人工智能基础

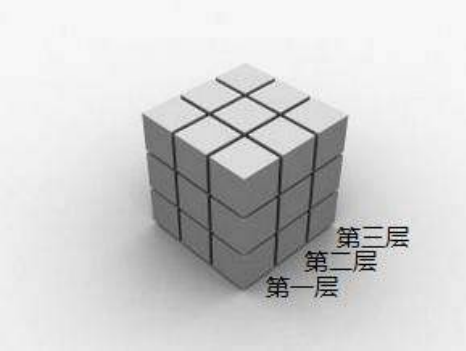
编程作业 1

http://staff.ustc.edu.cn/~linlixu/ai2016spring/

完成截止时间：2016/5/8
提交至：ustc_ai2016@163.com
助教： 郭俊良 (guojunll@mail.ustc.edu.cn)
蒋亮 (jal@mail.ustc.edu.cn)
张超 (zhangchao5656@163.com)

P1：三立方数码数码问题(50%)

本问题包括一个 3x3x3 的立方体，24 个写有数字(1-24)的单位立方体以及一个空位（由 0 表示），两个障碍位（涂黑的部分，可以用-1 表示）。与空位上、下、左、右、前、后相邻的棋子可以滑动到空位中，任何棋子都不能移动到障碍位中，且障碍位不可移动。游戏的目的是要达到一个特定的目标状态。



问题表示：

本次作业中，状态由一个 3 维矩阵表示，0 表示空位置，1-24 表示棋子，-1 表示障碍物，**本次作业障碍物的位置不固定，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。

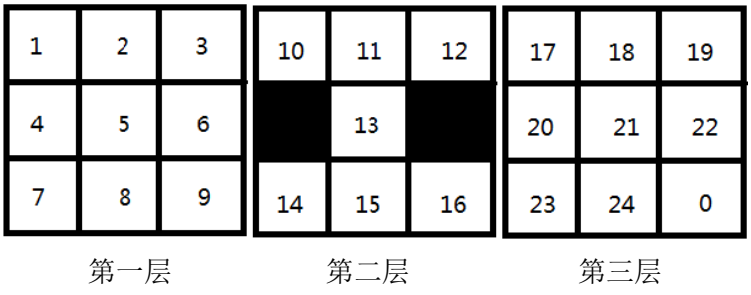


图 1：三立方数码问题的一个目标状态

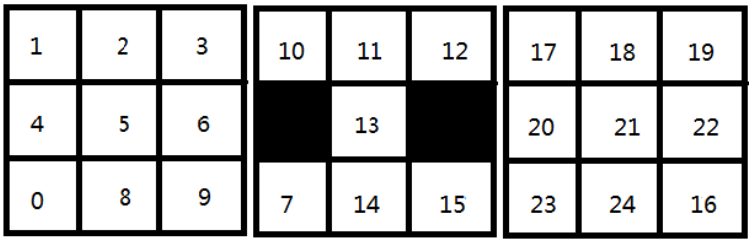


图 2：三立方数码问题的一个初始状态

定义空格棋子 0 的 6 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，F 代表 forward，即对棋子前移（靠近自己的方向），B 代表 back，即对棋子后移（远离自己的方向）。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三个方向距离和）

最后输出从初始状态到目标状态的动作序列，例如图 2 的初始状态到达图 1 的目标状态的移动序列是 BRRB。

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: d̂_limit ← d̂(s_0)
2: while d̂_limit < ∞ do
3:    next_d̂_limit ← ∞
4:    list ← {s_0}
5:    while list is not empty do
6:       s ← head(list)
7:       list ← rest(list)
8:       if d̂(s) > d̂_limit then
9:          next_d̂_limit ← min(next_d̂_limit, d̂(s))
10:      else
11:         if s is a goal then
12:            return s
13:         end if
14:         newstates ← apply actions to s
15:         list ← prepend(newstates, list)
16:      end if
17:   end while
18:   d̂_limit ← next_d̂_limit
19: end while
20: return fail
```

作业要求：

1. 统一从文件输入，初始状态文件名为 source.txt,目标状态文件名为 target.txt，输出到文件，所有文件和可执行文件在同一个文件夹。
2. 输入格式为按层排列共 3 个 3x3 的矩阵格式，例如，对于图 1-2 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
0 8 9

10 11 12
-1 13 -1
7 14 15

17 18 19
20 21 22
23 24 16
```

初始状态(source.txt)

```
1 2 3
4 5 6
7 8 9

10 11 12
-1 13 -1
14 15 16

17 18 19
20 21 22
23 24 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，每一层空一行（注意三维数组的存储方式）。其中矩阵的(2,1,2)位置和(2,3,2)位置为-1，作为障碍物，其余位置为 0-24 个数字是可移动的元素，每个数字之间一个空格，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，**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，ms 为单位，以及动作序列，输出到文件，例如

