《人工智能课程设计》
实验一：
A*算法求解8数码问题
（c++，c#）

同济大学电子与信息工程学院

班级：计算机科学与技术2班

姓名：夏文勇

学号：1851506

指导老师：王俊丽

2021 年 4 月 21日

Copyright By Tongji University

A*算法求解 8 数码问题

摘 要

8 数码问题是 A*算法的一大重要应用，而对于 A*搜索算法来说，算法的好坏与启发函数的评价标准设定有着直接的关系。同时为了优化算法，在算法执行的过程中构建了键值对和值键对的双 map，避免了需要遍历所有元素查找所消耗的时间。

本实验的具体实现分为 A*搜索树生成的内核部分与搜索树的绘制展示部分。内核部分使用 C++语言编写，主要是借助了 c++强大的 STL 容器；展示部分则使用 C#语言编写，利用其控件属性，很好的解决洁面问题。通过 dll 文件实现在 c#使用 c++函数，用以总体实现 A*搜索树的生成与绘制。

关键词：8 数码问题，A*搜索算法，dll 实现语言交互

Copyright By Tongji University

目录

Copyright By Tongji University

1. 实验概述

1.1.实验目的

熟悉和掌握启发式搜索策略的定义、评价函数 f(n)和算法过程，并利用 A*算法求解 8 数码问题，在代价最小的情况下将九宫格从一个状态转为另状态的路径在代价最小的情况下将九宫格从一个状态转为另状态的路径，理解求解流程和搜索顺序。

1.2.实验内容及要求

1.以 8 数码问题为例，实现 A*算法的求解程序（编程语言不限），要求设计两种不同的启发函数 h(n)。

2.设置相同初始状态和目标状态，针对不同的评价函数求得问题的解，比较它们对搜索算法性能的影响，包括扩展节点数、生成节点数和运行时间等。要求画出结果比较的图表，并进行性能分析。

