表

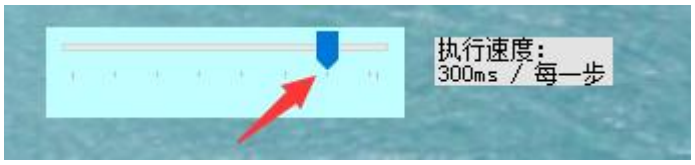
字符	出现频率	编码
s	2	110
d	2	001
g	2	000
1	1	1110
2	1	11111
6	1	11110
4	1	1001
a	1	1000
5	1	1011
3	1	1010
扶	1	0101
桑	1	0100
三	1	0111
个	1	0110

6、通过 C# .Net Framework 控件生成的哈夫曼树，鼠标放在节点上面会显示相关信息



7、可以在输出文本框看到输出的哈夫曼编码，并且每一段编码与其字符是相互对应的

8、过程中可以通过拖动速度条改变运行速度



第二部分 综合应用设计说明

2.1 题目

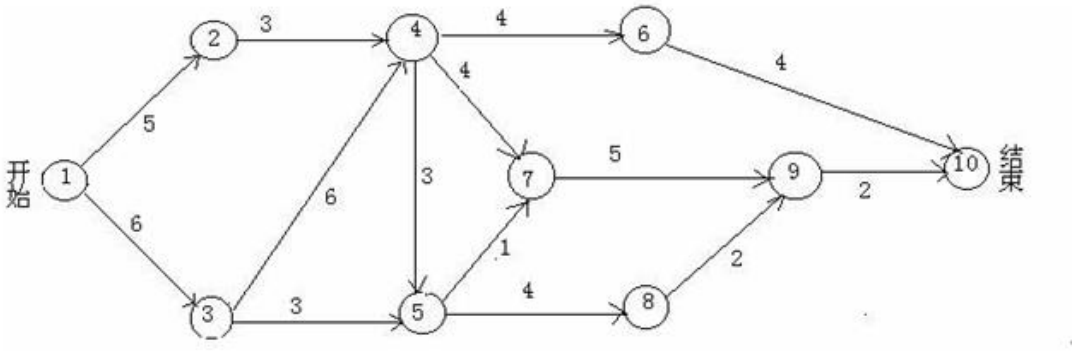
6. ★★已知假想的工程活动图 AOE 网，试设计一个算法，要求：

（1）判断工程是否可行；

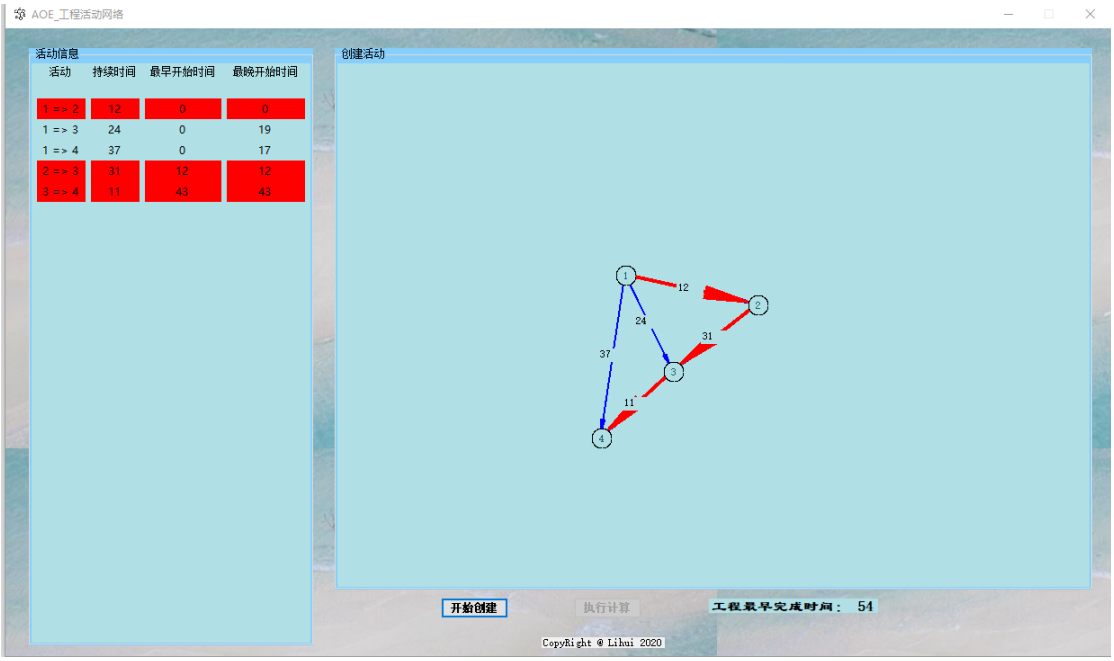
（2）求出工程中每个活动的最早开始时间 e(i)，最迟开始时间 l（i）和全工程可以完成的最早时间；

