Artificial Intelligence Section4

Wendi He 贺文迪 517030910315

1. Chessboard Game

1.1 Q1

There is only one case that player A can win the game. If the white piece and the black piece is just next to each other in the initial position, then the player A will win. Because the player A is supposed to play first and he can directly use the white piece to eat the black piece immediately by moving one step.

In other cases, the player B will eventually win. Moreover, the player B can always win in 4*n rounds. The detail proof will be shown in the following question.

1.2 Q2

We will prove B can eventually win in less than 4*n steps if the white piece and the black piece is not just next to each other in the initial position. The main idea we use to solve the problem here is the Manhattan distance, which is defined as abs(r1-r2)+abs(c1-c2). Besides, it is only danger for the black chess if the Manhattan distance becomes 1 after the steps by the Player B himself.

Since the black chess is able to move to the next first or second grid in each step while the white chess is only able to the next one grid in each step. So, if the Manhattan distance is greater than 3 (i.e. ≥ 4), which means the black chess will not face the threat of being eaten by the white chess, the black chess can use such concrete policy:

(1) If abs(r1-r2)=0, then the black chess will move two grids to decrease abs(c1-c2) by 2. Because the abs(c1-c2) ≥ 4, so the black chess is always able to decrease it by 2 and without facing the problem of being eaten. It is similar if abs(c1-c2)=0. In this case, the Manhattan distance will decrease by 2.

(2) If neither abs(r1-r2) nor abs(c1-c2)=0 equal to 0, then the black chess can move in both directions for one grid each. Similarly, the black chess will not face the problem of being eaten. In this case, the Manhattan distance will also decrease by 2.

As for the white chess, it can go one grid so that it can either increase the Manhattan distance by 1 or decrease the Manhattan distance by 1. Totally in one round, the Manhattan distance will either decrease by 1 or decrease by 3.

We can notice that in each round, when the current Manhattan distance is greater than 3 (i.e. ≥ 4), the black chess can directly follow the policy. So, we only to consider the situation when the black chess finds the current Manhattan distance is smaller than 4 (i.e. ≤ 3). If the current Manhattan distance is 1 or 2, then the black chess can directly move one or two grids to eat the white chess and the player B will just win. So, we only need to consider the situation of the current Manhattan distance is 3.

The first case is the value of abs(r1-r2) and abs(c1-c2) is 0 and 3, which means the two chess are in a line and their distance is 3, then the B can move 1 grid closer to decrease 3 to 2. Then, in the next round, the white chess has 3 choices. The first one is to get 1 grid closer and the Manhattan distance will decrease from 2 to 1 and it will be eaten right after. The second choice is to escape from this line so that the Manhattan

distance will increase from 2 to 3 and the value of abs(r1-r2) and abs(c1-c2) will become 1 and 2, which leads to the second case. The third choice is to keep moving on the line but get farther so that the Manhattan distance will increase to 3. However, the black chess will continue its movement but the white chess cannot keep moving on this line after reaching the border. Then, the white case must move off this line, so it also change to the second case.

The second case is the value of abs(r1-r2) and abs(c1-c2) is 1 and 2. Now, the black chess can move in the direction of the abs value is 2 by 1 grid so it will decrease the Manhattan distance from 3 to 2 and the value of abs(r1-r2) and abs(c1-c2) is 1 and 1. Then, in the next round, the white chess has 2 choices. The first one is to get 1 grid closer and the Manhattan distance will decrease from 2 to 1 and it will be eaten right after. The second choice is to get 1 grid farther which leads to the Manhattan distance increase to 3 and the value of abs(r1-r2) and abs(c1-c2) is still 1 and 2. The literation continues. But the direction for the white chess is limited, which means in each literation, the white chess has and only have two certain directions to go, so the white chess will eventually reach the corner and cannot make the first choice to increase the Manhattan distance to 3 but only has to come up to decrease the Manhattan distance to 1 and will be eaten right after. For example, if r1<r2 and c1<c2 and the Manhattan distance is currently 3, then the white chess can only choose to move up or left in each following round, which will eventually cause it to reach the left-up corner.

During the first process, which decrease the Manhattan distance at the pace of 1 or 3, it will cost less than 2n rounds. Because the maximum for the Manhattan distance is not greater than 2n for any initial position and in each rounds the Manhattan distance will decrease at least by 1, so it would cost less than 2n rounds to decrease the Manhattan distance to 4.

During the second process, since the direction of movement of the white chess is limited in two directions while the white chess will eventually reach the border in less than n rounds for any direction. So, for two directions, the white chess will eventually reach the corner less than 2n rounds. For example, if the white chess can only move up or left, then it can at most n rounds for up and n rounds for left which sums to be 2n rounds. Right after, the white chess will be eaten in the next round.

So, the first process cost less than 2n rounds and the second process also cost less than 2n rounds. To sum up, it will cost less than 4n rounds for the black chess to eat the white chess!