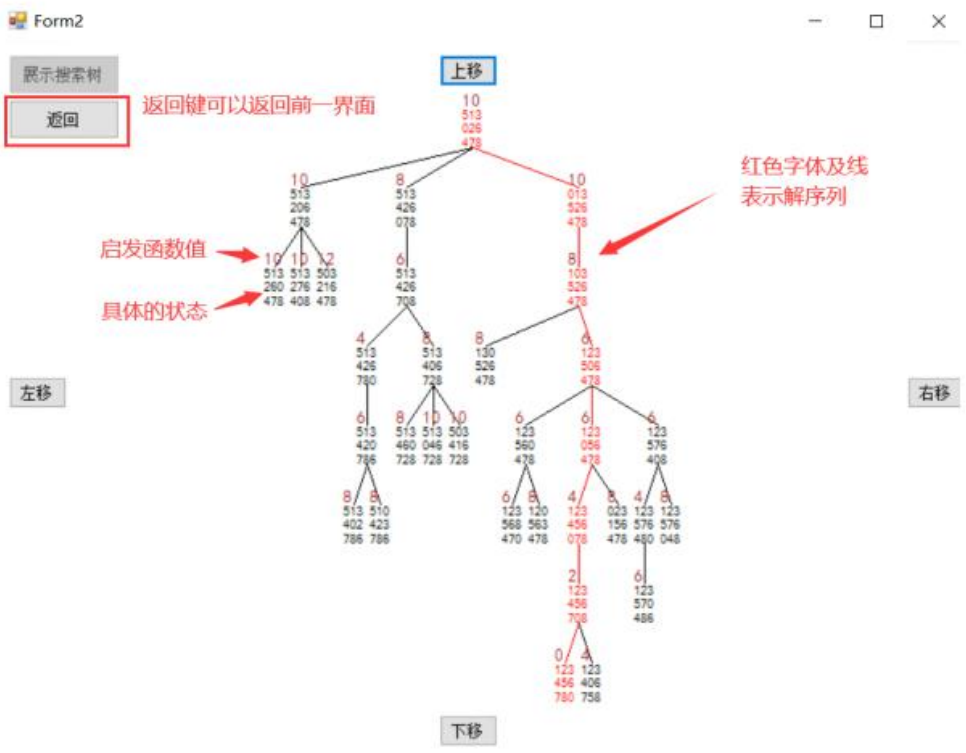
3.要求界面显示初始状态，目标状态和中间搜索步骤。

4.要求显示搜索过程，画出搜索过程生成的搜索树，并在每个节点显示对应节点的评价值 f(n)。以红色标注出最终结果所选用的路线。

5.撰写实验报告，提交源代码（进行注释）、实验报告、汇报 PPT。

Copyright By Tongji University

1.3.实验效果呈现



Copyright By Tongji University

2. 实验方案设计

2.1.总体设计思路与总体架构

整个程序总体上分为求解程序与界面展示两部分，求解程序主要使用 c++编写，充分利用 c++的 STL 容器，简化程序；界面展示部分主要是用 c#语言，利用其面向对象的特性，将功能具体到某个控件，方便图形化的展示。

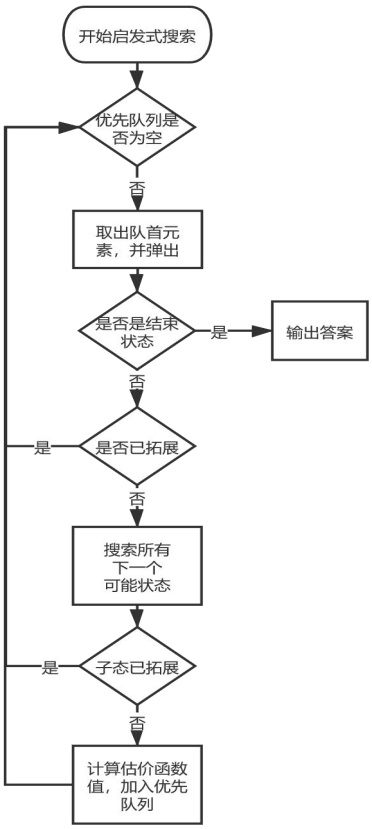
2.1.1 求解程序架构

1. 考虑到八数码呈现形式是一个 3*3 的数组，不好处理，先将其处理成一个 9 位的整数，将一组数据（9 位状态整数，估价函数值）作为整体；

2. 利用二叉堆这一数据结构实现优先队列，能够在 O(log(n))的时间复杂度内找出队列内优先级最高的数据；

3. 判断该数据是否是目标状态、是否已被拓展，如果都不是，就对其进行下一步的拓展，找出当前状态的所有子状态；

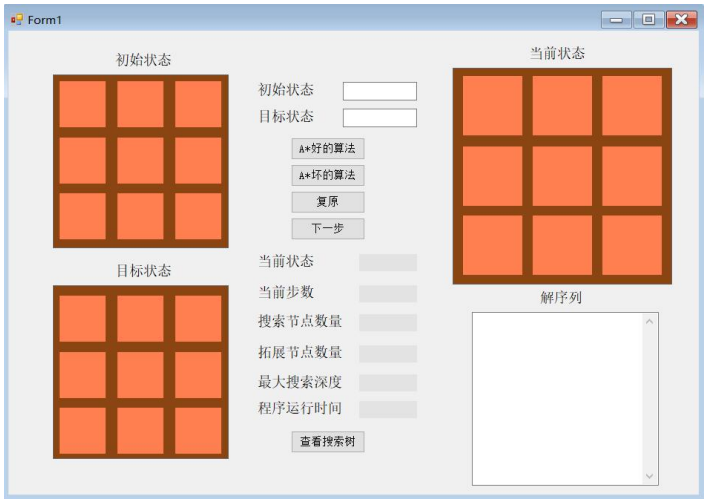
4. 计算拓展的子状态的估价函数值，将数据组（9 位状态整数，估价函数值）加入优先队列，返回第二步，开始下一轮搜索。