（3）确定工程中关键路径和可使整个工程的工期缩短的关键活动。

说明： 可以给出一个假想的工程活动图 AOE 网，如下图所示（仅为示例），也可以给出工程的活动情况，画出 AOE 网。



2.2 软件功能



本软件有如下功能：

用户可以点击“开始创建”自行建立 AOE 工程网络，鼠标点击创建面板生成一个事件，依次有序点击两个事件可以创建活动，创建的活动上会显示活动的权重。AOE 活动网络建成之后，点击“执行计算”，程序自动完成关键路径的计算，若构建的网络不满足条件，比如有多个源点、汇点、环路，或者事件已经创建，程序均会进行相应报错。当完成关键路径的计算后，会在创建面板以红色加粗显示关键路径，并且在左侧活动信息面板详细显示各个活动的信息，关键活动路径同样给予突出显示，在程序窗口右下角，会给出整个工程最早的完成时间，即关键路径活动时间之和。用户可以重复创建和计算关键路径的过程。

用到的主要控件是 panel 面板和 tableLayoutPanel 面板，用于在程序内 GroupBox 划分空间的基础上划分面板，事件的生成是生成一个按钮，连线的生成通过调用 System.Drawing.Drawing2D 的画图功能，活动信息的展现是增添表格。关键路径的计算采

用的是课本上的计算方法，在此之间需要先进行 AOV 的拓扑排序，确定没有环路方可继续程序运行。

2.3 设计思想

本软件仍然是利用 C#的.Net Framework 来写的图形界面。主要目的是能够生成或者创建一个 AOE 活动网络，并对这个网络进行求解关键路径。在 C++里用指针表示的邻接表很容易实现活动网络图结构的存储，然而 C#虽然构建图形界面比较方便，但是由于没有指针这一特性，很多方面会受到限制，只能用静态的数组来模拟指针构建邻接链表。主要采用三个类来表示 AOE 活动网络：活动类、事件类和 AOE 网络结构类，活动类包括活动的始终，活动的权重，活动的最早和最晚开始时间以及活动的起点终点坐标；事件类包含以该事件为起点的所有的活动的变长数组，相当于邻接表的某一行；AOE 网络类中包含各种求解过程中要用到的方法。难点在于打破 C++的指针思维方式，以变长数组构建邻接表。

主要算法思想：

1、 求活动最早开始时间：活动出发顶点（时间）的最早开始时间

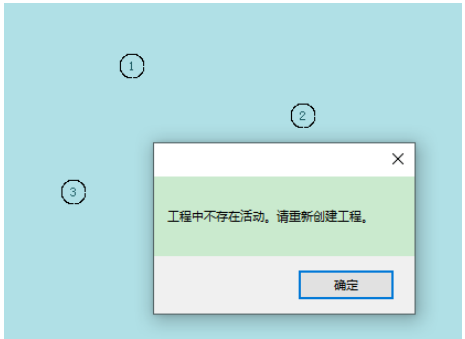
2、 求活动的最晚开始时间：从要求最晚截止时间开始，反向推导，最晚工程技术时间依次减去路径上各活动的时间

3、 求关键路径和关键活：根据各顶点的 ve 和 vl 值，求每个活动的最早开始时间 e(s)和最迟开始时间 l(s)，若某条弧（活动）满足条件 e(s)=l(s)，则该活动为关键活动，关键活动连接起来就是关键路径。值得注意的是：关键路径可能有多条。

