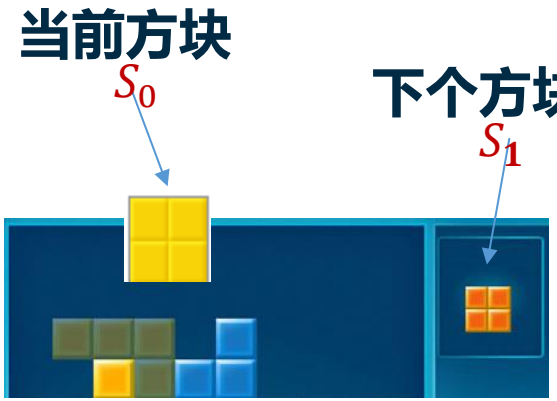
课题任务



输入：

OO

输出：

0 3
0

第一轮

- **输入：**
  - **第一行：两个单字符 S_0和S_1**
  - S_0 第一个方块（当前需要放置的方块）
  - S_1 第二个方块（候选区中的方块）
  - 例如：**OO**（当前O，候选O）
- **输出：**
  - **第一行：旋转角度、坐标**
  - c 顺时针旋转角度（0, 90, 180, 270）
  - x 方块在x轴上的最小值（x坐标）
  - 例如：**0 3**（不旋转，x轴取值3）
  - **第二行：得分**
  - 例如：0（没得分）

课题任务

输入：

O

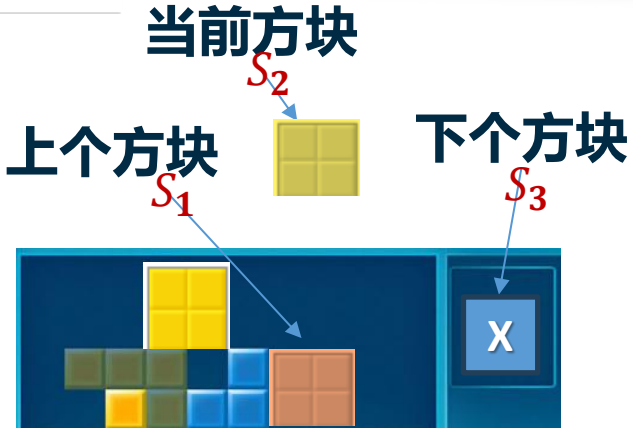
输出：

0 6
0

第二轮

- **遗留：**
  - S_1 第二个方块（候选区中的方块变成当前需要放置的方块）
- **输入：**
  - **第一行：单字符 S_2**
  - S_2 第三个方块（候选区中的方块）
  - 例如：**O**（候选O）
- **输出：**
  - **第一行：旋转角度、坐标**
  - c 顺时针旋转角度（0, 90, 180, 270）
  - x 方块在x轴上的最小值（x坐标）
  - 例如：**0 6**（不旋转，x轴取值3）
  - **第二行：得分**
  - 例如：0（没得分）