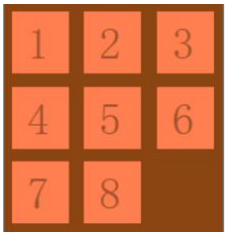
Copyright By Tongji University

2.1.2 可视化界面架构

整体框架与界面：



1. 以九个方格的形式展示初始状态和目标状态，方格固定；同时，另有九个可移动方块展示当前状态，当前状态可以受下一步、复原等操作的影响而变化。



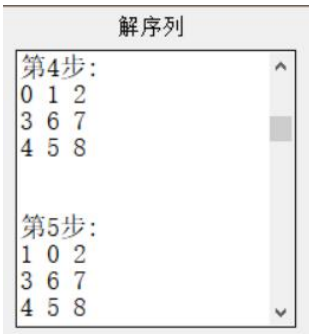
2. 设置输入框，用于接收用户的输入，比如初始状态和目标状态。

3. 设置开始按钮、下一步按钮、重置按钮，其中开始按钮又分为两种启发式函数，可供选择。

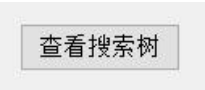
4. 由文本框和状态栏构成的状态显示部分，能够显示包括当前状态、当前步数、搜索节点数量、拓展节点数量、最大搜索深度、程序运行时间的各类信息。

Copyright By Tongji University

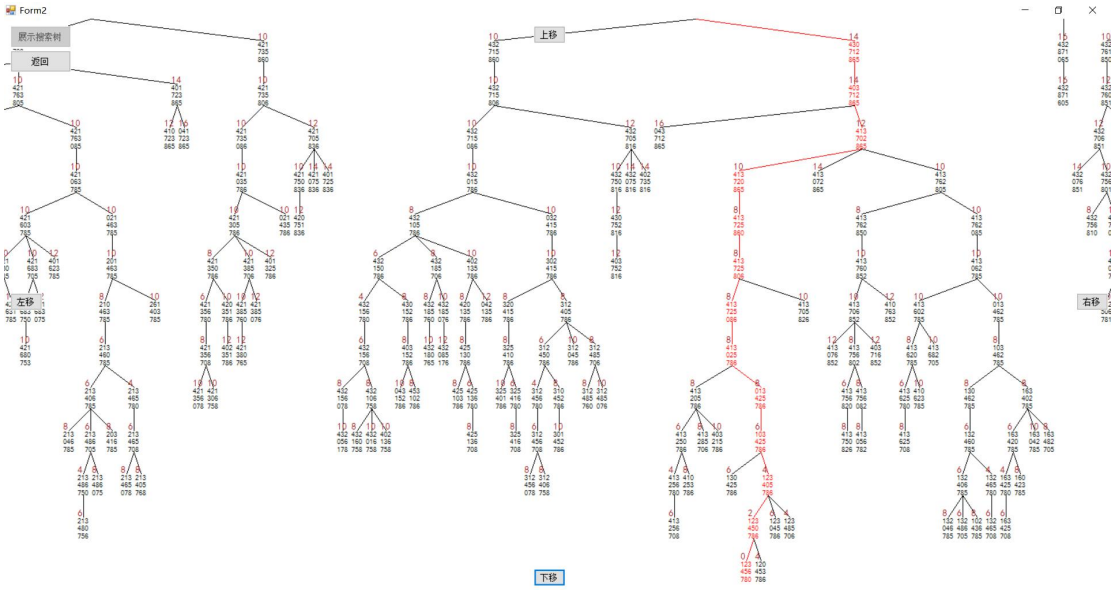
5. 解序列列表滚动展示解序列。



6. 搜索树查看界面，由主界面的“查看搜索树”按钮进入，在搜索树界面点击展示搜索树展示当前解的搜索树，点击返回可以返回主界面。



下图为某个搜索子树，用于展示本程序的子树可自由移动查看。



2.2.核心算法及基本原理

1. 优先队列：

利用二叉堆这一数据结构实现优先队列，每次新加入数据时，首先添加在叶子结点，然后依次与父节点比较，调整位置，直至二叉堆重新稳定，此时的最值就会保持在二叉堆的根节点；

优先队列能够实现每次添加或删除数据时，在 O(log(n))的时间复杂度调整二叉堆的顺序，以快速找出队列内优先级最高的数据；

2. 评估函数值计算：

Copyright By Tongji University

创建一个评估函数 f(n)=g(n)+h(n)用于每个结点，评估 “可取性”，确定哪个节点最有可能在通向目标的最佳路径上；其中 g(n)表示从初始节点 S 到达结点 n 已经花费的代价（实际代价），h(n)表示从结点 n 到目标结点 G 的最小代价路径的估计值，f(n)表示经过结点 n 的最小代价解的估计代价。配合优先队列，按可取性递减的顺序排列未扩展结点，扩展离目标最近的结点，可以很快的找到解。