容错性考虑：

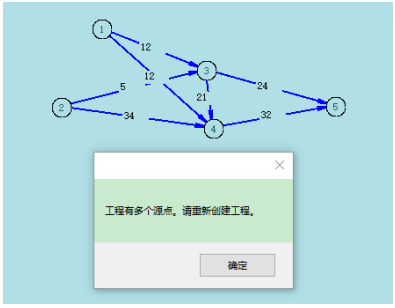
1、 判断工程中是否创建事件和活动

遍历所有事件节点，检查以该节点开始的活动的可变长数组是否是空的，相当于检查邻接表是否为空，若是空的，则工程中不存在活动，需要重新创建。



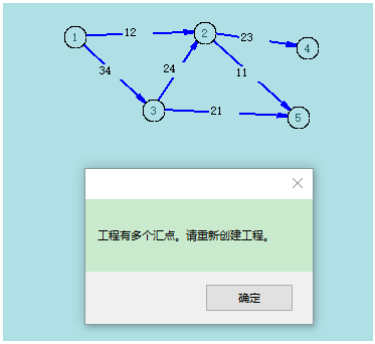
2、 检查工程中是否有多个源点：

遍历所有事件节点，统计所有入度为 0 的节点的数量总和，如果数量大于 1，则说明不止一个源点，工程不符合条件，不能求解。



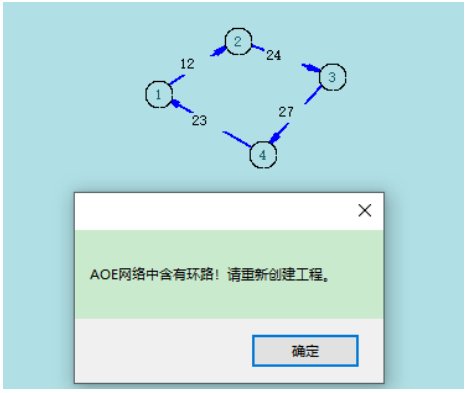
3、 检查工程中是否有多个汇点：

遍历所有事件节点，统计以该事件节点开始的活动的可变长数组为空的事件个数，如果统计出的个数大于 1，则说明有多个事件为截止事件，工程中存在多个汇点，无法求解关键活动。



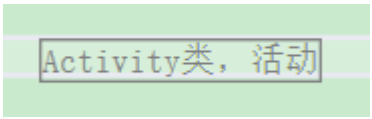
4、 检查工程中是否有环路：

AOE 活动网络中是不存在环路的，为检查活动网络中是否有环路，需要对所有事件节点进行拓扑排序，如果排序后的离散节点数与之前的节点数相同，则 AOE 活动网络中不存在环路，否则，需要重新创建工程。



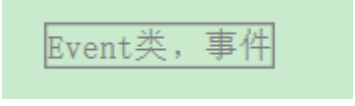
2.4 逻辑结构与物理结构

程序主要构成：



Activity 类：

```
public class Activity
    {
        public int start;         // 活动起始事件序号
        public int end;          // 活动结束事件序号
        public int weight;       // 活动持续时间，权重
        public int e;            // 活动最早开始时间
        public int l;            // 活动最迟开始时间
        public bool critical;    // 是否为关键活动
        public Point p_start = new Point();    // 活动起点坐标(画图用)
        public Point p_end = new Point();    // 活动终点坐标(画图用)
    }
```



Event 类：

```
    public class Event
```

```
    {
        public bool critical;  / 是否为关键路径上的事件
        public List<Activity> next = new List<Activity>();  // 以该事件为开始的
活动
        public int ins; // 入度
        public int e; // 事件最早发生时间
        public int l; // 事件最迟发生时间
    }
```

Network类，网络

NetWork 类：

```
public class Network
    {
        Graphics g;
        public List<Event> net = new List<Event>(); // AOE 网邻接表
        public int start;    // 工程起始事件序号
        public int end;    // 工程结束事件序号
        List<int> topology = new List<int>();   // 拓扑排序结果
      }
```

