

人工智能基础

编程作业 1

<http://staff.ustc.edu.cn/~linlixu/ai2018spring/>

完成截止时间：2018/5/12

提交至：ustc_ai2018@163.com

助教：盛鑫 [xins@mail.ustc.edu.cn]

申书恒 [vaip@mail.ustc.edu.cn]

赵若宇 [zry1997@mail.ustc.edu.cn]

梁润秋 [815556875@qq.com]

P1: 22 数码问题(50%)

本问题包括一个 5×5 的表格，22 个写有数字(1-22)的棋子以及一个空位（由 0 表示），两个**半透明障碍位**（涂黑的部分，可以用-1 表示）。与空位上、下、左、右相邻的棋子可以移动到空位中。

半透明障碍位为：上下不可穿透，左右可穿透，如下图中，图 1 为合法移动，图 2 为不合法移动。游戏的目的是要达到一个特定的目标状态。

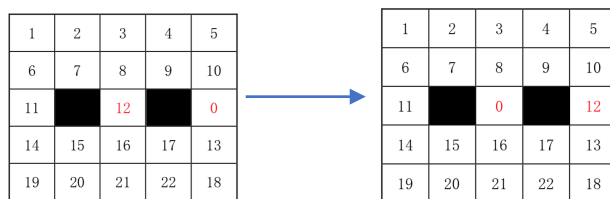


图 1：合法移动(半透明障碍物可左右穿透)

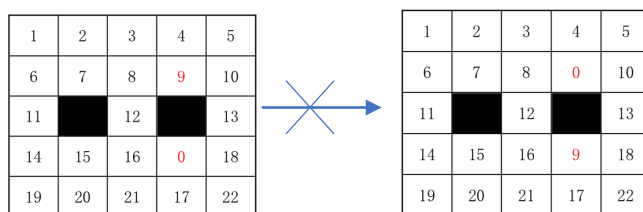


图 2：不合法移动(半透明障碍物不可上下穿透)

问题表示：

本次作业中，状态由一个 2 维矩阵表示，0 表示空位置，1-22 表示棋子，-1 表示半透明障碍物，本次作业半透明障碍物的位置固定(如图所示)，初始状态和目标状态都通过文件输入。如图所示，为一个例子。

1	2	3	4	5
6	7	8	9	10
11		12		13
14	15	16	17	18
19	20	21	22	0

图 3：22 数码问题的目标状态

1	2	3	4	5
6	7	8	9	10
0		11		13
14	15	12	17	18
19	20	16	21	22

图 4：22 数码问题的一个初始状态

定义空格棋子 0 的 4 个动作，U 代表 up，即对棋子上移，D 代表 down，即对棋子下移，L 代表 left，即对棋子左移，R 代表 right，即对棋子右移。所有动作均要合法。

本作业中，需要读取初始状态及目标状态，并实现两个求解八数码问题的算法：A*搜索及迭代深入 A*搜索(IDAS)，使用以下两种启发函数：

- $h1(n)$ = number of misplaced tiles (错位的棋子数)
- $h2(n)$ = ~~total Manhattan distance (所有棋子到其目标位置的两个方向曼哈顿距离和)~~ (不可采纳)
- $h2(n)$: 自己设计一种比 $h1$ 效果更好的可采纳的启发函数 (比如，欧几里得距离？或者曼哈顿距离的变种？)，并在实验报告中证明其可采纳性。(虽然曼哈顿距离不可采纳，但如果能证明出来其最优的话，也可以)

最后输出从初始状态到目标状态的动作序列，例如图 4 的初始状态到达图 3 的目标状态的移动序列是 **RDDRR**。

迭代 A*搜索算法的提出是为了解决 A*搜索在空间复杂度上的缺点，将迭代深入的思想用在启发式搜索上。IDA*和典型的迭代深入算法最主要的区别就是所用的截断值是 f 耗散值 ($g+h$) 而不是搜索深度；每次迭代，截断值是超过上一次迭代阶段值的节点中最小的 f 耗散值。以下为迭代 A*搜索算法。

Algorithm 3 Iterative deepening A* search (IDA*)

```
1:  $\hat{d\_limit} \leftarrow \hat{d}(s_0)$ 
2: while  $\hat{d\_limit} < \infty$  do
3:    $next\_d\_limit \leftarrow \infty$ 
4:    $list \leftarrow \{s_0\}$ 
5:   while list is not empty do
6:      $s \leftarrow head(list)$ 
7:      $list \leftarrow rest(list)$ 
8:     if  $\hat{d}(s) > \hat{d\_limit}$  then
9:        $next\_d\_limit \leftarrow \min(next\_d\_limit, \hat{d}(s))$ 
10:    else
11:      if  $s$  is a goal then
12:        return  $s$ 
13:      end if
14:       $newstates \leftarrow \text{apply actions to } s$ 
15:       $list \leftarrow \text{prepend}(newstates, list)$ 
16:    end if
17:  end while
18:   $\hat{d\_limit} \leftarrow next\_d\_limit$ 
19: end while
20: return fail
```

