人工智能基础

编程作业 1

http://staff.ustc.edu.cn/~linlixu/ai2014spring/

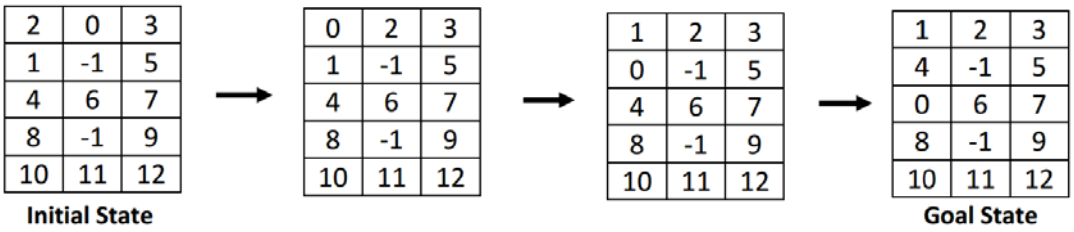
完成截止时间： 4/20/2014

助教： 仲小伟(zhxwmessi@gmail.com)
李亦锬(daniyitan@gmail.com)
王臻(wang1231991@126.com)

P1：十二数码问题(40%)

本问题包括一个 5x3 的棋盘，12 个写有数字的棋子以及一个空位(由 0 表示)，两个障碍位（由-1 表示）。与空位上、下、左、右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。

以下为一个十二数码游戏的例子：



图一. 十二数码游戏示例

问题表示

本次作业中，状态由一个矩阵表示，0 表示空位置，1-12 表示棋子，-1 表示障碍物，为简化问题，本次作业障碍物的位置固定，即在矩阵的（2,2）和（4,2）位置固定为-1。其余 0-12 数字任意放置。下图为一个例子：

2	0	3
1	-1	5
4	6	7
8	-1	9
10	11	12

定义 4 个动作， U 代表 up，即对空位 0 棋子上移，D 代表 down，即对空位 0 棋子下移，L 代表 left，即对空位 0 棋子左移，R 代表 right，即对空位 0 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水平竖直距离和）

最后输出从初始状态到目标状态的动作序列。

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。

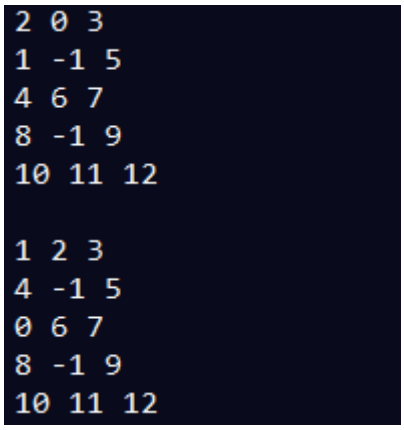
Algorithm 3 Iterative deepening A* search (IDA*)

```
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. 统一从命令行输入，输出到命令行

2. 输入格式为两个 5*3 的矩阵格式，如下图



上面一个矩阵为初始状态，中间空一行，下面的矩阵为目标状态。其中两个矩阵的（2,2）位置和（4,2）位置固定为-1，为障碍物；其余位置为 0—12 个数字任意放置，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，每个测试用例的两个矩阵的障碍物位置固定，即在（2,2）和（4,2）位置设置为-1，其余数字任意放置。

3. 输出为动作序列，例如`L D D R R U U U U L L D D R R D`，字母大写，字母之间隔一个空格。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. 实现 4 个算法，即，使用启发函数 h1(n)的 A*算法，使用启发函数 h2(n)的 A*算法，使用启发函数 h1(n)的 IDA*算法，使用启发函数 h2(n)的 IDA*算法。

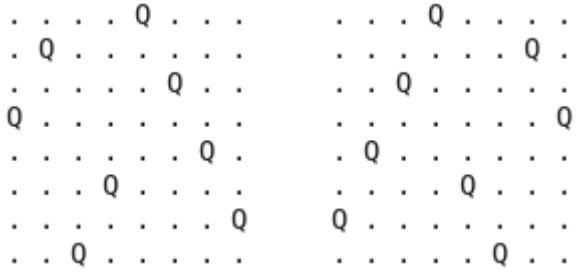
5. 比较使用不同的启发函数 h1,h2 的 A*搜索及迭代深入 A*搜索的性能，并分析性能差异的原因。

6. 在 windows 平台下，使用 c 或 c++编写（不要使用 c++ 11 的特性）。提交 4 个源文件 AStar_h1.c/cpp, AStar_h2.c/cpp, IDAStar_h1.c/cpp, IDAStar_h2.c/cpp，以及编译之后的相应的可执行文件，程序要能够编译运行，可用 readme 对你的程序进行说明（如果有必要的话）。

7. 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(60%)

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。

问题表示：

在向量 try[1…N]中存放第 1 行至第 N 行皇后摆放的列坐标。即 try[i]表示第 i 个皇后摆放在(i,try[i])位置上。

若 N 个皇后不相互攻击，则称 try[1…N]是 N-promising 的。

形式化：对$\forall i,j \in [1,N]$，若 $i \neq j$，有：

$$try[i] - try[j] \notin \{i-j, 0, j-i\} \quad (1)$$

若上式成立，则；

（1） 无行冲突，因为第 i 个皇后放在第 i 行

（2） 无列冲突，try[i]-try[j]不为 0

（3） 无 135° 对角线冲突：若第 i 行和第 j 列的皇后有此冲突，则 try[i]-try[j] = j-i。

（4） 无 45° 对角线冲突：若第 i 行和第 j 列的皇后有此冲突，则 try[i]-try[j] =i-j。

作业要求：

（1）实现一个 N 皇后问题的算法。（25%）

输入：皇后个数 N。

输出：三个满足问题要求的皇后摆放位置序列 try[1..3][1…n]。

函数格式：

```
size_t **NQueen(size_t Queen_number)
```

（2）空间复杂度：可解决的问题规模（N）越大越好。（5%）

（3）时间复杂度：在给定规模（N）下，解决速度越快越好。（30%）

采用随机算法的同学，我们将测试一定次数对结果取平均值作为最终结果。

请将你的算法如何节省存储空间，提升算法效率的思想写在说明文档内。

提交格式：

3 个文件：

<1>一个 P1 的.c/.cpp 文件，包含 as,idas 这 2 个函数相应的部分，和必要的 include,其余部分由助教自行添加（包括 main 函数，结果正确性测试）。需要添加的 include 如下（如果有其他需要的请同学们自行添加，以下 1 个是必须添加的，方便我们进行修改）：

```
#include<stdio.h>
```

<2>一个 P2 的.c/.cpp 文件，包含 NQueen 函数相应的部分，和必要的 include，其余部分由助教自行添加（包括 main 函数，时间统计，结果正确性测试）。需要添加的 include 如下：

```
#include<stdio.h>
#include<windows.h>
#include<time.h>
#include<math.h>
```

使用linux操作系统的同学，请在自己程序提交之前再添加第二行和第三行。为了保证测试平台的统一，我们会把所有程序都移植到windows下。请同学们避免使用C++的高级特性以及STL（这个应该用不到）中的数据结构和函数，以防实现不一样造成差错。

<3>一个说明文档，包括同学们解决N皇后问题的算法说明，时间、空间复杂度的分析。

请同学们把以上 3 个文件压缩成名为“学号（字母大写）_姓名”.rar/zip 等格式的压缩包。邮件主题为“人工智能第一次实验”，发送到任意一位助教的邮箱。

Caution：

请大家独立完成，我们会严格检查。