主要的数据结构是 List<T>形式的动态变长数组，该数组在计算机中占据连续的内存，属于物理结构。

程序计算的主要方法（函数）

```
private void button1_Click(object sender, EventArgs e)  //开始创建
private void panel1_MouseClick(object sender, MouseEventArgs e)  //画 AOE 网络
private void eventClick(object sender, MouseEventArgs e)  //点击事件
private void button2_Click(object sender, EventArgs e)//执行计算
protected override void OnPaint(PaintEventArgs e)//重新设置控件的形状，绘制圆形
按钮
```

```
public String Is_Able_Continue()   // 求解工程是否可行，可继续
public void eventsTime()   // 求各事件的最早和最迟开始时间
public void activitiesTime(AOE main)   // 求各活动的最早和最迟开始时间
```

2.5 开发平台

CPU:AMD Ryzen 7 4800H with Radeon Graphics 2.90GHz

系统类型：Windows10 64 位，基于 X64

开发环境：Visual Studio 2019 professional

Visual C#

.Net Framework

调用的系统命名空间：

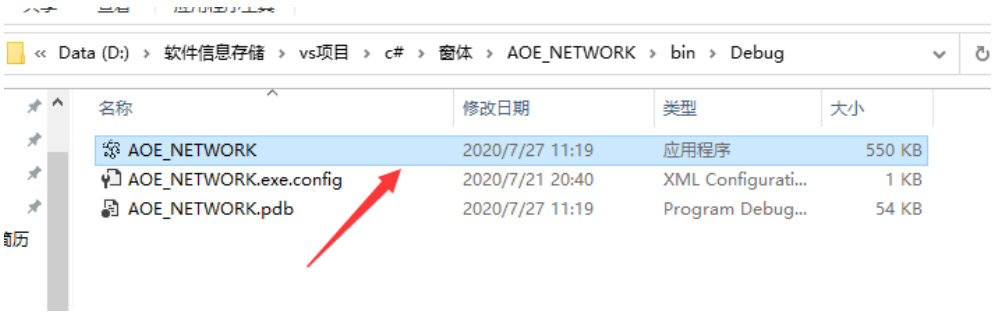
```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;
```

2.6 系统的运行结果分析说明

利用 Visual Studio C# .Net Framework 集成开发环境，先设计好整体页面，将各组件布置到位，主要部件为 panel 控件和 tabelLayoutPanel 控件，主要调用的绘图函数为 System.Drawing.Drawing2D 下的 DrawLine(pen, start, end)。题目要求算出 AOE 工程网络中的关键路径。本软件已达到基本要求，并且可以自己灵活创建 AOE 网络，网络中标出活动标号，和活动权重，当计算完毕后，会对关键路径加粗加红显示，关键活动的信息也会加红显示。经过验证，正确性良好，稳定性良好，已经充分考虑各种错误情况并且给予相应的处理，容错能力良好。并且给予相应的提示，比较人性化。