课题任务



输入：

X

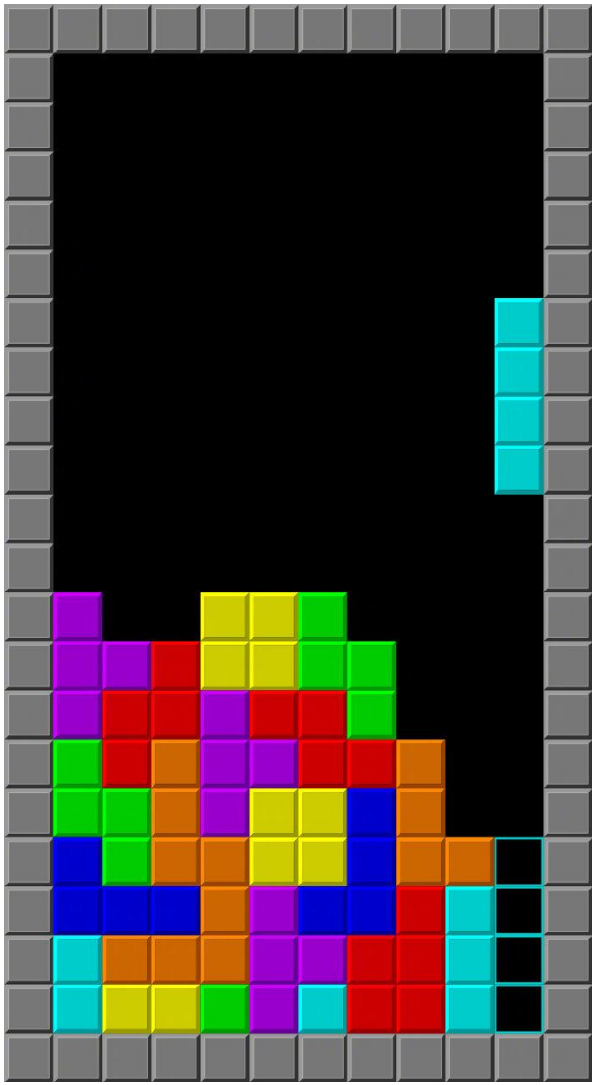
输出：

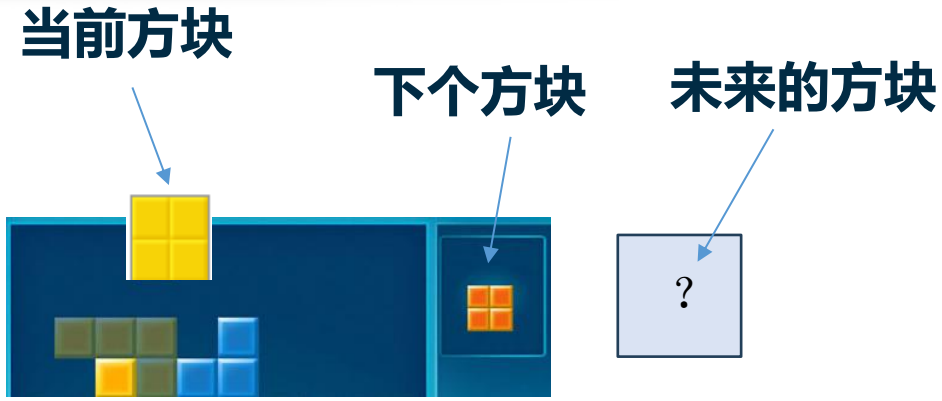
0 8
0

第n轮

- **遗留：**
  - S_{n-1} 第n-1个方块（候选区中的方块变成当前需要放置的方块）
- **输入：**
  - **第一行：单字符 $S_n = X\ or\ E$**
  - $S_n = \mathrm{X}$ 代表通关（正常结束）
  - $S_n = \mathrm{E}$ 代表失败（挑战失败）
- **输出：**
  - **第一行：旋转角度、坐标**
  - c 顺时针旋转角度（0, 90, 180, 270）
  - x 方块在x轴上的最小值（x坐标）
  - 例如：**0 8**（不旋转，x轴取值8）
  - **第二行：得分**
  - 例如：0（没得分）

课题任务





- 过程控制
  - 输入方块类型
  - 输出旋转角度、放置位置与得分
- 根据输入方块类型计算放置位置
  - 旋转角度（0, 90, 180, 270）
  - 放置位置（x轴坐标）
- 计算消除成绩
  - 不同消除层数获得不同得分

