程序由回合判断，棋盘存储，读取数据，吃子，可行性判断，计算最优解，输出部分组成

**回合判断(round)**：

**棋盘存储**：定义数组chess[16][16]，其中1代表黑棋，-1代表白棋。

**可行性判断(feasibility)**：

满足以下条件（以右图\_向右可行\_为例）：

该点右侧一位是白子

该点向右循环多个均为白子

该点在黑子处停止

输入：

点的位置**I , j** in chess[i][j]

输出：

存储每个可行点的数组ok[2][2][]

返回值 ok\_direction ：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **向右可行** |  | **-1** | **向左可行** |
| **2** | **向上可行** |  | **-2** | **向下可行** |
| **3** | **向右上可行** |  | **-3** | **向左下可行** |
| **4** | **向右下可行** |  | **-4** | **向左上可行** |
| **0** | **不可行** |  |  |  |

**计算最优解(optimum\_solution)：**

输入：存储每个可行点的数组ok[2][2][]

**吃子(eaten)**：

Get ok\_direction //假设是1

向右循环，若非起始子，乘以-1

结束

输入:

点的位置**I , j** in chess[i][j]

可行性判断的返回值 ok\_direction

原始数组 chess[][]

输出：

操作后的数组chess[][]

**输出(output)：**