2.7 操作说明

1、打开可执行程序

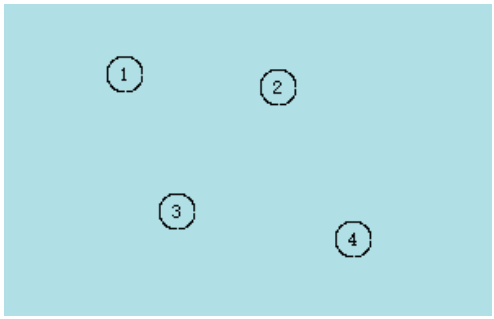


程序初始界面：

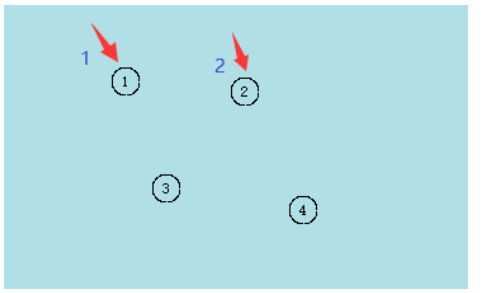
2、点击“开始创建”按钮，开始创建工程

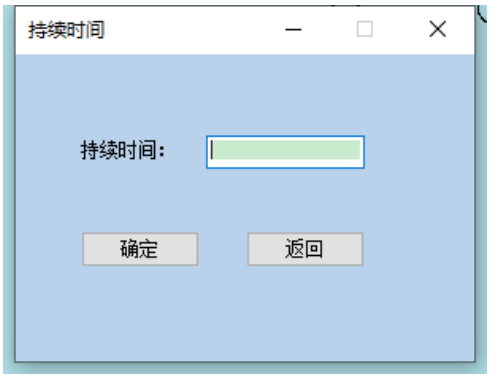
4、 创建活动

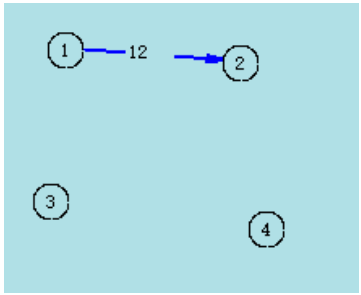
鼠标在面板上点击，生成事件



鼠标依次按照顺序点击活动上的两个事件，开始创建活动

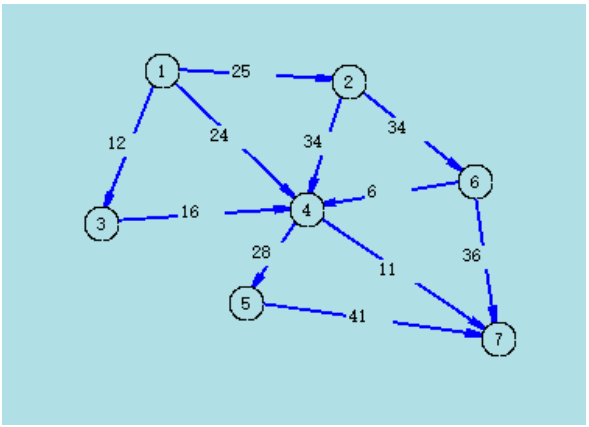




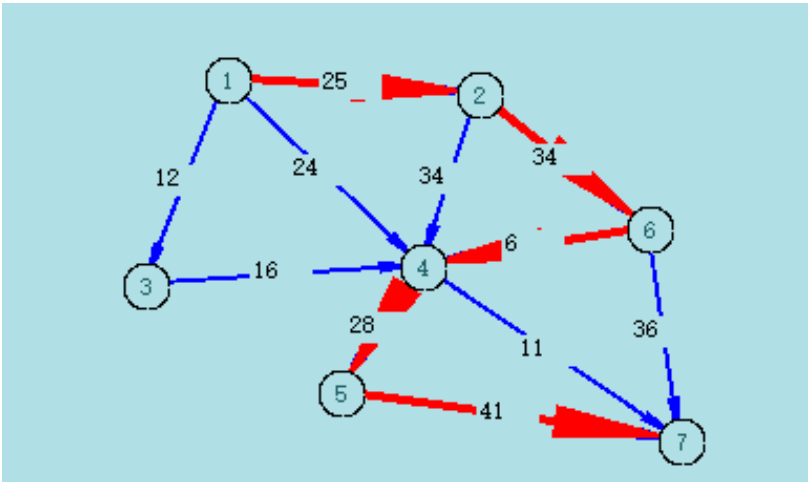


5、 创建完活动，点击“执行计算”