在这里，我构造了多种评估函数用于结果的测试：

（1）$h(n)=\sum_{i=0}^{8}\left|nx_i-endx_i\right|+\left|ny_i-endy_i\right|$，其中$(nx_i,ny_i)$表示状态$n$中数字$i$的位置，$(endx_i,endy_i)$表示目标状态中数字$i$的位置，整个表达式的含义是，当前状态每个数字与目标状态的绝对距离。

（2）$h(n)=\sum_{i=0}^{8}(nx_i=endx_i)\&(ny_i=endy_i)$，其中$(nx_i,ny_i)$表示状态$n$中数字$i$的位置，$(endx_i,endy_i)$表示目标状态中数字$i$的位置，整个表达式的含义是，当前状态每个数字与目标状态数字所在位置相同的数量。

3. 判断是否有解：

一个状态表示成一维的形式，求出除 0 之外所有数字的逆序数之和，也就是每个数字前面比它大的数字的个数的和，称为这个状态的逆序，若两个状态的逆序奇偶性相同，则可相互到达，否则不可相互到达。

证明：当左右移动空格时，逆序不变。当上下移动空格时，相当于将一个数字向前（或向后）移动两格，跳过的这两个数字要么都比它大（小），逆序可能±2；要么一个较大一个较小，逆序不变。所以可得结论：只要是相互可达的两个状态，它们的逆序奇偶性相同。

4、深度优先搜索绘制搜索树：

首先利用深度优先搜索，计算所有节点的宽度，其中节点宽度是其所有子节点宽度的和，对于叶子结点，其宽度设为节点矩阵所需要宽度即可；接下来，再次深度优先搜索，对于每个节点，在其所需要的宽度的中点输出答案即可。

2.3.模块设计

1. 数据处理模块

考虑到对状态的记录问题，八数码问题的呈现形式是一个 3*3 的数组，与常规数据不同，因此在记录状态时，可以将其转化成 9 位整数，方便状态的记录；而在需要进行移动或具体计算时，再将整数转化回数组。

在这里可以构造函数，专门用于两者的转化，使得程序更加简洁。

Copyright By Tongji University

2. 搜索函数模块

搜索函数利用优先队列和评估函数，实现按可取性递减的顺序排列未扩展结点，每次优先扩展离目标最近的结点。

在确定了扩展节点后，还需要对其进行检验，避免重复扩展；然后利用其数组形式，判断当前状态的子态，计算得到其评估值后，加入优先队列。

3. 路径打印函数模块

一般来说，利用优先队列求解是不能保存路径的，因此需要额外的参数来记录每个状态的父亲状态，最后在输出答案时，逆向输出即可。

4. 搜索树打印模块

首先利用深度优先搜索，计算所有节点的宽度，其中节点宽度是其所有子节点宽度的和，对于叶子结点，其宽度设为节点矩阵所需要宽度；

接下来，再次深度优先搜索，对于每个节点，在其所需要的宽度的中点输出答案。

2.4.其他创新内容或优化算法

1. 哈希的使用

由于在这里记录状态需要使用 9 位的整数，无法直接使用数组来存储每个状态的详细信息（评估函数值、父亲节点等），因此在这里使用哈希的方法，将 9 位整数转化为能够存储的值，再存储其具体的信息。

2. c#与 c++的交互

首先将 C++代码编译成动态库 dll，在这一步需要使用 extern "C" _declspec(dllexport) 函数名，目的是为了使用 DllImport 调用非托管 C++的 DLL 文件，因为使用 DllImport 只能调用由 C 语言函数做的 DLL。

接着将 dll 拷贝到 c#项目输入目录，一般在 bin/debug 下面。

最后在 c#程序中添加

```
[DllImport("*.dll", CallingConvention = CallingConvention.Cdecl)]
extern static void  函数名(参数);
```

