搜索算法

1.盲目搜索--迷宫（无需提交）

尝试利用DFS、BFS、深度受限算法、迭代加深算法、双向搜索算法解决迷宫问题算法. 迷宫问题的输入数据类型为仅包含"1","0","S","E"的文本数据, 其中S表示起点,E表示终点,1表示墙,0是可通行. 输出的解数据类型为 `list`, 其中list里面记录了智能体移动过程中的位置信息.

若选择DFS算法, 则函数名为 `DFS`， 若选择BFS算法, 则函数名为 `BFS`，其他算法可采用类似的函数命名.

例子: 输入文本数据为

```
11111111111111111111111111111111111
10000000000000000000000000000000S1
10111111111111111111111101111111101
10110001000100000011111100011000001
10110101010101111011111111011011111
10110101010100000000000011011000001
10110101010101011110011000011111101
10100101010001000011011111110000001
10110101011111111011000000011011111
10110100011000000011111111011000001
10000111111011111110000011011111101
11111100000010000000111101101000001
10000001111110111110100001101101111
10111111000000100000001111101100001
10000000011111101111111111101100110 1
11111111110000000000000000011111101
1E0000000001111111111111111000000001
11111111111111111111111111111111111
```

则调用 `DFS(puzzle)`后输出解(仅供参考)

```
[(1, 34), (2, 34), (3, 34), (3, 33), (3, 32), (3, 31), (3, 30), (4, 30), (5, 30),
(5, 31), (5, 32), (5, 33), (5, 34), (6, 34), (7, 34), (7, 33), (7, 32), (7, 31),
(7, 30), (8, 30), (9, 30), (9, 31), (9, 32), (9, 33), (9, 34), (10, 34), (11, 34),
(11, 33), (11, 32), (11, 31), (11, 30), (12, 30), (13, 30), (13, 31), (13, 32),
(13, 33), (13, 34), (14, 34), (15, 34), (16, 34), (16, 33), (16, 32), (16, 31),
(16, 30), (16, 29), (16, 28), (16, 27), (15, 27), (15, 26), (15, 25), (15, 24),
(15, 23), (15, 22), (15, 21), (15, 20), (15, 19), (15, 18), (15, 17), (15, 16),
(15, 15), (15, 14), (15, 13), (15, 12), (15, 11), (15, 10), (16, 10), (16, 9),
(16, 8), (16, 7), (16, 6), (16, 5), (16, 4), (16, 3), (16, 2), (16, 1)]
```

2.启发式搜索--15-Puzzle（无需提交）

利用A*算法和IDA*算法解决15-Puzzle问题, 可自定义启发式函数. Puzzle问题的输入数据类型为二维嵌套list, 空位置用 `0`表示. 输出的解数据类型为 `list`, 是移动数字方块的次序.

若选择A*算法, 则函数名为 A_star; 若选择IDA*算法, 则函数名为 IDA_star.

例子: 输入

```
puzzle = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[0,13,14,15]]
```

则调用 A_star(puzzle)或 IDA_star(puzzle)后输出解

```
[13,14,15]
```

3.博弈树搜索--中国象棋

文件介绍

- main.py是main函数的程序，直接运行这个文件可以实现人机博弈对抗。
- 其他.py文件都是程序运行所需要的类，包括ChessBoard、Game等。
- images文件夹是可视化界面所需的图片。
- 对手AI在ChessAI.py中实现，对手AI类已被pyarmor加密，需要安装pyarmor库才能运行此py文件。另外，我们提供了linux、windows、mac三个版本的加密文件，根据自己电脑的系统选择对应版本的程序代码。
- MyAI.py提供了ChessAI.py中部分代码逻辑，其中包括了Evaluate、ChessMap、ChessAI三个类。Evaluate类提供了当前象棋局面的奖励值，即每个棋子在棋盘上发任意位置都会有一个奖励值，所有棋子的奖励值之和为整个棋面的奖励值。提供的奖励值仅仅作为参考，如果想要以更大的概率打败对手AI，建议修改奖励值。ChessAI是实现算法的核心类，须在此类中实现搜索算法。
- 最终评估方法：与对手AI共博弈2次，其中先手、后手各评估一次（在main.py中未实现算法的红黑机指定代码，需自行实现）。积分规则：胜一局记3分，平一局记1分，负一句记0分。

代码运行

建议使用python3.7或python3.6运行代码

需要安装pygame、numpy、pyarmor库：

```
pip install pygame numpy pyarmor
```

开始程序的命令：

```
# 在terminal中运行：
python main.py
```

```
# 在pycharm或vscode中运行：
main.py
```

提示

- 重复走棋子：重复走子，判输
- 和棋：如果30个回合没有棋子被吃，判和