俄罗斯方块的放置策略

平稳堆叠

原则1：尽可能保持平稳的堆叠方块，平坦的地形上可以放置各种方块

不要有一个以上的高度差超过3

原则2：超过3格高的虚洞需要 | 来填充，消除几率太小

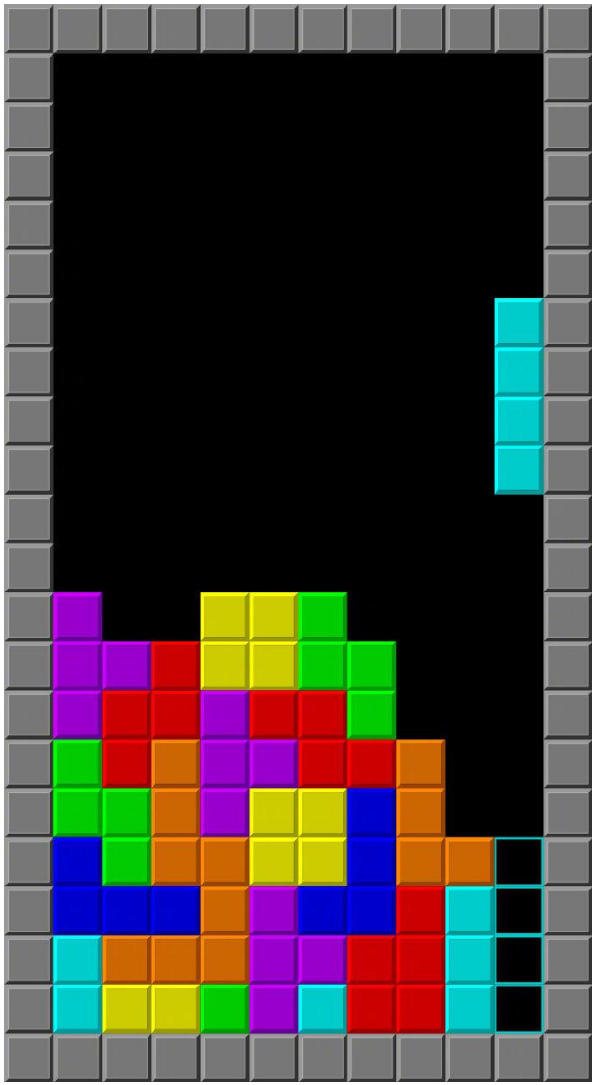
平放方块

原则3：尽可能将方块水平放置，减少宽度为1的虚洞

俄罗斯方块的放置策略

- **空洞维护：**洞越少越好
  - 实洞：有遮挡，除了将上层消除，否则无论通过放置消除的洞
  - 虚洞：可以通过下落消除的洞
- **高低关系：**维持平坦
  - 平台越多越好：连续2个以上一样高的为平台（越多越平坦）
  - 有1个高度差的高低台：左高右，右高左，至少各有一个（支持SZ）
  - 有2个高度差的高低台：左高右，右高左，可以各有一个，再多了就不好了（可以落F）
  - 有3个以上的高低台：可以有一个，再多了就不好了（需要 I）
  - 总高度差：4个高度差之内可接受，越大越不好
  - 总高度：总高度越低越好
- **策略选型：**
  - 激进：争取更多消除行，一行惩罚，两行不奖，三行四行大奖
  - 平衡：一行不奖，2，3，4行均奖
  - 保守：一行奖励，尽量消除

俄罗斯方块的放置策略v1



- **假设方块类型和旋转角度是确定的，最佳的放置行为是什么？**
  - 一次消除多行最好
  - 尽可能少留空洞（有消除不掉的风险）
  - 最高和最低行差不要太大（有结束风险）
  - **好的行为要奖励：**
    - 一次消除4行最好（100）
    - 一次消除3行次之（90）
    - 一次消除2行较好（80）
    - 一次消除1行能接受（70）
    - 将实洞变为虚洞（20）
    - 降低了高低行差（10）
  - **差的行为要惩罚：**
    - 出现了空洞（-20）
    - 增大了高低行差（-10）
    - 增大了总行高（-10）

俄罗斯方块的放置策略v2

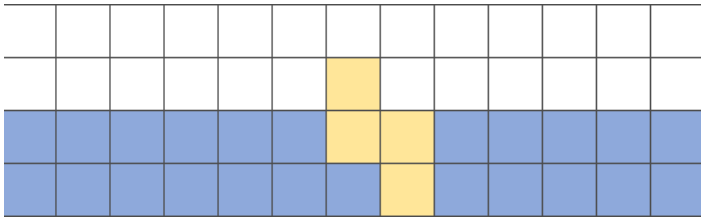
- **能否有更通用的判断方式?**
- **选择评估参数**
  - **消除行数**：当前方块落下后能消除多少行
  - **平均高度**：消除行后，当前局势的平均高度
  - **空洞数**：消除行后，当前局势还有多少个实洞
  - **平整度**：消除行后，各列之间的高度差绝对值之和（虚洞数）
- **选择评估函数**
  - 如何用选择的参数构建出一个可用于对比的值
  - Value = ∑ 权重×评估维度