这样就可以在 c#程序中调用 c++的函数。

3. 在计算前，首先判断是否有解

一个状态表示成一维的形式，求出除 0 之外所有数字的逆序数之和，也就是每个数字前面比它大的数字的个数的和，称为这个状态的逆序，若两个状态的逆序奇偶性相同，则可相互到达，否则不可相互到达。

证明：当左右移动空格时，逆序不变。当上下移动空格时，相当于将一个数字向前（或向后）移动两格，跳过的这两个数字要么都比它大（小），逆序可能±2；要么一个较大一

Copyright By Tongji University

个较小，逆序不变。所以可得结论：只要是相互可达的两个状态，它们的逆序奇偶性相同。

3. 实验过程

3.1.环境说明：操作系统、开发语言、开发环境及具体版本、核心使用库等

操作系统：windows10 版本 20H2

开发语言：c++（数据处理）、c#（前端界面）

开发环境： Vscode Version 1.55

Visual Studio2019

核心使用库：

C++：map,queue

C#：System.Windows.Forms

3.2.源代码文件清单，主要函数清单

3.2.1 8digits.cpp

数据处理与计算源文件，用于实现具体的计算过程，用于答案求解。

1. 数据说明

```
int cntSearch, cntExpand, maxDeep;    //以搜索节点，已扩展节点，搜索最大深度
int begFlag, endFlag;          //目标状态，初始状态
struct point {          //节点详细信息
    int dn, fn, par;  //某节点的d[n],f[n],父节点
    changeInfo bef; //原状态
    changeInfo aft; //现状态
};
unordered_map<int, point>info;   //记录状态的详细信息
unordered_map<int, int>vis;      //记录状态是否重复搜索
```

2. 启发函数值计算函数，use 是状态数组，布尔类型 hao 表示是否使用好的启发函数

```
int calh(int use[3][3],bool hao)
```

3. 3*3 数组转化为 9 位整数

```
int mat2int(int use[3][3])
```

4. 9 位整数转化为 3*3 数组，n 表示 9 位整数，(x,y)记录数字 0 的位置

```
void int2mat(int use[3][3], int n, int& x, int& y)
```

5. 初始化函数，用于每次重新输入后数据的清空

```
void init()
```


Copyright By Tongji University

6. 具体的搜索函数，利用优先队列进行搜索

```
void solve(bool hao)
```

7. 用于输出答案，包括解序列、搜索序列、运行信息（时间、拓展节点等）

```
void printAns(double usetime)
```

8. 主程序，接受起始状态和结束状态，以及是否使用好的启发函数

```
void solve_ans(int start_, int end_,bool hao)
```

3.2.2 Form1.cs

用于展示所有按钮、数字块

1. 图形界面主界面

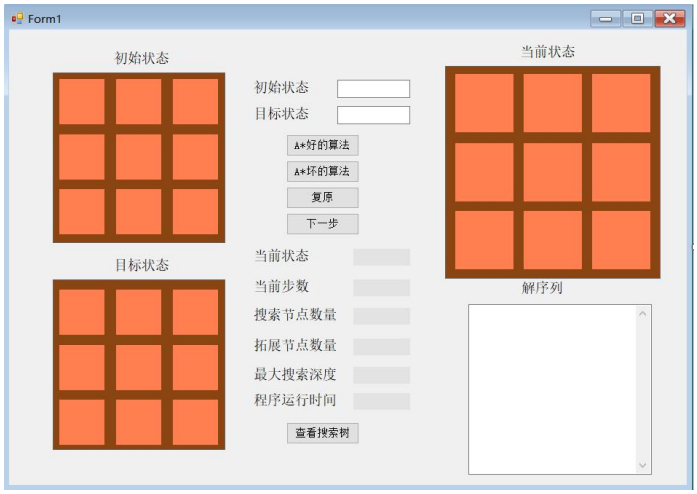


图 1. Form1 主界面

2. 数据说明

```
int[] nums = new int[10005];     //记录解序列
int[] use = new int[10];         //用于数组与数字转化的中间工具
int wz = 0;          //当前是第几步
int totSteps;        //总的解的步数
string begStatus, endStatus, cntSearch, cntExpand, maxdeep, timeuse;
//文件中读取的信息，初始状态、目标状态、搜索节点数、拓展节点数、最大深度、运行时间
```