作业要求:

1. 统一从文件输入，文件名为 **input.txt**, 目标状态文件名为 **target.txt**，注意检查初始状态与目标状态是否匹配，输出到文件，所有文件和可执行文件在同一个文件夹。
2. 输入格式为 5×5 矩阵格式，例如，对于图 3 的示例初始状态文件和目标状态文件内容如下：

1	2	3	4	5
6	7	8	9	10
0	-1	11	-1	13
14	15	12	17	18
19	20	16	21	22

初始状态(input.txt)

1	2	3	4	5
6	7	8	9	10
11	-1	12	-1	13
14	15	16	17	18
19	20	21	22	0

目标状态(target.txt)

以上左边的矩阵为初始状态，右边的矩阵为目标状态。其中(3,2)位置和(3,4)位置为-1，作为**半透明障碍物**。其余位置为 0-24 个数字是可移动的元素，数字之间用空格隔开，每行行末回车。为方便测试，请严格按照上述输入格式。我们会生成许多测试用例（并不是所有的状态都能回到目标状态，大家测试的时候尽量手动生成一个可行的状态以免搜索不到解，也可以用我们产生的初始状态作为测试，见 **22 数码问题初始状态.doc**，我们给的**不一定是最佳的步数**），**注意检查输入的初始状态和目标状态障碍位置是否一致。**

3. 输出为所花费的时间(以 s 为单位)、动作序列以及总步数，输出到文件，例如

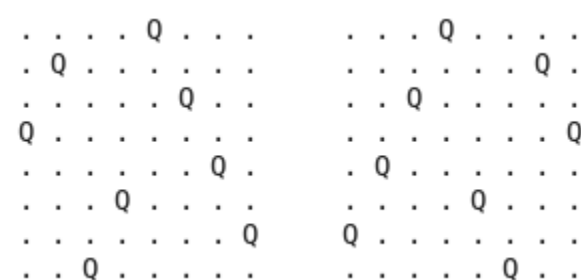
```
0.000228
RDDRR
5
```

字母大写，字母之间无空格。U 代表 up，即对棋子上移，D 代表 down，即对棋子下移，L 代表 left，即对棋子左移，R 代表 right，即对棋子右移。输出的动作序列应为从初始状态开始，到目标状态结束时，中间经过的所有的空格棋子操作动作。在测试时，我们会从输入状态开始，执行你的输出动作序列，看你的动作序列是否合法，通过此动作序列能否到达目标状态等。

4. 使用 C/C++实现 4 个算法，即，使用启发函数 $h_1(n)$ 的 A*算法: Ah1.cpp，使用启发函数 $h_2(n)$ 的 A*算法:Ah2.cpp，使用启发函数 $h_1(n)$ 的 IDA*算法:IDAh1.cpp，使用启发函数 $h_2(n)$ 的 IDA*算法:IDAh2.cpp。输出文件名与算法对应。以 h_1 为启发函数的 A*算法输出到 output_Ah1.txt；以 h_2 为启发函数的 A*算法输出到 output_Ah2.txt；以 h_1 为启发函数的 IDA*算法输出到 output_IDAh1.txt；以 h_2 为启发函数的 IDA*算法输出到 output_IDAh2.txt。
5. 提交源代码和可执行文件（4 个算法所以有 4 个代码和可执行文件），若使用命令行编译请给出编译命令。在实验报告(word 或者 pdf)中写明如何运行你的程序以及对每个程序的说明。并大致说明你算法（A*和迭代 A*）的时间复杂度和空间复杂度。使用表格列出我们所提供的样例的结果，包括步数和运行时间(如果对于某些样例，你的代码无法得出结果，标记为×)，我们对时间或者空间优化相应加分。
6. 严禁抄袭，我们会用软件进行代码查重，4 个算法都要求实现，我们会查看源代码，严禁只实现一个算法，其余 3 个用该算法代替，虽然最终都能测试通过。一旦发现上述情况，以 0 分计。

P2: “N 皇后” 问题（50%）

原始“8 皇后”问题: 在 8×8 格的国际象棋上摆放八个皇后，使其不能相互攻击，任意两个皇后都不能处于同一行、同一列或同一斜线上。



上图是两种合理的摆法，点表示没有摆放皇后的位置，“Q”表示摆放皇后的位置。

8 皇后问题可以拓展成为“N 皇后”的问题：N*N 的棋盘上摆放 N 个皇后，使其不能互相攻击，任意两个皇后不能处于同一行、同一列或同一斜线上。

在本实验中，我们对一般拓展过的“N 皇后”问题做一些修改，要求棋盘中会出现 M 对皇后互相攻击，其余的皇后之间依然要求不能互相攻击，即除这 M 对皇后互相攻击外，任意两个皇后都不能处于同一行、同一列或同一斜线上。如下图所示为 N=8, M=2 时的一种合理的摆法。

```
. Q . . . . .
. . . . Q . .
. . . . Q . .
Q . . . . .
. . . . Q .
. . . Q . . .
. . . . . Q
. . Q . . . .
```