评估维度	价值计算（平衡）	评估权重
消除行数	1行不奖励（0），2~4行奖励递增（100/500/1000）	50%
平均高度	高度增加1不惩罚，增加2~4惩罚（-20/-40/-80）	20%
空洞数	空洞数增加1不惩罚，增加2~4惩罚（-20/-40/-80）	20%
平整度	高低差增加1不惩罚，增加2~4惩罚（-20/-40/-80）	10%

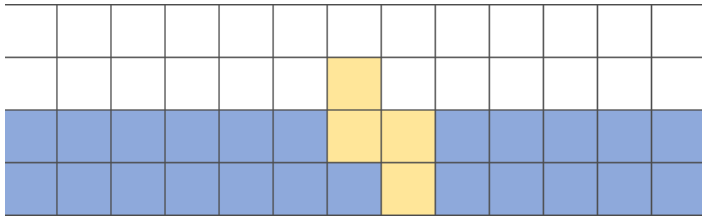
这是个例子，需要后继进一步调整得分和评估权重

Pierre Dellacherie算法

- **Pierre Dellacherie算法以方块堆叠矮和紧凑为目标，只考虑局部最优，不考虑全局最优**
- **评估参数**
  - **LandingHeight**：下落后的高度，即放置后方块重心距离底部的距离
  - 例如：方块下落后距离底部距离是 0

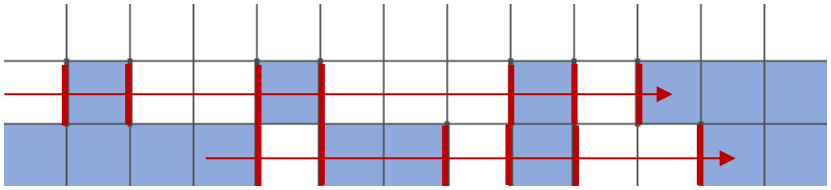


  - **ErodedPieceCellsMetric**：消除贡献值，等于消除行数乘以该方块参与消除的格子数，代表消除的行数与当前摆放的板块中被消除的小方块的格数的成绩
  - 例如：方块下落之后会消除2行，而且自身贡献的小方格数是3个，所以返回值是3*2=6

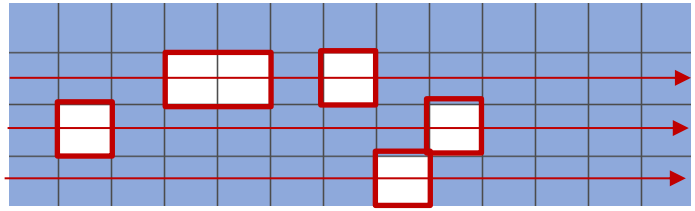


Pierre Dellacherie算法

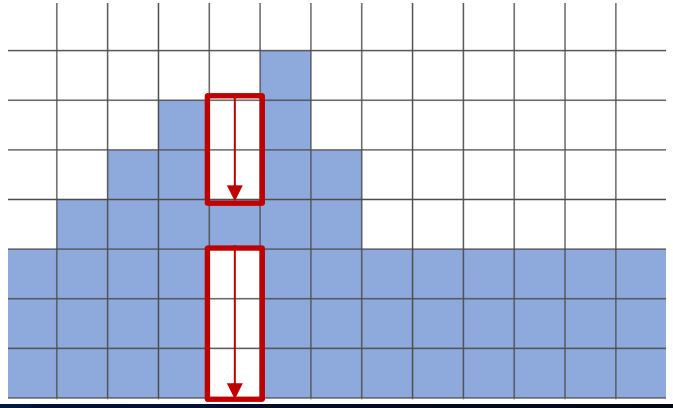
- **RowTransitions**：行变换数，从空白格进入被占格或从被占格进入空白格算作一次变换，按行遍历
- 例如：第一行为7次变换，第二行为6次变换



- **ColTransitions**：列变换数，计算上下边界的变换次数，按列遍历
- **BuriedHoles**：空洞数，每列中某个方块下面没有方块的空白位置
- 例如：空洞数为6