3. 解序列打印函数

```
private void printAnsText()
```

4. 界面初始化函数，打印初始状态、目标状态，并清空其他位置

```
private void printInit()
```

5. 打印下一步函数

```
private void nextStep()
```

6. 下一步按钮相应函数

```
private void Nxt_Click(object sender, EventArgs e)
```

7. 重置按钮相应函数

```
private void Reload_Click(object sender, EventArgs e)
```


Copyright By Tongji University

8. 进入搜索树展示界面按钮响应函数

```
private void Tree_Click(object sender, EventArgs e)
```

9. 求逆序对数量判断是否有解函数

```
private bool checkNoans()
```

10. 输入数字检查函数，包括输入数字长度，内容

```
private bool checkNum()
```

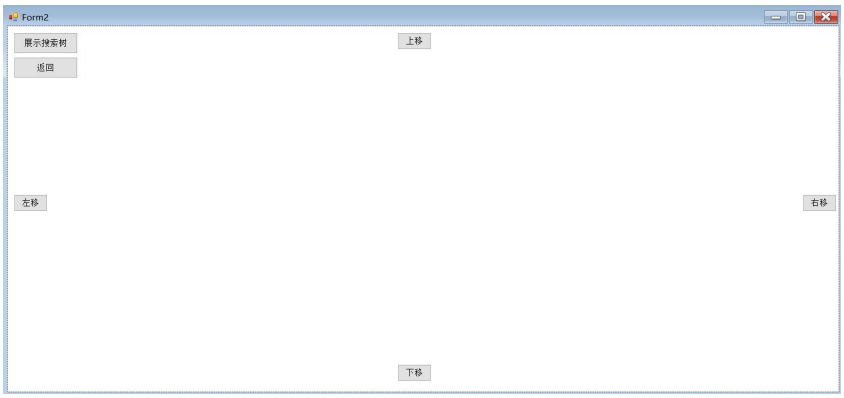
11. 文件读取函数

```
private void readin()
```

3.2.3 Form2.cs

用于展示搜索树图形界面

1. 界面展示



2. 数据说明

```
const int width = 200;       //搜索树和右侧距离
const int height = 60;       //每层高度
const int radious = 20;      //每个节点宽度
const int gap = 20;          //叶子结点宽度

int Pfrom, Pto, Hfrom, Hto; //用于读入文件数据
int Px = 0, Py = 0;
int Hmax = 0;   //最大深度
int[] num = new int[100000];     //各状态子节点数量
int[] vH = new int[100000];      //各节点启发函数值
int[] mat = new int[100000];     //各节点原值（9 位整形）
string[] str = new string[100000];  //各节点字符串值（9 位 string）
int[,] edge = new int[100000, 4];   //各节点子节点离散化值
int[] Wmy = new int[100000];        //各节点宽度
int[,] Wson = new int[100000, 4];   //各节点子节点宽度
int mpfrom, mpto;
static Dictionary<int, int> mp = new Dictionary<int, int>();
//mp 用于离散化，将 9 位整形离散化为较小值
static Dictionary<int, bool> mp2 = new Dictionary<int, bool>();
```


Copyright By Tongji University

```
        //mp2 记录各节点是否为解序列上节点
        int cnt = 1;     //用于 9 位整形的离散化
```

3. 搜索树展示按钮响应函数

```
private void Show_Click(object sender, EventArgs e)
```

4. 返回按钮响应函数

```
private void Return_Click(object sender, EventArgs e)
```

5. 上下左右按钮响应函数

```
private void Left_Click(object sender, EventArgs e)
private void UP_Click(object sender, EventArgs e)
private void Right_Click(object sender, EventArgs e)
private void Down_Click(object sender, EventArgs e)
```

6. 文件读取函数

```
private void readin()
```

7. 递归计算节点宽度，now 表示当前节点，返回值为当前节点宽度

```
private int calW(int now)
```

8. 打印 3*3 数字矩阵，g 画布，now 当前节点，h 当前高度，w 当前宽度

```
private void print(Graphics g, int now, int h, int w)
```