1.3 Programing part

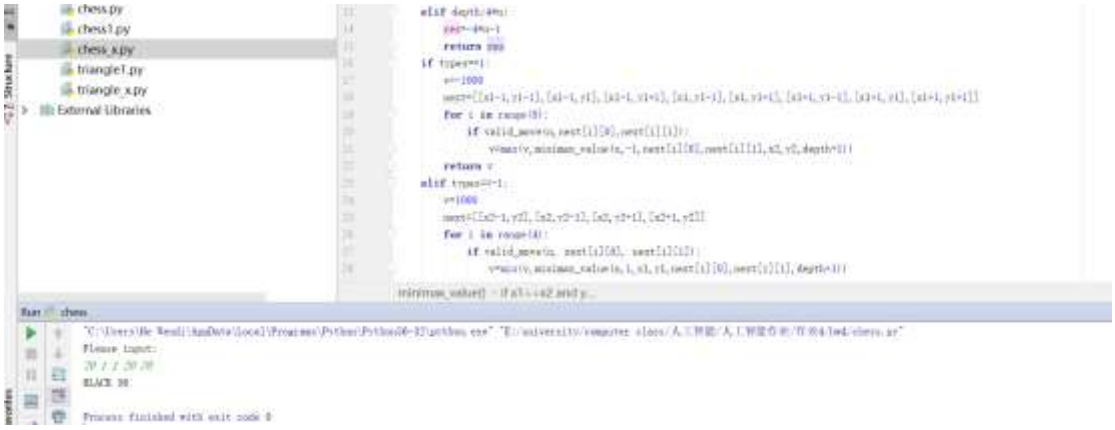
The main idea of the algorithm is the minimax value computing. There are 7 parameters in the minimax_value function, n for the size, types for max node or min node, x1, y1, x2, y2 for positions, depth for the current rounds.

In the recursion, for the max node (i.e. the black chess), it has 8 places to choose for the next step as it can move for either one grid or two grids. Besides, there will be a valid check function to make sure if the position is valid to reach. And it will recall the minimax_value function with the new position and the types will be opposite, as well as the depth will plus 1. It returns the max score of every possible next state.

The min node (i.e. the white chess) is similar. While when the function reaches the terminal state, which means that one chess eats the other one (i.e. r1=r2, c1=c2), the algorithm will return 30-depth for max node and -30-depth for min node. So, it encourages the black chess to win in as fewer rounds as possible while encourages the white chess to survive as longer as possible.

So, if the input satisfies the condition for the white to win, the output will show "WHITE 1". Otherwise, it will show the black chess win and how many rounds it takes.

The detail of the algorithms can be seen in the python file.



Result for 20 1 1 20 20

2.Traingle War

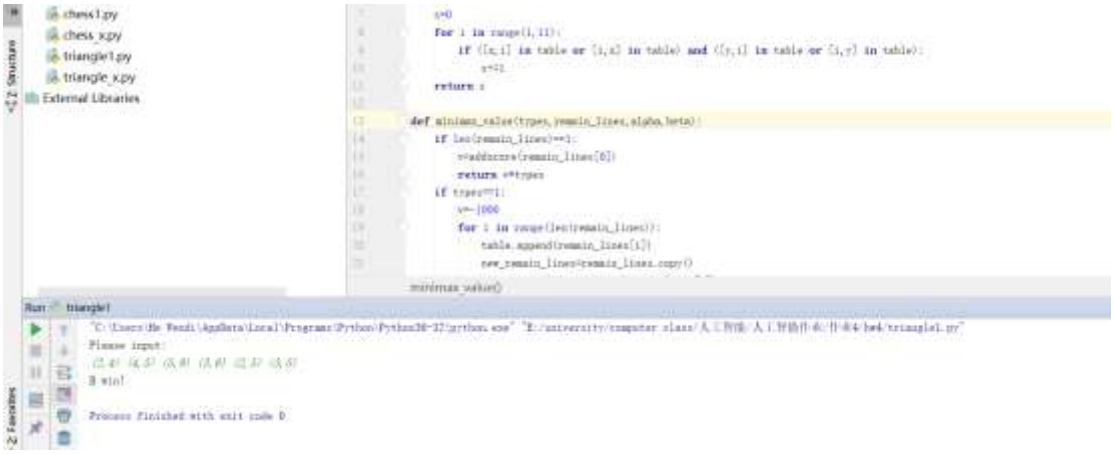
The main idea of the algorithm is the minimax value computing. Besides, we use the α−β pruning to optimise the search procedure. There are 4 parameters in the minimax_value function, types for max node or min node, remain_lines for the lines that are not drawn, alpha and beta for the value in α−β pruning. Besides, there is a global list edges to record all the edges and table to record the current drawn edges.

In the recursion, for the max node (i.e. the player A), the next state he could choose is all the edges still haven't been drawn yet. It will choose one of the to draw so consequently this edge will be removed from the remaining edges. The outcome this draw will be computed in the addscore function, maybe 0, 1 or 2. And it will recall the minimax_value function again with the new remaining set. It returns the max score of every possible next state. Besides, if a draw brings some outcome (which means this edge form a new triangle), then the player has to draw another edge. So, the flag will change to the opposite if such situation happens, indicating it is still the player's turn. Moreover, the α−β pruning will be applied here to accelerate the search procedure.

The min node (i.e. the player B) is similar. While when there is only one remaining edge, the algorithm will compute the value this draw can bring, and return value for max node and -value for min node. Since the player A wants the score to be as large as possible as he will win if the final score is positive and the player B wants the score to be as small as possible as he will win if the final score is negative, the player A will prefer the positive score and the player B will prefer the negative score. Otherwise if the final score is 0, it will be a draw for both sides.

So, if the final total result is positive, the output will show “A win!” and if the final total result is negative, the output will show “B win!”. Otherwise, if the final total result is 0, the output will show “Draw!”.

The detail of the algorithms can be seen in the python file.



Result for (2,4) (4,5) (5,9) (3,6) (2,5) (3,5)

Result for (1,2) (2,3) (1,3) (2,4) (2,5) (4,5)