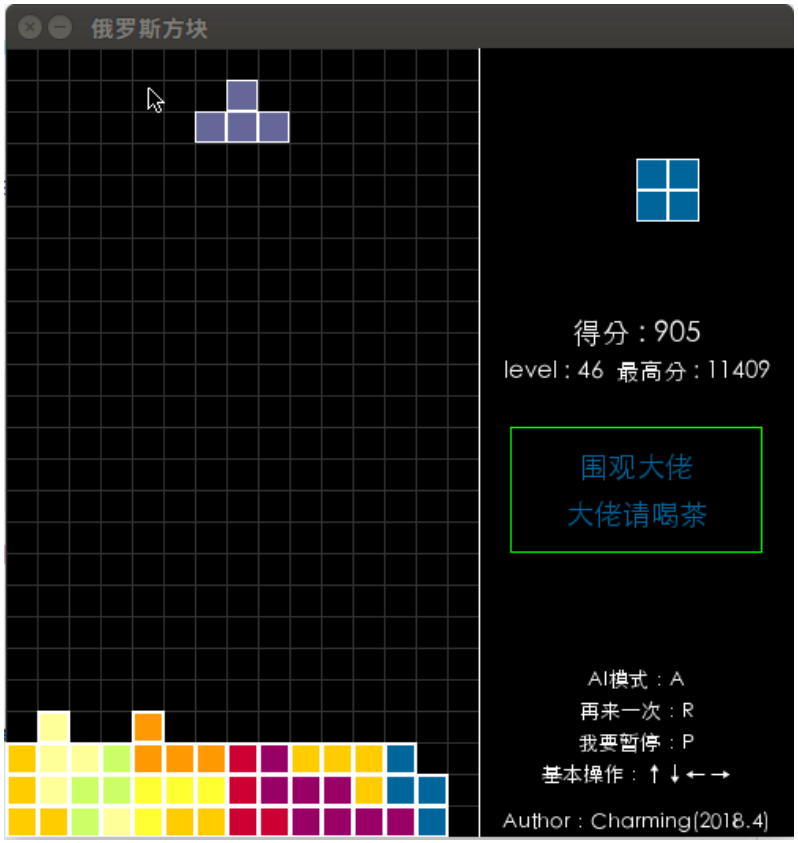


- **Wells**：井数，指某一列中，两边都有方块的连续空格，返回值为井的深度的连加
- 例如：图中有两个“井”，深度分别为2和3



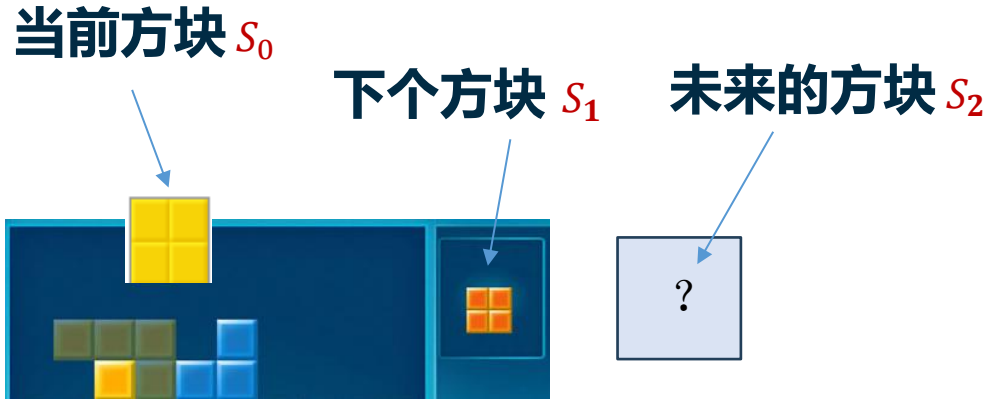
Pierre Dellacherie算法

- **评估函数：**计算每个可能放置位置的价值，value越大该位置越优
  - value = -45 * LandingHeight + 34 * ErodedPieceCellsMetric - 32 * RowTransitions - 93 * ColTransitions - 79 * BuriedHoles - 34 * Wells
- **优先度：**当出现两个位置评分相同时，选择priority小的
  - priority = 100 * 水平移动次数 + 旋转次数（对我们这个课题意义不大）



