同济大学人工智能原理课程实验报告

实验题目：**8 数码状态表示法**

姓名:**张天然**　　学号:1751237　　专业:**信息安全**
姓名:**李子琦**　　学号:1752249　　专业:**信息安全**

一．实验概述

【实验目的】

通过实验加深对搜索策略的理解，尤其是对启发式搜索的基本原理的理解，使学生能够通过编程实现图搜索的基本方法和启发式搜索算法，并能够解决一些应用问题。

【实验问题描述】

在 3×3 组成的九宫格棋盘上，摆有八个将牌，每一个将牌都刻有 1-8 八个数码中的某一个数码。棋盘中留有一个空格，允许其周围的某一个将牌向空格移动，这样通过移动将牌就可以不断改变将牌的布局。这种游戏求解的问题是：给定一种初始的将牌布局或结构（称初始状态）和一个目标的布局（称目标状态），问如何移动将牌，实现从初始状态到目标状态的转变。

初始状态：

8 个数字将牌和空格在九宫格棋盘上的所有格局组成了问题的状态空间。其中，状态空间中的任一种状态都可以作为初始状态。

后继函数：

通过移动空格（上、下、左、右）和周围的任一棋子一次，到达新的合法状态。

目标测试：

比较当前状态和目标状态的格局是否一致。

路径消耗：

每一步的耗散值为 1，因此整个路径的耗散值是从起始状态到目标状态的棋子移动的总步数。

【实验原理】

1） 存在性：当始末状态的逆序数奇偶性相同时存在，反之不存在；

2） 主要原理：f=g+h。

g 表示从初始节点 s 到当前节点 n 的最短路径的耗散值；h 表示从当前节点 n 到目标节点 g 的最短路径的实际耗散值，f 表示从初始节点 s 经过 n 到目标节点 g 的最短路径的耗散值。

【实验环境】

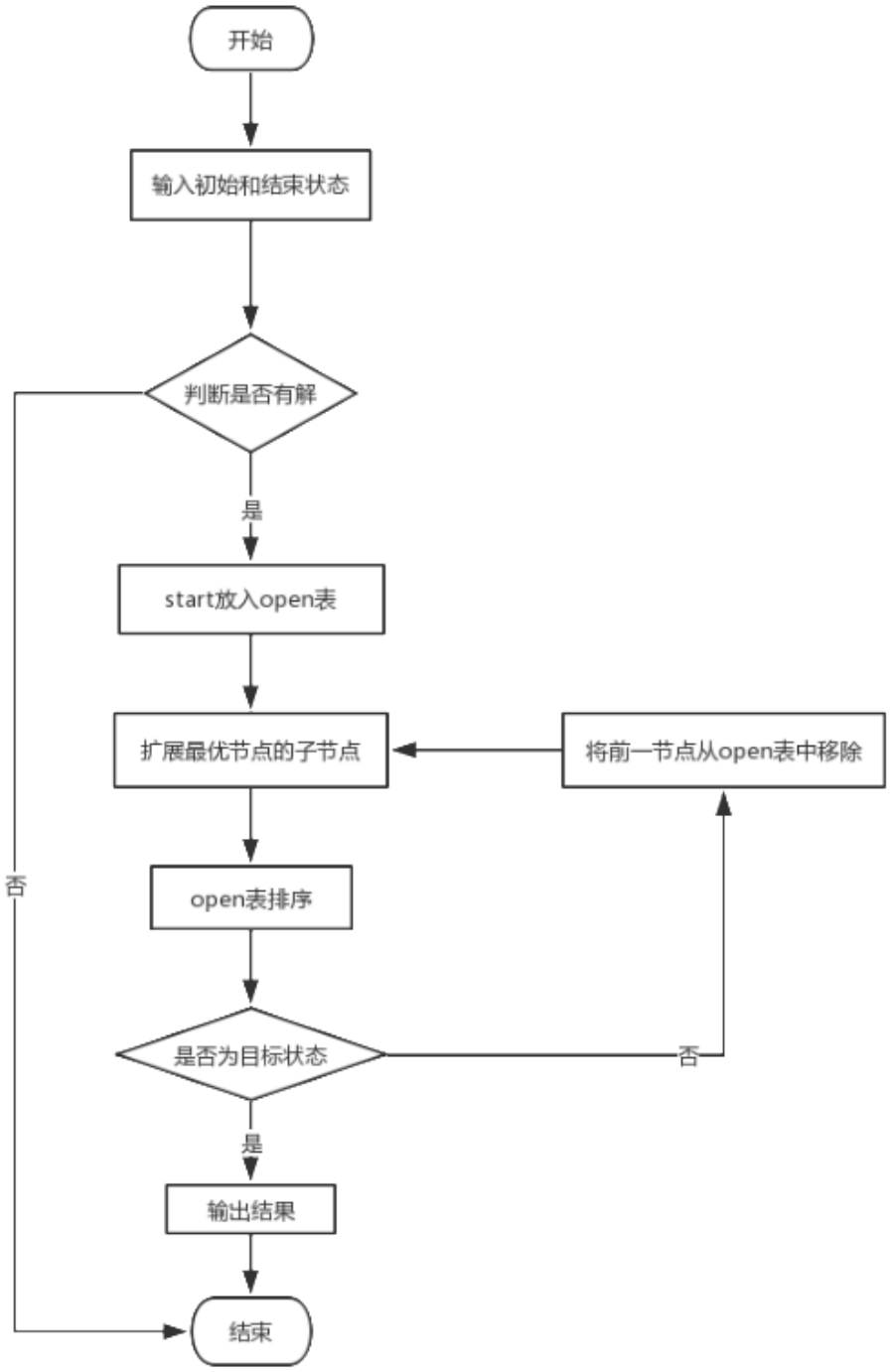
在 Windows10 中，以 visual studio 2019 为开发环境。

