PA3 重力四子棋（Connect4） 实验报告

1 实现思路

1.1 基本算法

我在设计AI时采用了蒙特卡洛方法和信心上限树，在每个回合建立UCT树，通过随机落子、用信心上限函数拓展最优的叶节点，最终选择根节点的一个最优子节点落子。

每次扩展，从根节点出发，若当前节点可以扩展新的子节点，则随机选择其中一个未扩展的子节点进行随机模拟；否则根据信心上限公式，选择数值最大的一个，递归执行上述扩展过程，直到找到一个可以扩展的节点，随机选择它的一个未扩展子节点进行随机模拟。模拟时，双方轮流随机落子，直到一方胜利。用这一结果更新所有祖先的得分，若胜者和该节点的决策者相同，则+1，否则-1。信心上限公式如下：

$$\frac{Q(v)}{N(v)}+C\sqrt{\frac{2\ln(N(u))}{N(v)}}$$

其中 v 为 u 的子节点，$N(v)$ 为被回溯到的次数，$Q(v)$ 为回溯过程中累积的得分。

1.2 创新算法

最优落子策略可以抽象为，给定一个局面，输出一个胜率最高的落子点，如果我们将随机落子改为按照最优落子策略，那么只用一次模拟即可，但目前我们无法形式化地表达一个最优落子策略，因此需要通过多次模拟来反映大致的趋势，因此，我们可以断言，在随机落子中引入一些策略，可以让随机模拟的结果更加接近真实的胜率。具体而言，我实现了一个 `passive_policy` 函数，即“消极策略”，它检查对方上一次落子处是否形成了三枚棋子相连的情形，如果有，并且下一轮可以立即封锁住走势，那么下一次就必然落子在那里；若没有，则下一次落子仍然随机。这一策略可以极大地加快模拟结束的判定、提高单次模拟的效果、增加模拟次数。

2 实现细节

在 `UCT` 命名空间中，我实现了 `chessboard,mcnode,optim` 三个类。

`chessboard` 用一个 `std::bitset<192>` 来记录当前局面，1表示当前位置有落子，用两个 `chessboard` 的实例即可完整描述一个局面。利用 `bitset` 的性质，也能高效的判断是否达到终止条件。

`mcnode` 是蒙特卡洛树节点，其中

- `mcnode *prev` 记录父节点
- `mcnode *child[]` 记录子节点
- `int nullkids,kids` 记录还未扩展的子节点个数，以及总的子节点数
- `who player` 记录当前局面的决策者
- `int nsim,score` 表示模拟次数、模拟总分
- `int deadEnd` 表示当前局面是否为终局

通过预先开一个 `mcnode pool[]` ，每次在这个池子中取出一个节点，可以避免 `new` 带来的延时。

`optim` 是优化器，其中

- `chessboard my, op` 记录局面
- `int top[]` 记录每一个槽的顶部位置
- `update` 用于回溯更新
- `passive_policy` 用于辅助随机落子
- `simulate` 对传入的节点进行模拟，并回溯更新

此外，还有函数

- `nextStep` 根据根节点的子节点的信息计算下一步的最佳落子位置
- `calcState` 计算传入的节点是否是终止状态
- `extendLeaf` 用于找到将被随机模拟的节点

策略过程如下：

1. 调用 `UCT::extendLeaf` 找到被扩展的节点
2. 定义一个 `UCT::optim` 实例，并调用其 `simulate` 方法进行模拟、更新
3. 若时间还没到 `THRESHOLD`，则返回 1 继续进行模拟
4. 调用 `nextStep` 得到下一步落子位置，返回答案

3 实验结果

以下提供批量测试的网址：

`https://www.saiblo.net/batch/23820`

总计 98 胜，2 负，0 平，胜率为 98%。

由于两轮测试显然不足以充分反映AI的相对战力，因此我在本地进行了更加充分的测试，本机 CPU 为 `i7-4850HQ, 2.3GHz`。用我的 AI 与 `92.so` ~ `100.so` 分别对战了50轮（100局），总共500次对局，胜率汇总如下：

92.so	**94.so**	**96.so**	**98.so**	**100.so**
89%	75%	91%	92%	80%

将 AI 的时间限制略微调整后派遣，在游戏的天梯上排名为 #80 左右。

4 总结

在实验中我充分探索了信心上限树的诸多性质，总结如下

1. 模拟次数越多，AI能力越强，我提交的AI目前能在最开始的几轮中模拟 10^6 次，后续几轮模拟次数可以达到 $3 \cdot 10^6$。
2. 性能越好的AI，在与策略足够优秀的对手对战时，对对方落子的预测是非常准确的（这是由于重力四子棋的行棋策略非常有限）。助教可在本人的 `Makefile` 中加入 `-DLOCAL` 选项查看调试信息，可以发现，`100.so` 的落子几乎和我的AI的预测一样
3. 常数 C 用来调整不同子节点之间的模拟次数的均衡性，事实上，若在随机模拟中采用了较好的策略，C 可以设置得非常小，从而将所有的模拟都放在最优的那个子节点上，若用必胜策略进行模拟，则蒙特卡洛树自然地退化成一条链。