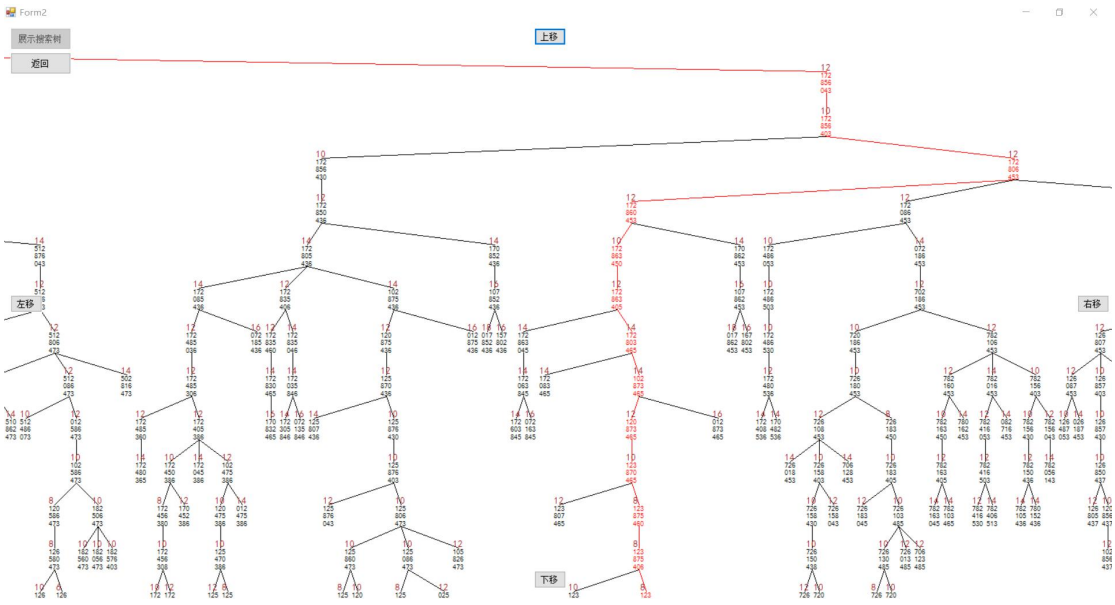
9. 递归打印搜索树函数，g 画布，now 当前节点，hnow 当前高度，wbase 当前节点左侧宽度，midfa 父节点位置 x 坐标

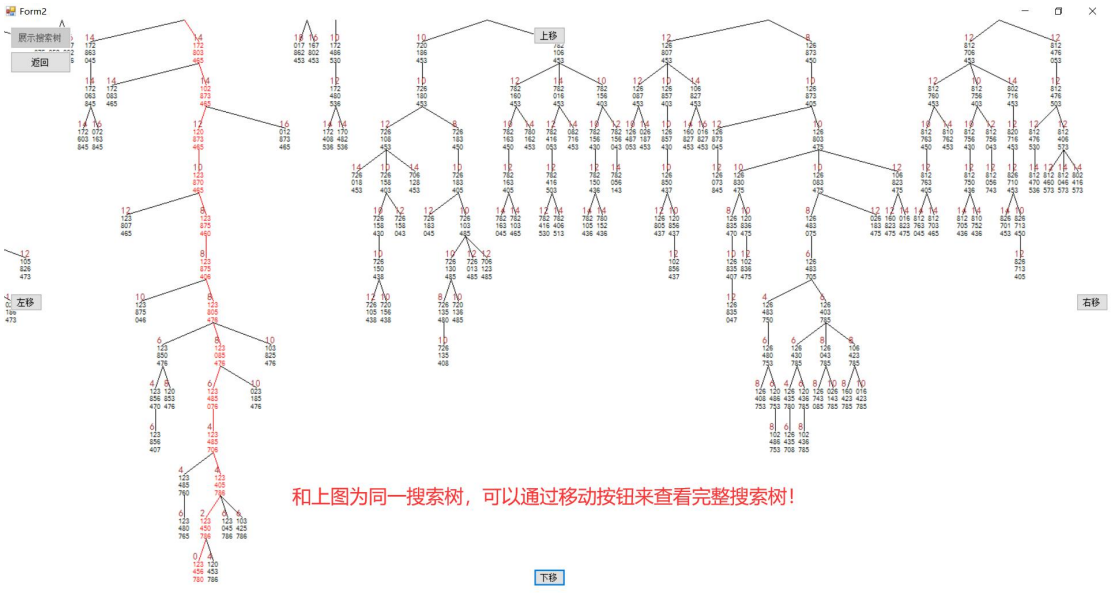
```
private void draw(Graphics g, int now, int hnow, int wbase, int midfa)
```


