Mid-Term Project -Report

許琨杰 莊若其 李曄彤

Division of labor:

李曄彤(x1052047)：

- Build the chessboard
- Update the chessboard after moves
- Determine if the game is terminal or not, and return the winner if there's any.

莊若其 (103062336)：

- Implement four steps of MCTS (Selection, Expansion, Simulation, BackPropagation)

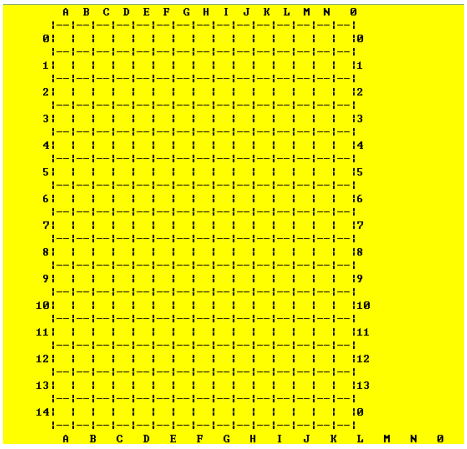
- Implement "Node" data structure for tree search
- Use UBT as the child selection policy

許琨杰(103031111)：

- Design an evaluation function for simulation in MCTS
- Evaluate the scores of the vacancies near the stones
- Search for potential lines and raise their scores

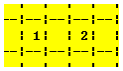
李曄彤

When I need to build a chessboard, because the place of chess piece can be presented by coordinate (x, y), I decide to define an two-dimension array chessboard[Length][Length].The length is 15.In fact, x and y are integers, as to the requirement that the coordinate is (number, character)(ex:6H),I used a temp value to transfer the number into the ASCII of a character or to transfer the ASCII of a character into a number. .For example, when I enter the coordinate 6A, it means the real array coordinate is (6,0).When chessboard[6][0]=1, it means the human has occupied the point.When chessboard[6][0]=2, it means the computer has occupied the point.When a point is not occupied, the value of the array is 0.



I define a function called draw() to draw a real chessboard.The framework is composed of two main character "-"and "|".In order to make the human move the chess easily,I need to make the procedure show all coordinates.Also,I need to present every chess piece on the board.So I define another multi-dimensions array to show them.For example, if(chessboard[j][i]==BLANK) strcpy(p[j][i],"\0").Through the sentence"printf(" %2dl%2sl%2sl%2sl%2sl%2sl%2sl%2sl%2sl%2sl%2sl%2sl%2sl %2sl%2sl%2sl%d\n",j, p[i][0],p[i][1],p[i][2],p[i][3],p[i][4],p[i][5],p[i][6],p[i][7],p[i][8],p[i][9],p[i][10],p[i][11],p[i][12],p[i][13],p[i][14],j)",the character"|"and "-"and the output of array p compose the real board.Print the whole board we can get the following graph. I define a struct Point including (X,Y) as the piece array subscript and then,I define two points &Point1,&Point2.Point1 always represent human and Point2 always represent computer.Through the function "Makepoint",we can renew the array and

redraw the board.At the same time, we can distinguish between different players.



If we want to judge the winner every step, we need a function getWinner.I make the function get the coordinate of the chess piece which is placed latest step and then count the chess pieces with same color(same value of player) in different directions including(up and down, right and left, upper left and lower right, upper right and lower left).If the count of different color chess pieces is not less than 5, the function would return player.,otherwise the game would continue.

莊若其

In the project, there're three classes - Game, MCTS, Node. I implemented Node and most of MCTS.

Node:

```
class Node
{
public:
    Node *parent = NULL;
    vector<Node*> children;
    int wins = 0;
    int visits = 0;
    coordinate action;
    int player = -1;
    int depth = 0;

    int state[15][15] = {{0}};
    int number_of_chess;

    Node(Node* parent, coordinate action, int player, int depth, int (*state)[Length], int number_of_chess);
    string ToString();
};
```

For each node in the tree, it holds a 15x15 board which can represent the state in each node. Also, the "action" is a coordinate, indicating the latest step of the state; that is, the latest move which ends up with that "state". For "wins" and "visits" value, they'll be used to calculate the "UBT" to pick the node with highest UBT, and the "action" of the node is the best coordinate for player.

MCTS:

```
class MCTS
{
public:
    Node* Selection(Node* current, int (*board)[Length]);
    Node* Expand(Node* current, int (*board)[Length]);
    int Simulate(Node* current, int (*board)[Length], int player);
    void BackPropagation(Node* current, int value);

    Node* getBestChild(Node* current, int Cp); // by UCT
    coordinate getBestAction(int (*board)[Length], int player, int g_iPointLen);
    int opponent(int startPlayer);

    set<coordinate> getValidMoves(const int (*state)[Length], int player);
    bool isTerminal(int (*board)[Length], coordinate move);
    int getWinner(int (*board)[Length], coordinate move);
    void mark(int (*board)[Length], int player, coordinate move);

};
```

The "Game" in main.cpp will call "getBestAction" to get the best coordinate to move.