```
0
BRRB
```

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，F 代表 forward，即对棋子前移（靠近自己的方向），B 代表 back，即对棋子后移（远离自己的方向）。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。

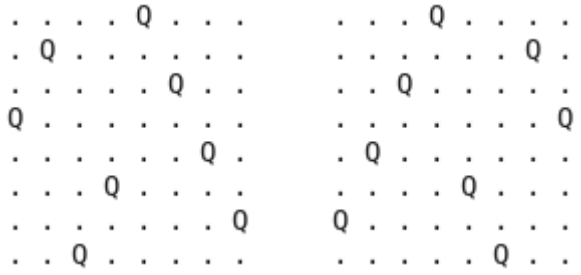
4. 使用 C/C++实现 4 个算法，即，使用启发函数 h1(n)的 A*算法：Ah1.cpp，使用启发函数 h2(n)的 A*算法:Ah2.cpp，使用启发函数 h1(n)的 IDA*算法:IDAh1.cpp，使用启发函数 h2(n)的 IDA*算法:IDAh2.cpp。输出文件名与算法对应。以 h1 为启发函数的 A*算法输出到 output_Ah1.txt；以 h2 为启发函数的 A*算法输出到 output_Ah2.txt；以 h1 为启发函数的 IDA*算法输出到 output_IDAh1.txt；以 h2 为启发函数的 IDA*算法输出到 output_IDAh2.txt。

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),用 readme 文件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。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 皇后问题:N*N 的棋盘上摆放 N 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。

这个问题的难点在于,时间复杂度随着问题规模是指数型增长的,高效解决这个问题是本作业的重点。

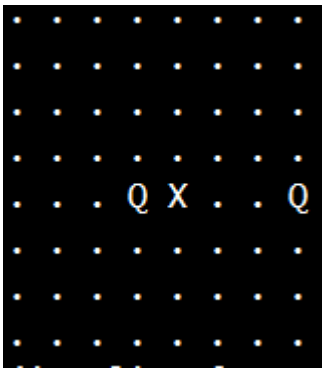
本次实验要求:棋盘中某个位置有一个障碍。分别处于障碍相反两侧,并和障碍在一条直线上的两个皇后不会相互攻击。障碍不可以放置皇后。

问题描述:

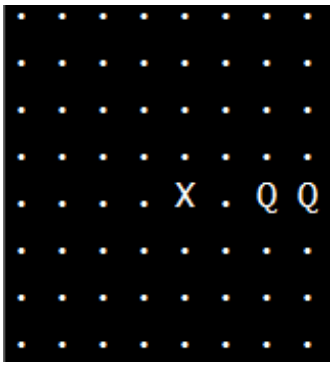
有一个 N*N 的棋盘,棋盘中第 i 行(i 从 1 开始)第 j 列(j 从 1 开始)的位置,记为(i,j),i j 分别为行坐标和列坐标。点 x(m,n)处是障碍。

现在有 N 个皇后,对任意两个皇后 p,q,需要满足

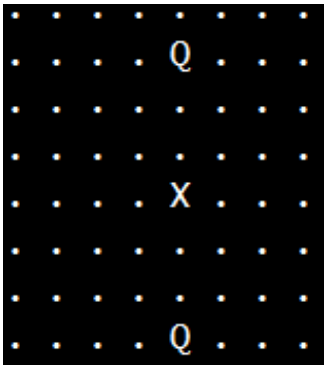
1) p,q 不在同一行上,除非 p,x,q 在同一行上,并且 p,q 分别在 x 的左右两边(图一合法,图二不合法)
2) p,q 不在同一列上,除非 p,x,q 在同一列上,并且 p,q 分别在 x 的上下两边(图三合法,图四不合法)
3) p,q 不在同一条斜线上,除非 p,x,q 在同一条斜线上,并且 p,q 分别在 x 的两边。(图五图七合法,图六图八不合法)