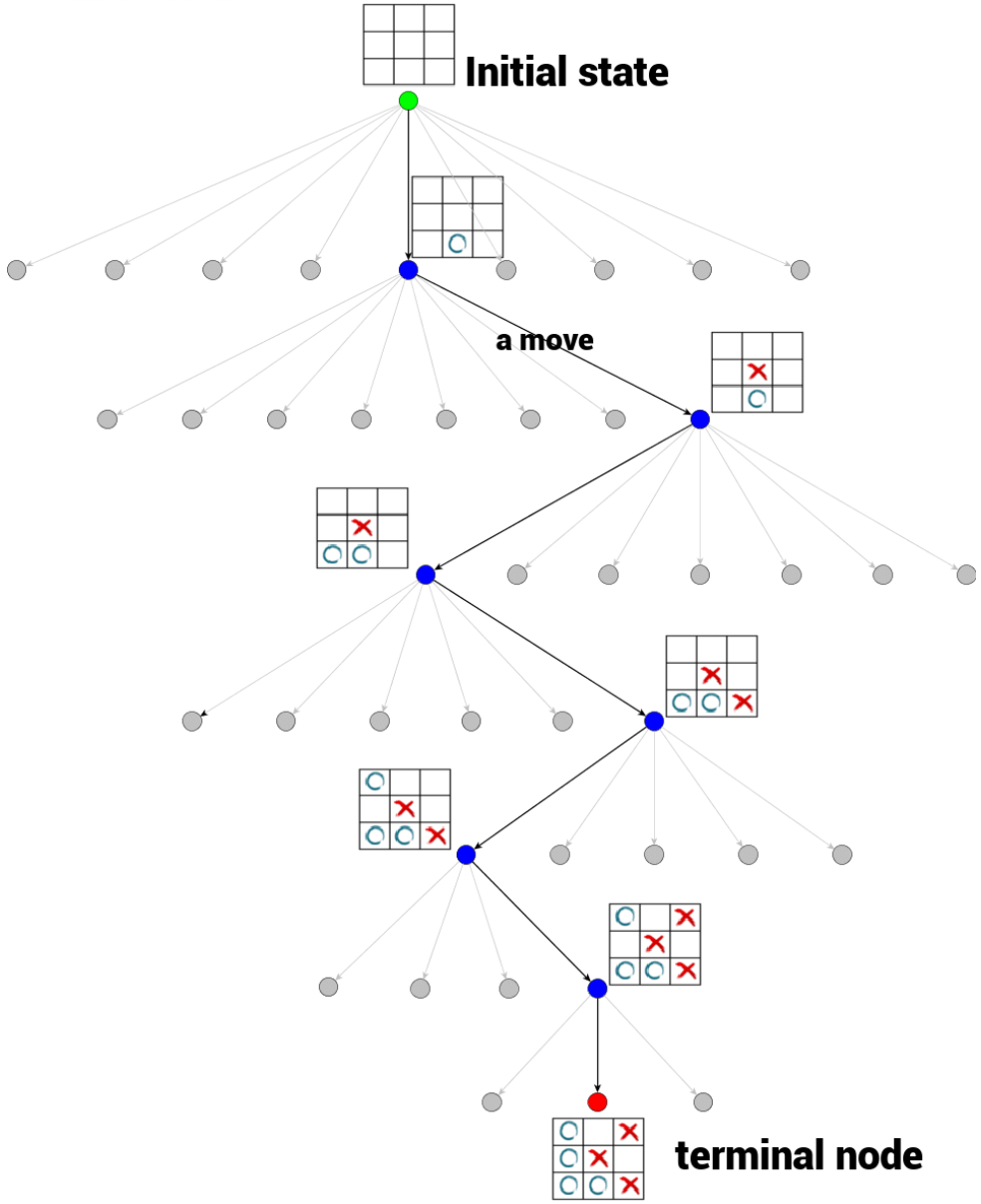
从局部最优到全局最优

- 实现步骤：
  - 获取某方块的所有形态可以放置的所有位置
  - 计算所有位置的value值和priority值
  - 比较值的大小找到最优位置
- 问题：
  - 当前方块的放置可能影响后继方块的放置（产生空洞等）
  - 当前方块最优放置位置可能导致后继方块难以获得最优放置
  - 当前方块放置位置可能与后继方块放置位置一起获得更优奖励
  - …