“getBestAction” will run MCTS for 5000 times, and then it will pick the node with highest UBT to return. For each run of MCTS:

```
Node *current = Selection(root, board);
int value = Simulate(current, board, opponent(player));
BackPropagation(current, value);
```

In “Selection”:

```
do{
    set<coordinate> validMoves = getValidMoves(current->state, current->player);
    if(validMoves.size() > current->children.size()) {
        Node *node = Expand(current, board);
        return node;
    }
    else {
        double Cp = 1.44;
        current = getBestChild(current, Cp);
    }
}while(!isTerminal(current->state, current->action) || current->number_of_chess>0);
return current;
```

It will select the child node of current node with the highest UBT. As it encounters an “action” which is new to the current node, a new child node with the action is therefore created and added to the tree. (This is “Expansion”)
At the end, the function will return a node with terminal state or a node which is new to the search tree.

In “Simulate”, a “getValidMoves” is called, and will return a set of moves that is beneficial to the player. And the function will pick one of them randomly to simulate the game. And it will return a value, which could be 1 or 0. “1” represents win, “0” represents lose. The value will be added to “wins” value of all the nodes in that path; also, the “visits” value of nodes will increase by 1 in each run of simulation. (That is BackPropagation)

UBT is calculated by the formula: $UCT = \overline{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}}$

許琨杰

Summary

Our group takes almost 2 weeks to complete the project during the midterms period.Even if we all take great pressure to complete it, we all try our best to make it more perfect.
After we accomplished our own part of the project and wanted to unit them, we found many bugs including the repetition of different user-defined libraries, which troubled us during a long period.Through our effort we dealt with the problem.And then, we found that the characters of the chessboard were coding in different types so they were represented in different ways in different computers.We had to draw the chessboard again use common characters.

許琨杰

getValidMoves()

In the simpliest AI of Five in a Row chess game implemented by Monte Carlo Tree search, the child nodes in the steps like selection and expansion are simply chosen by random method which is not heuristic at all. Therefore, I implemented the getValidMoves() which contains two evaluation functions that would compute the scores of the vacancies near the stones and searching for potential lines and raise thier scores. After that, getValidMoves() would send a set of coordinates which have the highest or the second highest scores to do MCTS.

Steps:

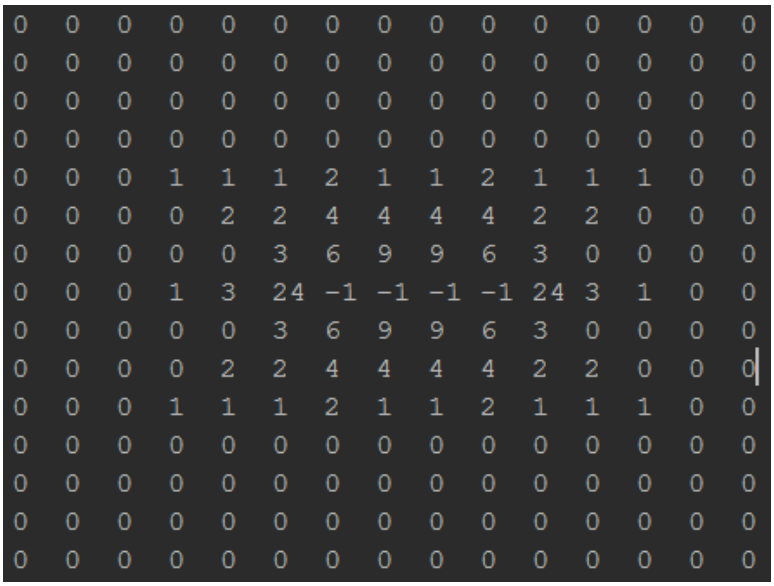
0. temp_board: a copy of current board which uses 0, 1, 2 to represent vancant points and the stones
1. manager()
   In order to make the later board manipulation easier, manager() will convert 1 and 2 in the temp_board to -1 and -2.
2. grader()
   Compute the the scores near the stones in a gradient descent form.

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  1  0  0  1  0  0  1  0  0  0  0
0  0  0  0  0  2  0  2  0  2  0  0  0  0  0
0  0  0  0  0  0  3  3  3  0  0  0  0  0  0
0  0  0  0  1  2  3 -1  3  2  1  0  0  0  0
0  0  0  0  0  0  3  3  3  0  0  0  0  0  0
0  0  0  0  0  2  0  2  0  2  0  0  0  0  0
0  0  0  0  1  0  0  1  0  0  1  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

3. multiplier()

If there's a n-in-a-row line, multiplier will times n to the scores of the vacant points which are in front of or behind the line. Also, if there's a four-in-a-row line, it will return their coordinates as prioirty.

(This is the longest function I have ever coded, almost **1000** lines!)



4. converter(): looking for the highest, second highest value and send their coordinates to MCTS

getWinner()

This function will find the five-in-a-row line and is used in current board checking and MCTS's simulation. Actually, this function was implemented by my another partner. However, there are bugs in her codes, so I rewrote it by my own style, and it's quite accurate.

At last, even though our AI could perform the game, it was a little stupid, so we spent a lot of time to optimize the MCTS and evaluation function to make it more smarter until we are all satisfied with it.
Though very tired, we would be glad to accomplish a procedure independently and have a profound understanding of MCTS.We also wish a good result of our AI.