Copyright By Tongji University

3.3.实验结果展示

Copyright By Tongji University





和上图为同一搜索树，可以通过移动按钮来查看完整搜索树！

Copyright By Tongji University

3.4.实验结论

3.4.1 不同启发函数性能展示

	h1(n)(各数字与目标状态位置绝对距离)			h2(n)(各位置与目标状态数字不同数量)		
测试数据	搜索节点数量	扩展节点数量	程序运行时间	搜索节点数量	扩展节点数量	程序运行时间
263841750	3659	5835	46-58ms	19872	29167	260-278ms
865713420	7937	12247	108-124ms	68884	90227	960-1024ms
384651720	4245	6591	56-64ms	36171	50888	480-510ms

Copyright By Tongji University

3.4.2 不同启发函数性能分析

由前一小题表格可以得知，不同启发函数的效率有巨大差异，对于同一个问题，差别能达到 5-10 倍，选择了不好的启发函数，对于绝大多数问题，效率都会变差。

就我选择的两个启发式函数来说，“各数字与目标位置绝对距离”这一启发式函数的效率较好，一般的问题搜索节点在 500-10000，扩展节点在 1000-15000，时间在 0.01-0.08s；而对于“各位置与目标状态数字不同数量”这一启发式函数的效率较好，一般的问题搜索节点在 10000-100000，扩展节点在 15000-200000，时间在 0.05-0.4s。

因此可以看出，首先构造出一个好的启发函数会对整个程序的性能带来巨大的提升。

Copyright By Tongji University

4. 总结

4.1.实验中存在的问题及解决方案

在实验过程中，我同时希望使用 c++与 c#语言，是因为 c++有较多的内置函数，特别是 STL 容器，能够轻松实现优先队列以及红黑树，对于数据的处理与计算有很大的便利；而 c#的模块化功能能够实现前端界面的展示，在展示数据方面比较有优势。

但是如何使得两个不同程序语言模块交互呢？在这一点我考虑了比较久，最终选择将 c++程序编译为 dll 文件，并且通过命令加载到 c#程序中，在 c++程序计算结果，并将结果输出到指定文件，最后在 c#程序中读取结果并展示。

4.2.心得体会

首先，通过八数码问题，对 A*算法有了更深入的理解，特别是对启发式函数，通过比较，更直观的感受到了不同的启发式函数对于程序效率的影响，也初步掌握了一些启发式函数的构造方法。

其次，在图形化界面的展示上，我选择了使用 c#语言，直接使用其内置的一些模块方法，能非常简便的实现图形展示功能，让计算出的结果更直观的呈现出来；而经过了这个项目，我也初步掌握了简单的 c#程序的编写，包括文件读取，用例颜色、文字的改变，数据存储与处理等等。

4.3.后续改进方向

1. 增加 4*4 的游戏模式，由于 3*3 与 4*4 模式基本相同，因此可以在程序修改一些写死的数字就可以完成任务；同时需要在前端也增加一些自适应程序，以实现多种游戏模式的切换。

2. 将 c++代码改写为 c#，自己在 c#实现 STL 容器，这样计算程序得到的结果可以直接使用，而不需要通过文件传递。

Copyright By Tongji University

5. 成员分工与自评

整个项目由我个人独立完成，包括 c++程序、c#程序，均为独立完成。其中，在程序编写方面没有借鉴与参考网上的任何相关资料，完全自主独立完成。

总体的任务有：

1、启发式函数的构造

2、c++程序，用于具体问题的答案求解与计算，并输出完整过程

3、c#程序，用于与 c++程序求得的答案交互，将整个实现过程可视化

4、c++程序编译为 dll 并加载到 c#程序

5、测试数据的构造与程序的测试，包括具体的性能分析

6、利用 c#程序对搜索树的绘制

7、实验报告的撰写

Copyright By Tongji University