为方便观察结果，从 input.txt 文件输入，输出存放在 output.txt 中。

二．实验过程及结果

【实验方案设计】

设计流程图如下：



【源程序】

```
#include<iostream>
#include<stdio.h>
#include<cmath>
#include<fstream>
using namespace std;
ifstream infile("input.txt", ios::in);
ofstream outfile("output.txt", ios::out);

int open_cnt = 0;
int open_node_cnt;//open表节点个数

struct Node {
    int a[3][3];
    int x, y;
    int f, g, h;//f=g+h
    int flag; //上一次移动方向
    Node* father;
}start, End;

struct Open_Close {
    int f;
    Node* np;
}open[10000], close[10000];

bool isable() {//判断是否有解:逆序数之和奇偶性相同，有解
    int s[9], e[9];
    int tf = 0, ef = 0;
    int k = 0;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            s[k] = start.a[i][j];
            e[k] = End.a[i][j];
            k++;
        }
    }
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < i; j++) {
            if (s[i] > s[j] && s[j] != 0) tf += 1;
            if (e[i] > e[j] && e[j] != 0) ef += 1;
        }
    }
    if ((tf % 2 == 1 && ef % 2 == 1) || (tf % 2 == 0 && ef % 2 == 0)) return true;
    else return false;
```

```
}

int a_start_h(Node * node) {  //求 h
    int old_x = 0, old_y = 0, End_x = 0, End_y = 0;
    int h = 0;
    for (int k = 1; k < 9; k++) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (node->a[i][j] == k) {  //相应开始点的下标
                    old_x = i;
                    old_y = j;
                }
                if (End.a[i][j] == k) {   //相应目标的结点下标
                    End_x = i;
                    End_y = j;
                }
            }
        }
        h += abs(old_x - End_x) + abs(old_y - End_y); //计算h
    }
    return h;
}

void input() {               //输入
    //cout << "=====输入起始图=====" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            infile >> start.a[i][j];
            if (start.a[i][j] == 0) {
                start.x = i;
                start.y = j;
            }
        }
    }
    outfile<< endl;
    //cout << "=====输入目标图=====" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            infile >> End.a[i][j];
            if (End.a[i][j] == 0) {
                End.x = i;
                End.y = j;
            }
        }
```

```
    }
    outfile << endl;
    start.g = 0;
    start.h = a_start_h(&start);
    start.f = start.g + start.h;
}

int show(Node * node) {                //显示
    Node* p = node;
    if (p == &start) return 1;
    else show(p->father);
    outfile << "==============\n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            outfile << p->a[i][j] << " ";
        }
        outfile << endl;
    }
    outfile << "==============\n\n";
}


bool isend(Node * node) {          //判断是否为目标节点
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (node->a[i][j] != End.a[i][j])
                return false;
        }
    }
    return true;
}


void sort(Open_Close * open) {       //open表排序
    int min = 99999, min_flag = 0;
    Open_Close temp;
    for (int i = 0; i <= open_cnt; i++) {
        if (min > open[i].f && open[i].f > 0) {
            min = open[i].f;
            min_flag = i;
        }
    }
    temp = open[min_flag];
    open[min_flag] = open[0];
```

```
    open[0] = temp;
}



void move(int flag, Node * node) {   //向四个方向扩展
    int temp;
    if (flag == 1 && node->x > 0) {     //turn left
        Node* n = new Node();
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                n->a[i][j] = node->a[i][j];
            }