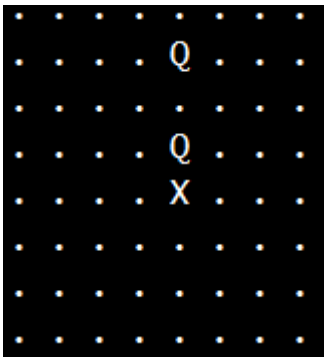
图一



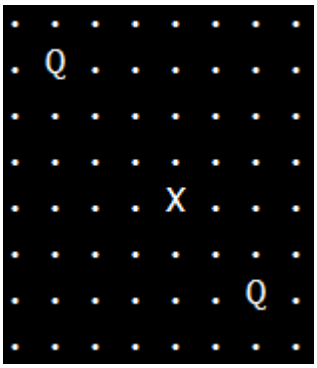
图二



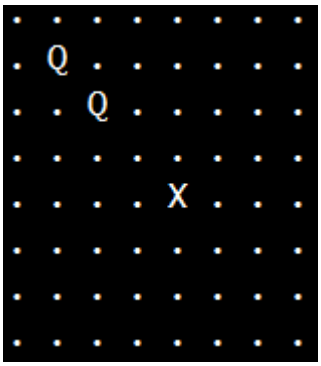
图三



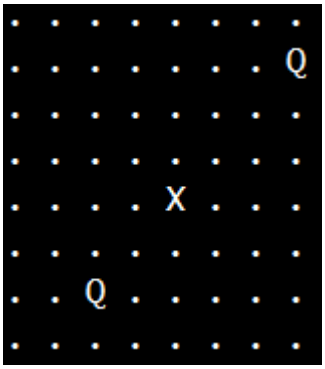
图四



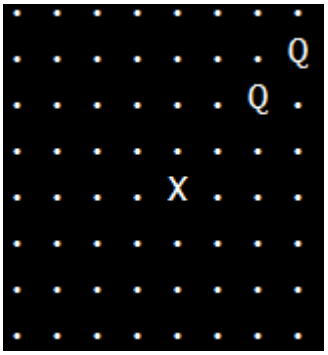
图五



图六



图七



图八

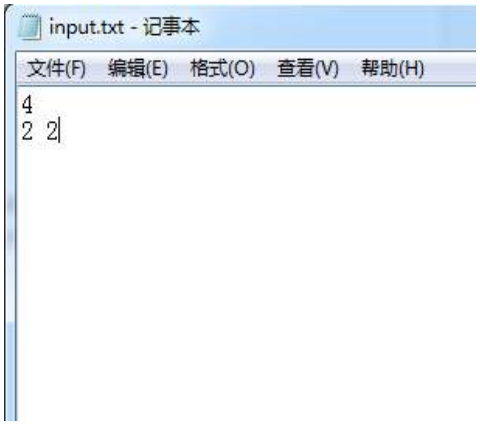
作业要求：

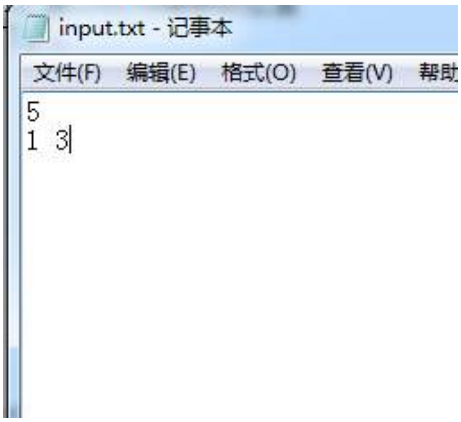
从四个算法中选两个算法实现：爬山算法、遗传算法、模拟退火算法以及 CSP 问题的局部搜索算法。如果你可以找到其他更好的基于搜索的算法并实现，可以考虑适当加分。

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%）

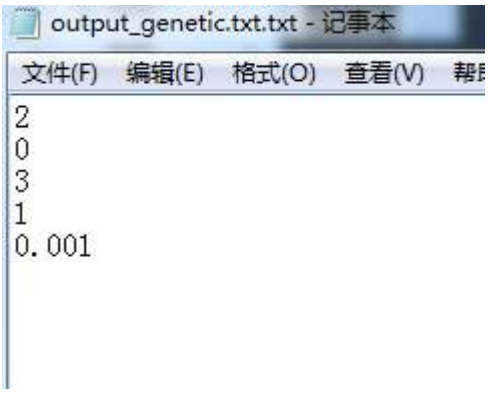
输入输出都是文本（请勿使用屏幕输入输出）。

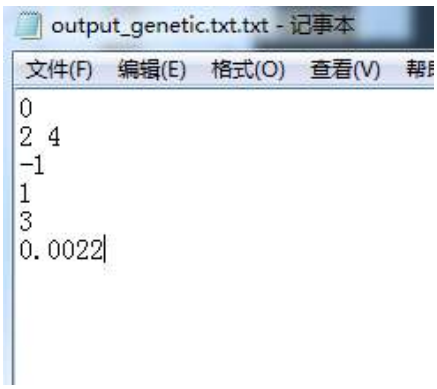
输入文件名为 input.txt，第一行为皇后的个数 N，第二行为障碍的位置(行,列)。例如输入 N=4 以及障碍(2 2)，则输入文件如左图；输入 N=5 以及障碍(1 3)，则输入文件如右图。





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=4 时，输出文件如左图，N=5 时，输出文件如右图：





文本的前 1 到 N 行中，第 i 行表示的是棋盘中第 i 行上的皇后所在的列坐标([0,n-1])，**如果第 i 行有多个皇后，请用空格隔开，如果第 i 行没有皇后，输出-1（如右图所示）**，最终输出 N 个皇后。第 N+1 行输出程序执行的时间，单位是 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) 算法效率，分析时间复杂度，速度越快给分相应提高
(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）。**实验一和实验二都请用 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_ai2016@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。

将立方数码和 n 皇后所需要提交的文件分别放在“**立方数码**”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