计算前



计算后：关键路径加粗加红显示

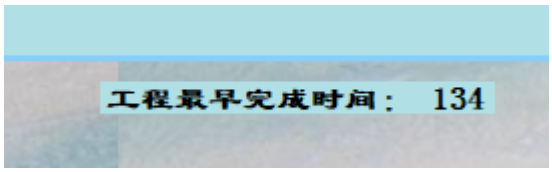


活动信息：关键活动加红突出显示

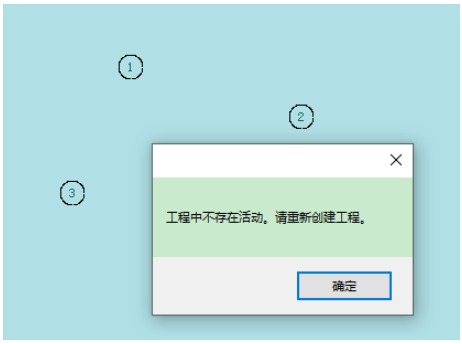
活动信息

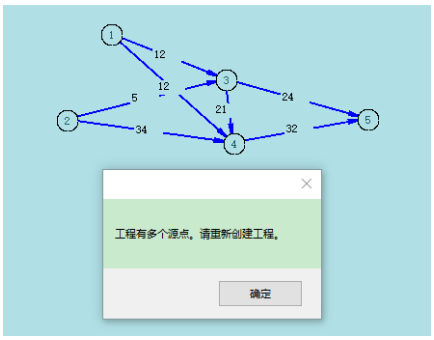
活动	持续时间	最早开始时间	最晚开始时间
1 => 3	12	0	37
1 => 2	25	0	0
1 => 4	24	0	41
2 => 4	34	25	31
2 => 6	34	25	25
3 => 4	16	12	49
4 => 5	28	65	65
4 => 7	11	65	123
5 => 7	41	93	93
6 => 4	6	59	59
6 => 7	36	59	98

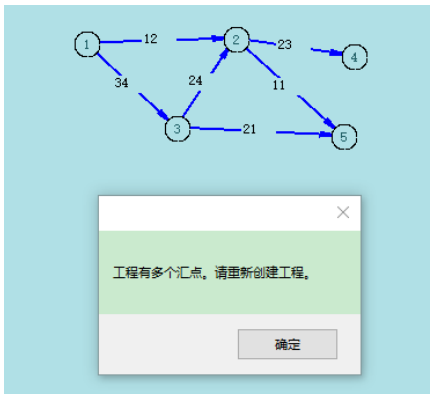
显示工程最早完成时间：

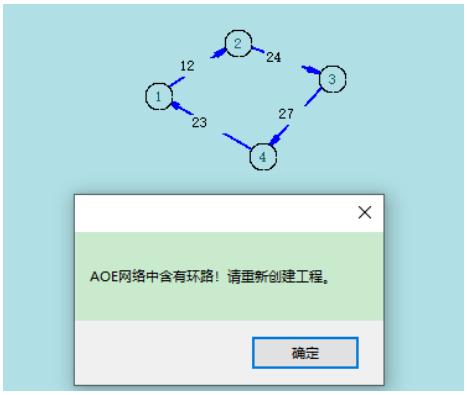


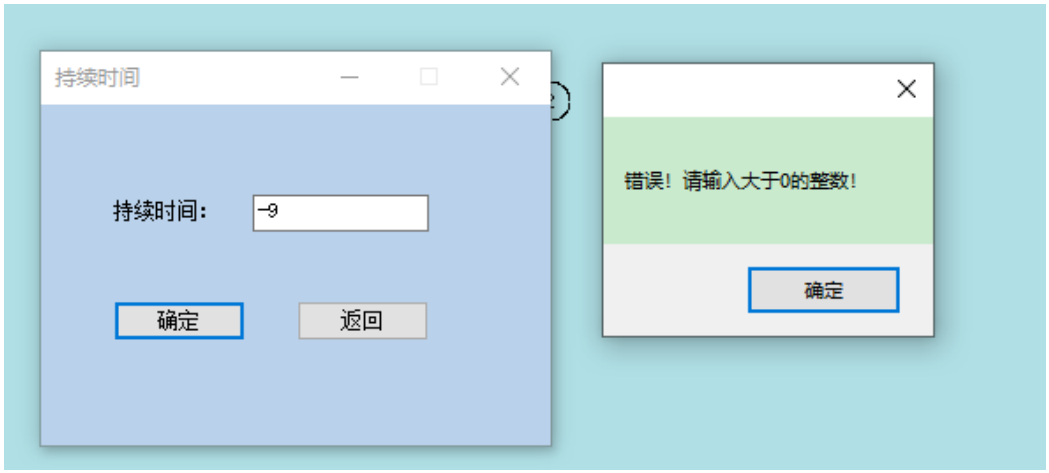
6、 错误提示











第三部分 实践总结

3.1.所做的工作

1、复习数据结构课程中所学的知识点，重点复习哈夫曼树构造算法、关键活动计算方法以及树和图的存储形式

2、学习 C#基本语法

3、学习 C# winform 窗体基本构建，各个控件的特性，使用方法

4、设计程序用到的数据结构及选择相应合适的.Net Framework 窗体控件

5、构建窗体，完善各个部件的具体功能

6、通过参考书籍和网络资料解决问题，不断改进，提高容错性和用户体验

7、总结复盘，回顾反思

3.2.总结与收获

《数据结构》课程设计是对《数据结构》课程的加强和巩固，通过实践做图新界面软件的形式强化对数据结构的理解以及加强对各种开发环境的掌握应用，通过完成《数据结构》课程设计，个人受益匪浅。

