1. **迷宫问题**

请在3月31日前于学在浙大提交

**一、写在前面**

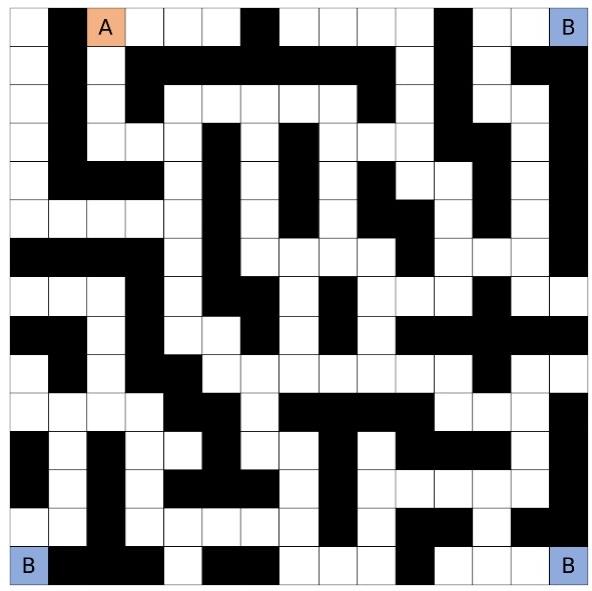
经过课堂上的学习和课后的复习相信大家对迷宫问题和一些简单的搜索方法已经有了 一定的了解。为了进一步巩固学习成果，我们将分别使用这些搜索方法(DFS、 IDDFS、

BFS、UCS 和 A\*) 解决一个更复杂的迷宫问题。

同时，本次作业也将帮助大家熟悉如何用python 语言进行编程，并锻炼大家的代码阅 读及书写能力。两个问题的报告分开写，结合代码一起打包在一个压缩包内。

**二、题目描述**

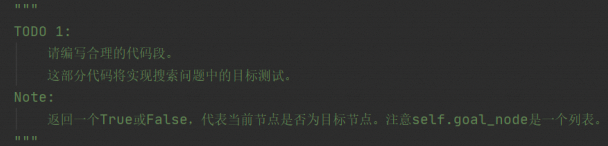
考虑如下所示的迷宫问题，起点为A 所在的橙色方格，迷宫有三个出口，分别用B 所 在的蓝色方格标明。迷宫中的黑色方格为障碍物，不可通行。由于机器人的特殊构造，其向 上下左右移动一步的代价分别为2、1、3、5。请使用搜索算法找到任——条通向迷宫出口 的路径，并计算该路径对应的代价。



**三、作业内容**

在 src 文件夹的search.py 中，给出了包含DFS、IDDFS、BFS、UCS和 A\*这 5 种 搜 索算法(图搜索)代码框架的文件，,请你将**算法核心部分补**充完整。

需补充的地方有TODO 和 NOTE 的提示(一共有9处)。 提示：建议先熟悉search.py 中 的BaseSearch 基类。



**以广度优先搜索为例，在将** **search.py** **补充完整后，打开终端，在** **src** **目录下输** **入"** **python** **main.py--search\_type** **bfs",即可看到相应的结果**。

**四、提交要求**

请提交一个以**hw1\_** **学号.zip** 命名的zip格式压缩文件，此压缩文件应当包含：

1)补充完整的search.py;

2)作业报告 report.pdf。

注意，请在你补充的代码中必要的地方写上注释。

作业报告要求：

1)附上迷宫问题中5个搜索算法的运行结果，包括路径及其代价(由于文本打印的路 径可能很长，可以用可视化图片说明搜索到的路径);

2)对每个问题中的5个结果进行简要的比较和说明，了解对应的算法性质(最优性、 复杂度等);

3)阐述你设计的AstarSearch 类中的启发式函数；

4)试着改变上下左右移动的代价(BaseSearch 基类中的 self.actions\_costs),在 不 同的问题参数设置下，5个搜索算法的运行结果有何变化?