这个问题的难点在于，时间复杂度随着问题规模是指数增长的，高效解决问题是本次实验的重点。同时相比“N 皇后”问题，这个问题的难点在于，要求存在 M 对皇后间互相攻击。

本次实验要求：棋盘中存在 M 对皇后会互相攻击，同时，其余的皇后之间依然按“N 皇后”问题所描述的要求不能互相攻击。

问题描述：

有一个 N*N 的棋盘，棋盘中第 i 行（i 从 0 开始）第 j 列（j 从 0 开始）的位置，记为(i, j)，其中 i、j 分别为行坐标和列坐标。

现有 N 个皇后，请按照以下规则摆放皇后：

- 1) 存在且仅存在 M 对皇后互相攻击，即这 M 对皇后中每对的行坐标相同，或列坐标相同，或分布在同一斜线上；
- 2) 对于除 1) 中的互相攻击关系外，棋盘上不能存在其他的互相攻击的关系。

作业要求：

从四个算法中选两个算法实现：爬山算法、遗传算法、模拟退火算法以及 CSP 问题的局部搜索算法。如果你可以找到其他更好的基于搜索的算法并实现，可以考虑适当加分。

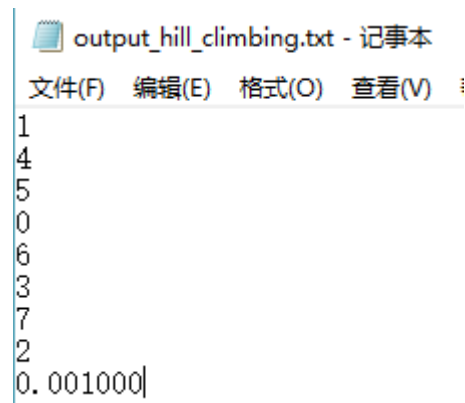
采用 C/C++ 编写实现 N 皇后问题的算法，返回一个解满足以上要求。(50%)

输入输出都是文本（请勿使用屏幕输入输出）。

输入文件名为 input.txt，只有一行，为皇后个数 N 及冲突对数 M，中间用空格分隔，例如输入 N=8, M=2 则输入文件如下图：



输出文件名为 `output_algorithmname.txt`（例如 `output_hill_climbing.txt`, `output_genetic.txt`, `output_simulated_annealing.txt`, `output_csp.txt`），一共 $N+1$ 行，例如 $N=8$, $M=2$ 时，输出文件如图，



文本的前 1 到 N 行中，第 i 行表示的是棋盘上第 i 行上的皇后的列坐标 ($[0, N-1]$)，最终输出 N 个皇后。第 $N+1$ 行输出程序执行的时间，单位为 `ms` 毫秒，一共 $N+1$ 行，行与行之间用回车分隔。

请严格遵守以上格式，格式不对会导致扣分，其中 N 的规模不会超过 **1000000**，如果要额外实现其它的算法，请**勿使用随机算法**（比如 **Las Vegas** 算法），因为测试可能只测一次，不一定会得到好的解。

建议采用以下代码计算时间：

```
#include <time.h>

clock_t start, finish;

double totaltime;

start = clock();

... //你的程序

finish = clock();

totaltime = (double)(finish-start)/CLOCKS_PER_SEC;
```

说明文档要求：（50%）

- (a) 算法思想
- (b) 算法如何节省存储空间，分析空间复杂度
- (c) 算法如何提升效率，分析时间复杂度，速度越快给分相应提高，最好能

给出**规模-时间**图。

- (d) 实验结果说明
- (e) 文档保存为 pdf 格式

注意：

请大家独立完成，我们会严格检查（将会采用程序匹配，改变量名和移动代码结构是没用的）。**实验 P1 和实验 P2 都请用 C/C++ 实现，并且请不要使用 C++11 的高级特性，以防助教测试时编译不通过或运行错误。**

实验提交

在截止时间之前将作业提交到 **ustc_ai2018@163.com**, 邮件主题为“**学号_姓名_实验一**”。提交后会收到确认接受的邮件，以此邮件为准。

将“22 数码”和“N 皇后”所需要提交的文件分别放在“**22 数码**”和“**N 皇后**”文件夹中，这两个文件夹放在“学号_姓名_实验一”文件夹中，压缩成“**学号_姓名_实验一.zip**”。将压缩包作为邮件附件一起提交到指定邮箱。

务必按时提交实验，不接受逾期提交的实验。

实验中有任何问题请直接联系助教