        }
        n->a[node->x][node->y] = node->a[node->x - 1][node->y];
        n->a[node->x - 1][node->y] = 0;
        n->x = node->x - 1;
        n->y = node->y;
        n->flag = 3;
        n->father = node;
        n->g = node->g + 1;              //  求 g（）
        n->h = a_start_h(n);
        n->f = n->g + n->h;  //  求 f（）
        open_cnt++;
        open_node_cnt++;
        open[open_cnt].np = n;         //添加到open表
        open[open_cnt].f = n->f;
    }

    else if (flag == 2 && node->y < 2) {     //go up
        Node* n = new Node();
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                n->a[i][j] = node->a[i][j];
            }
        }
        n->a[node->x][node->y] = node->a[node->x][node->y + 1];
        n->a[node->x][node->y + 1] = 0;
        n->x = node->x;
        n->y = node->y + 1;
        n->flag = 4;
        n->father = node;
        n->g = node->g + 1;              //  求 g（）
        n->h = a_start_h(n);
```

```
        n->f = n->g + n->h;             //  求 f ()
        open_cnt++;
        open_node_cnt++;
        open[open_cnt].np = n;          //添加到open表
        open[open_cnt].f = n->f;
    }
    else if (flag == 3 && node->x < 2) {    //turn right
        Node* n = new Node();
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                n->a[i][j] = node->a[i][j];
            }
        }
        n->a[node->x][node->y] = node->a[node->x + 1][node->y];
        n->a[node->x + 1][node->y] = 0;
        n->x = node->x + 1;
        n->y = node->y;
        n->flag = 1;
        n->father = node;
        n->g = node->g + 1;              //  求 g ()
        n->h = a_start_h(n);
        n->f = n->g + n->h;//  求 f ()
        open_cnt++;
        open_node_cnt++;
        open[open_cnt].np = n;          //添加到open表
        open[open_cnt].f = n->f;
    }
    else if (flag == 4 && node->y > 0) {    //go down
        Node* n = new Node();
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                n->a[i][j] = node->a[i][j];
            }
        }
        n->a[node->x][node->y] = node->a[node->x][node->y - 1];
        n->a[node->x][node->y - 1] = 0;
        n->x = node->x;
        n->y = node->y - 1;
        n->flag = 2;
        n->father = node;
        n->g = node->g + 1;              //  求 g ()
        n->h = a_start_h(n);
        n->f = n->g + n->h;//  求 f ()
        open_cnt++;
```

```
            open_node_cnt++;
            open[open_cnt].np = n;          //添加到open表
            open[open_cnt].f = n->f;
        }
    }
    void expand(Node * node) {    //节点扩展
        for (int i = 1; i < 5; i++) {
            if (i != node->flag) move(i, node);
        }
    }


    int main() {
        input();
        infile.close();
        open[0].np = &start;//start放入open表
        open_node_cnt = 1;
        if (isable()) {
            while (true) {//open表不为空
                if (isend(open[0].np)) {
                    outfile << "\n路径: \n";
                    show(open[0].np);
                    outfile << open[0].np->g << endl;
                    break;
                }
                expand(open[0].np);//扩展最优节点的子节点
                open[0].np = NULL;
                open[0].f = -1;
                open_node_cnt--;
                sort(open);   //open表排序
            }
        }
        else outfile << "无解" << endl;
        outfile.close();
        system("pause");
        return(0);
    }
```