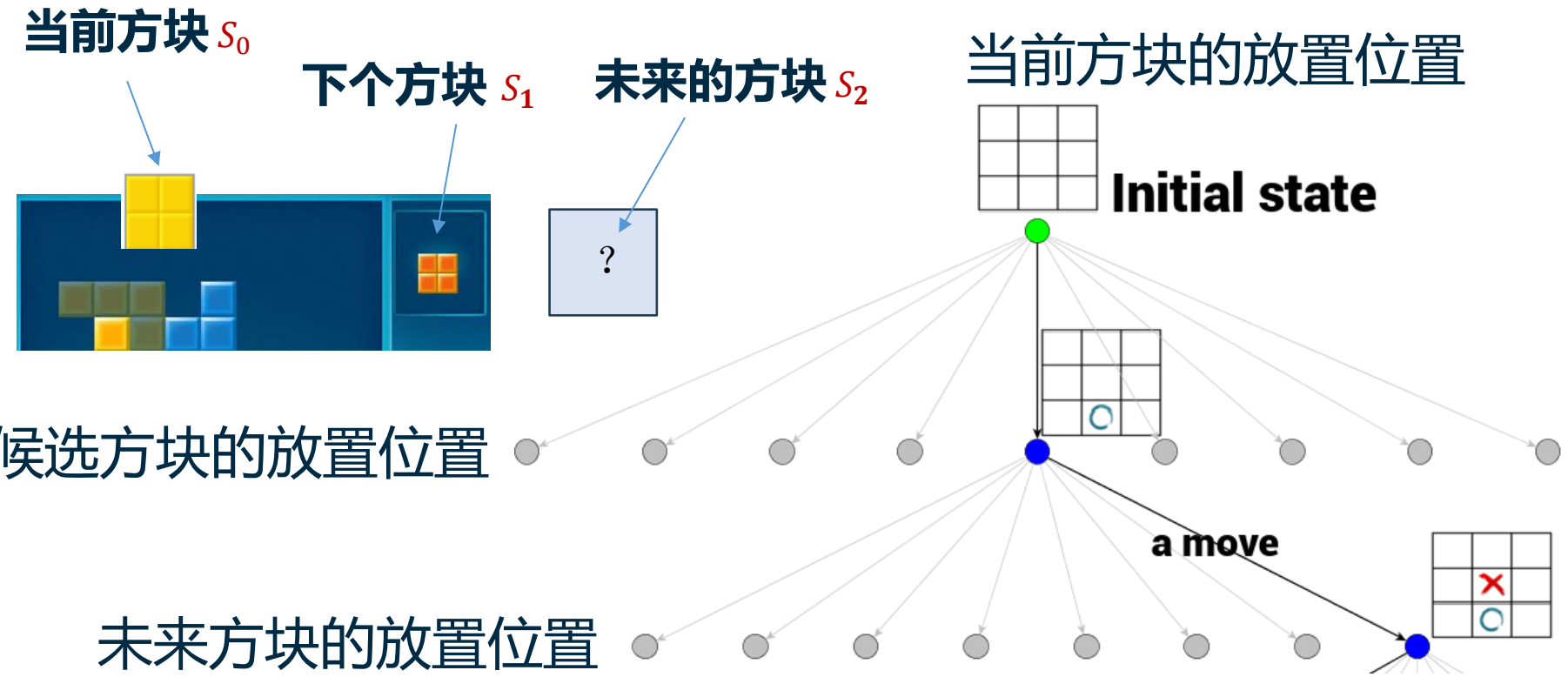


如何判断当前决策是不是最优的?

博弈类游戏过程可以用一颗树表示，树的节点是盘面当前状态（如围棋的盘面，俄罗斯方块的盘面），树的每一个分支是博弈双方的决策（如围棋的落子决策，俄罗斯方块的放置决策）



俄罗斯方块的最佳放置策略探索



- 模拟当前方块的所有放置方式（最多4种旋转×10个位置）
- 模拟每一种放置方式条件下，候选方块的所有放置方式（最多4种旋转×10个位置）
- 扩展模拟未来方块的放置方式，并计算可能的收益