着手之初，不知道该从何下手，数据结构知识还好，学期内已经有大量练习了，对堆、栈、树、图等已经有了比较清晰的理解，难点在于图形界面的制作，这点在课程中是没有接触到的。所以只好先查阅资料，经过检索，发现图形化界面可以用 Qt、easyX、MFC、C#等来做,这些与比较熟悉的C++比较接近,有些是在C++上的直接扩展,其他的包括Java、Python等都有很成熟的库可以调用生成图形界面，但是对这些语言还不是很熟，所以果断放弃。后面经过大致的浏览，发现 C# .Net Framework 窗体程序构建比较容易操作，而且目前 C#在很多开发的地方均有应用，所以想趁着这个暑假完成课设的机会把 C#及其开发知识顺便给学了。首先是通过官网和书籍进行基本的 C#语法学习，然后在官网上进行.Net Framework 窗体组件的学习，对 C#工具箱里的大部分组件都已经了解了一二。

以上准备工作做完，应该是可以开动做课设了，但发现没那么简单，因为一些数据结构在 C++里面可以很简单表示出来，但在 C#里面就不太好表示，一方面有自己不太熟的原因，另一方面，C#的许多机制与 C++不相同，比如 C++里的指针 C#就没有，树和图在 C++里用指针很容易表示，C#语法却比较难表示。最后通过不断查阅资料完善知识框架，可以通过.Net 里面的一些已有组件，比如 treeView 和 DataGridView 等， 这些组件里面已经内嵌了树结

构或者数组结构，不用为没法用指针表示而烦恼，不过深入学习这些组件的框架和使用方式也是花费了不少时间和精力。在学习 C#基础的时候，知道了 C#的委托机制和 C++的指针类似，也可以利用静态数组来模拟指针，C#的委托机制是高级用法，具体实现起来比较麻烦，相对对静态数组来模拟链表之类的操作可能比较容易上手，所以在 AOE 网络中用 List<T>变长数组来表示邻接链表。

做完课程设计，收获很大，感觉到了平时做的练习题与这种实际的软件还是有比较大的差距，但是课程里学的数据结构知识都是这些程序的内在骨架和灵魂，把程序比作前端的话，数据结构和算法知识就是后端，支撑着程序运行的逻辑。难点在于对开发环境和知识的掌握和理解，要求短时间内掌握窗体设计方法并融合学习过的数据结构知识。整体做下来发现其实课程设计不是很难，当熟悉了各组件的应用方式和与数据结构的结合，剩下的就是实现一些简单的逻辑了，C#与 C++大部分相似，类和方法的概念都差不多，但没有指针表示，不同的语言和环境之间得做出取舍和相应的改变。其实无论是采用什么开发环境，万变不离其宗，数据结构和算法知识是不会改变的，只要掌握根本就好。

第四部分 参考文献

[1] 严蔚敏 吴伟民.数据结构（C 语言版）.北京：清华大学出版社，2007

[2] Matthew MacDonald.WPF 编程宝典——C# 2010 版.北京：清华大学出版社，2011

[3] Daniel M.Solis.图灵程序设计丛书：C#图解教程（第 4 版）.北京：人民邮电出版社,2012

[4] 菜鸟教程.C#教程. https://www.runoob.com/csharp/csharp-tutorial.html

[5] C 语言中文网.C# WinForm 界面设计教程（C# Windows 窗体应用程序）. http://c.biancheng.net/csharp/winform/

[6] Microsoft 官网.使用 .NET Framework 开发自定义 Windows 窗体控件. https://docs.microsoft.com/zh-cn/dotnet/framework/winforms/controls/developing-custom-windows-forms-controls

[7] 孙学琛,李新洁.哈夫曼树的图形化算法设计[J].山东理工大学学报(自然科学版),2008,22(06):108-110.

[8] 简书.关键路径算法演示（AOE 网）. https://www.jianshu.com/p/1857ed4d8128