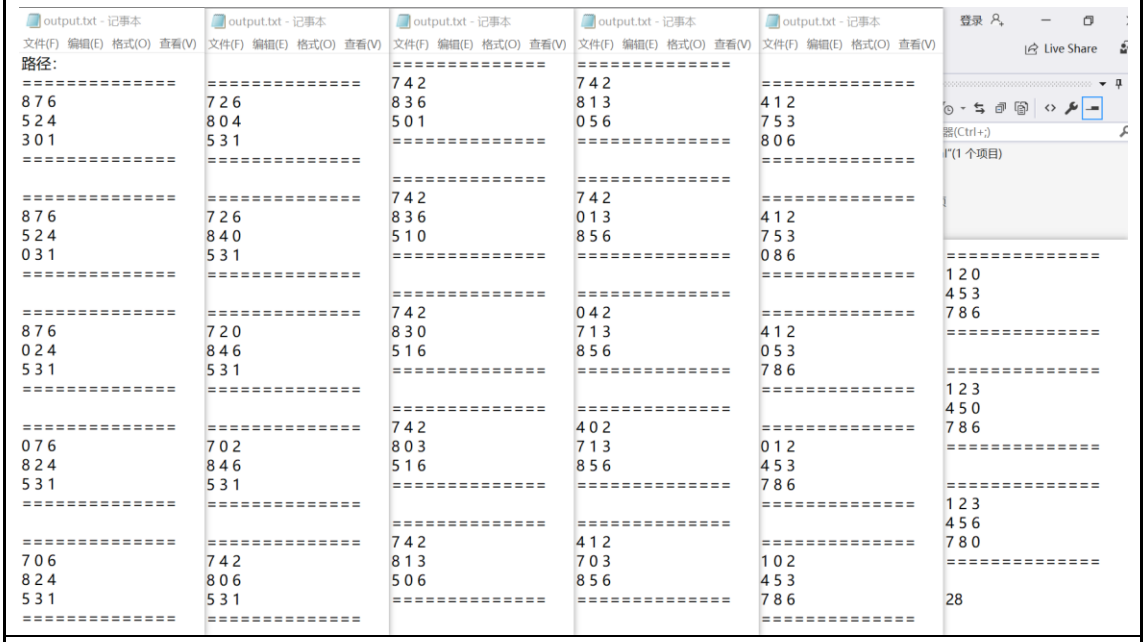
【实验结果及结论】

input.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看

```
8 7 6
5 0 4
3 2 1


1 2 3
4 5 6
7 8 0
```



三．实验汇报

【ppt 展示】

汇报展示所用 ppt 是否条理清晰，内容是否充实、完整等

【汇报答辩表现】

是否在规定时间内完成自己的陈述，汇报时口语表达清楚流畅，回答问题全面到位

四．参考文献
1） 路径规划之 A*算法：https://zhuanlan.zhihu.com/p/54510444 2） 习题课：八数码问题（上）: https://zhuanlan.zhihu.com/p/26920320 3） A* Pathfinding for Beginners:https://www.gamedev.net/articles/programming/artificial-intelligence/a-pathfinding-for-beginners-r2003/
三．小结
在构建出实验的流程图之后，本次程序实现的难点在于如何正确地实现节点的扩展。这一点上很长时间都没有进展，在参考了一些资料之后，采取了构建 move(int, Node*)的方式，考虑四种移动方向，同时计算 f 值。
成绩

附注：

说明

1．实验原理：简要说明本实验所涉及的理论知识。
2．实验环境：实验用的软硬件环境(配置)。
3．实验方案设计(思路、步骤和方法等)：概括整个实验过程。
　　对于操作型实验，要写明依据何种原理、操作方法进行实验，要写明需要经过哪几个步骤来实现其操作。
　　对于设计型和综合型实验，在上述内容基础上画出流程图、设计思路和设计方法，并配以相应的文字说明。
　　对于创新型实验，注明其创新点、特色。
4．实验结果及结论：包括实验过程中的数据、截图等。并根据实验过程中的现象，做出结论。
5．小结：实验过程中遇到的问题，对本次实验的心得体会、思考和改进建议等。
6．每一部分的空间根据需要自行调整。