My understanding to MCTS:

Monte Carlo tree search is a classic tree search algorithm. The core idea is to put resources on branches that are more worthy of searching, that is, to concentrate computing power in more valuable places.

The MCTS algorithm is mainly divided into four steps, namely selection, expansion, simulation, and backpropagation.

STEP 1: Selection

Starting from the node, the optimal child node is recursively selected, and finally a leaf node is reached.
Upper Confidence Bounds (UCB) is the basis for judging the quality of nodes:

$$UCB\ \ (S_i) = \overline{V_i} + c\sqrt{\frac{\log N}{n_i}}, c = 2$$

v is the average value of the node; c is a measurement constant; N is the total number of explorations; n is the number of explorations of the current node.
Small c focuses on utilization; Big c focuses on exploration.

STEP 2: Expansion

If the current leaf node is not a terminal node, we'll create one or more child nodes and select one of them for expansion.

STEP 3: Simulation

Starting from the expansion node, run the output of a simulation until the end of the game.

STEP 4: Backpropagation

Using the results of the third step simulation, the repercussions are propagated to update the current action sequence.

Code idea:

1. Establish the root node from the current situation, generate all the child nodes of the root node, and simulate the game respectively;
2. Starting from the root node, perform the best first search;
3. Use the UCB formula to calculate the UCB value of each child node and select the child node with the maximum value;
4. If this node is not a leaf node, use this node as the root node and repeat step 2;
5. Until a leaf node is encountered, if the leaf node has not been simulated before, simulate the game for this leaf node; otherwise, randomly generate child nodes for this leaf node, and simulate the game;
6. Update the node and ancestor nodes at all levels according to the corresponding color with the benefits of the simulated game, and at the same time increase the number of visits to all nodes above this node;
7. Return to 2, unless the search time of this round ends or the preset number of cycles is reached;
8. Select the child node with the highest average profit from the current situation and give the best move.
9. Other strategies to make win more possible ……

My contribution in the project:

I wrote the basic code for at least 24 hours (6 hours×4 days) in the week starting from November 6th, which was able to realize the basic Monte Carlo tree algorithm (including the 2 classes, basic 8-9 methods, check_winner functions, etc), and the live-3 live-4 checking strategy.

After that, with the joint discussion of the plan by our group members, I further participated in the code modification of strategy errors, improvement of loopholes, addition of forbidden moves, prediction within a specific range, AI vs AI testing, etc.

Peer evaluation:

CHEN, Longyin: 5
FENG, Zekai: 5
SHAO, Kuichen: 5
ZHANG, Mingtao(me): pass