在第一阶段的重力四子棋游戏中，我们利用蒙特卡罗树搜索算法（MCST）设计了AI模型，该模型通过计算大量的模拟，帮助我们在博弈中逼近完美落子策略。

**算法框架介绍**

从选择，扩展，模拟，回溯四个阶段介绍蒙特卡洛搜索树算法：

第一阶段——选择：从根节点开始，按照UCT准则递归选择最优的子节点，最终到达叶子节点，其中每个节点代表一个棋盘的状态，子节点则对应该状态下落子后形成的棋盘状态。UCT公式作为评估每个节点探索价值的度量，利用超参数c平衡了经验胜率和胜率的不确定性(综合考虑了节点的获胜次数和访问次数)；在模拟游戏的过程中，我们发现基于蒙特卡洛的AI在棋局后半程有时出现考虑不周的情况，没有及时发现对手的威胁进攻，因此，设定超参数c随棋盘进行，逐渐线性地从1增加到1.1，主动拓宽蒙特卡洛搜索树的宽度，优先考虑更多的可能性；与此同时，基于重力四子棋为二人零和博弈问题的性质，故相邻两层节点存储的经验胜率的意义是相反的，我们采用negamax思想，每向下递归一层，棋手的身份就变化一次，并将对手假想为绝对理性人，总是做出最优的选择。

第二阶段——扩展：创造当前叶子节点下的一个或多个子节点，即从当前节点未被尝试过且可行的动作列表中，随机选择一个动作并落子，表征新的棋盘状态，作为新生成的子节点。

第三阶段——模拟：从上一阶段新生成的节点出发，开始用随机策略进行一轮游戏，直到本次模拟的棋局结束，返回棋局结果。

第四阶段——回溯：利用上一阶段模拟游戏的结果，基于negamax思想，更新由当前节点至根节点路径上的经验胜率。若模拟结果是平局，则各祖先节点的胜利次数增加0.5，若否，则胜利次数以此交替增加0或1，模拟次数统一增加1。

**创新优化与算法提速**

引入 numba 库进行提速：在引入numba库进行加速后，每秒模拟次数的数量级从10^3提升到了10^4。我们对各个节点的胜率估计基于极大似然法，因此，越多的模拟次数，越能减小方差，能得到更稳健的胜率估计，也就越逼近完美的必胜策略。

先查看必胜节点：模拟前先检查棋盘是否有必胜的落子位置，若有，则直接落子。这样一方面减少了模拟的时间消耗，另一方面避免了因随机性而忽视该必胜落子位置的可能。

及时阻止对手连线：模拟前先检查棋盘是否有对方必胜的落子位置，若有，则直接落子。尽管这样无法保证阻止对方胜利，但能减少对方一步制胜的潜在威胁，从而提高我方获胜机会。

第一步基于经验观察：在观察高分数AI的基础上，发现第一步落子于中间区域胜算更大，故我们不为第一步落子分配时间，而直接改为下在棋盘的中间区域。该策略来源于人类的观察经验，并为之后的思考留下更多的超额时间。

自适应的超参数 c 增长机制：定义系数k=n/N，其中n是棋盘上已有的双方棋子数之和，而 N 是棋盘上所有的落子位置数量。设定超参数c=1+0.1\*k，随棋局进行，c逐渐从1增大到1.1，使得AI更倾向于探索模拟次数较少的节点。这样有助于 AI 在棋局的后半程“看到”更多可能性，提前“意识到”对手可能的连线趋势，进而阻止对方获胜。

超额时间分配函数：通过观察分析本班同学的AI对战结果，我们发现大多比赛在棋子填满3/4的棋盘之前结束，故我们将大多数的超额时间分配在棋局前半程，更充分地利用超额时间。此外，越靠近棋局终点，则能探索的潜在节点会越少，需要的探索时间应该随比赛的进 行，按照指数下降的方式进行分配。最后，由于初始的探索空间非常之大，而且双方博弈的随机性也很大，所以我们为初始的几个节点分配的时间相对较少。我们设计的时间分配函数为： t = tstep + min(k ∗ ttotal, 5) 其中，t 是为本次思考分配的超额时间，tstep 是系统默认的每步思考时间（默认为2秒），ttotal是剩余的总超额时间，k的含义与上文一致，用于衡量棋局的进度。