5)报告的具体格式没有要求。

**五、成绩评定**

代码补充实现占50%,report占50% 。

1. **Tic\_Tac\_Toe问题**

请在3月24日前于学在浙大提交

**1. 任务**

本编程作业要求实现基于MiniMax Search的tic\_tac\_toe问题解决方案。

**2.程序框架**

样例程序包含两个文件： **main\_tic\_tac\_toe.py**以及**tic\_tac\_toe.py tic\_tac\_toe.py**:

 实现了GameJudge类， 用于判断当前状态的输赢情况， **这部分代码不需要修改**；

 实现了MiniMax\_Search函数的基本流程， 但是**关键步骤的子函数代码 (eg. min\_value, max\_value, utility函数) 要求自己编写**；

**main\_tic\_tac\_toe.py**：

 测试文件， 从tic\_tac\_toe.py中导入GameJudge和Minimax\_Search， 实现用户和电脑之间的博 弈过程。

完成tic\_tac\_toe.py文件之后， 可以运行main\_tic\_tac\_toe.py测试自己写的Minimax Search算法得到 的落子是否合理。一般情况下， 如果Minimax Search算法实现合理， 电脑和用户总是能下成平局。

**3. Minimax Search算法实现说明**

本作业要求实现Minimax Search算法， 并要求有限搜索深度 (depth=3)。实现过程中， 为了保证程 序细节的一致性， 我们事先做出以下约定：

1. 电脑玩家 (用1表示) 使用circle， 你 (用 1表示) 使用cross；

2. 电脑玩家是MAX user， 你是MIN user， 也就是说电脑的落子要使utility最大， 而你落子要使 utility最小；

3. Minimax Search的搜索深度限定为3 (depth=3足够得到合理的落子)， 也就是你落子的时候会往 前多算两步；

4. 你先落子；

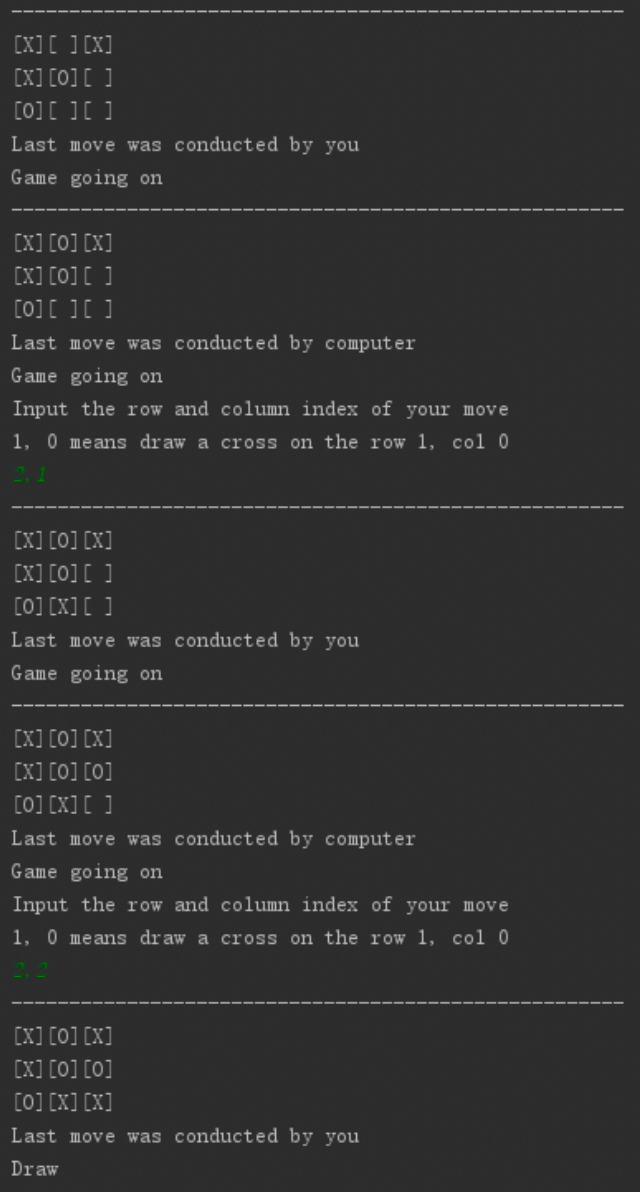
Minimax Search算法的基本流程为， 对于电脑玩家：

1. 找出当前状态下所有可以落子的地方；

2. 采用depth limited minimax search方法估计每一个落子的地方的utility；

3. 在utility最大的地方落子；

**4. 正确运行样例**



**五、成绩评定**

代码补充实现占50%,report占50% 。