俄罗斯方块的评价空间

- I, T, O, J, L, S, Z：7种类型
- 0, 90,180, 270： 4种旋转角度
- 10种放置位置（x轴坐标）

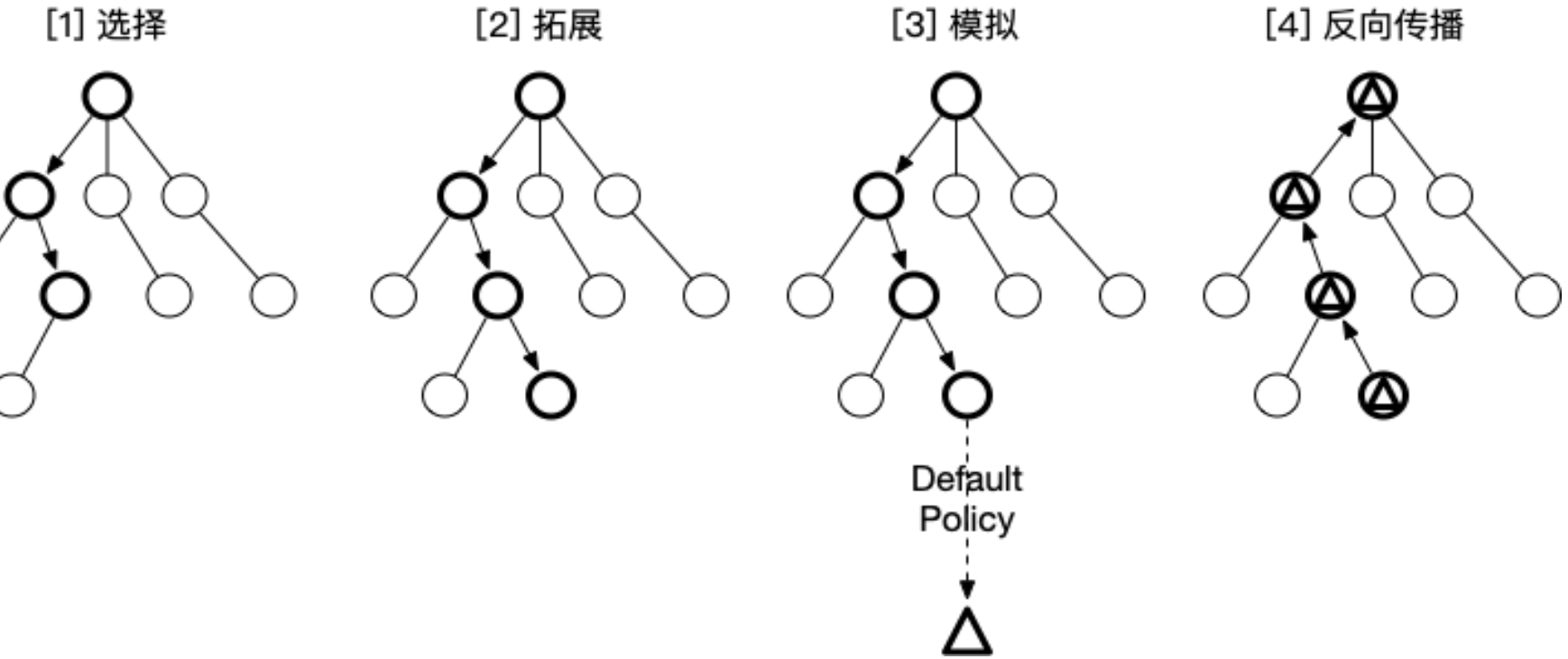
- S_0方块类型是确定的，旋转角度是可变的
  - 1×4×10=40种
- S_1方块类型是确定的，旋转角度是可变的
  - 1×4×10=40种
- S_2方块类型是不确定的，旋转角度是可变的
  - 7×4×10=280种
- 如果迭代2层：40×40=1600组结果综合评价
- 如果迭代3层：40×40×280=448000组结果综合评价
- …**不能无限扩展探索空间**…

蒙特卡洛树搜索

蒙特卡洛方法(Monte Carlo method)

蒙特卡洛树搜索（Monte Carlo Tree Search, MCTS)

- 高效的启发式搜索算法



蒙特卡洛树搜索

每个节点在模拟之后，需要将模拟结果反向传播更新拓展树
可根据反向传播的结果评估不同分支的拓展价值
基于评估结果选择后继拓展分支进一步拓展模拟