Zehao Li
CS165A
Report

Architecture:

I have 2 class for this project.

Gobang class contains the main function. It will process the command line from the terminal. All the setting part are in this function. After the first setting settles down, it will create a game project which is the place to save all the information about that game.

Also, the human player and com player will play one by one. If a step is played by com, it will call amove() in game class. If a step is played by human, it will call emove() in game class. After that, it will print out the information as the instruction requires.

Game class contains all the ai part. It saves all the information of a game. For the member variables, it saves aic as com color and eic as human color. It also saves the size which is the size of the whole board. astep array list is to save all the points of computer. estep arraylist is to save all the points of human. Hashtable tt is for all the possible next moves of the computer and their heuristic values.

Basically, it has some common helper functions like print() used to print the current board. stringprocess() to change the string step into a point, pointprocess() to change the point step into a string, notout() to check if a point is out of range, existe() and exista() to check if the point is in estep or astep, notexist() to check if a position is empty.intt() is to check if the point is already in the possible next step list. iswinner() to check. findmaxinhash() it to caculate the point with the largest value in tt hashtable.

There are three parts of functions are the most important.

The first is emove(). Which is used to record the move of the human. It simply save the move into the estep arraylist and see if it will results in the win.

The second one is amove(). it is used to calculate the move of the computer. I will use minmax() function to get the min-max tree (it includes the a-pruning). In evaluate() called by minmax(), it uses payoff to calculate the pay off of a specific part of points, which ensures the max possibility of the win of the computer.

The third part is processvarious() processhelper() is the increase the point value by checking the 2-link, 3-link or 4-link situation. They will influence the point evaluation a lot because in some situations computer must go the specific move to let human not win.

Search:

In the amove(), I use the min-max tree and a-pruning to predict the next step.

First I will see if computer is the first step, if it’s the first step, it would go a random point in the middle part of the board. It’s not necessary to evaluate the whole empty board. If it’s not the first step, it could start it evaluation by initialize an empty tt hashtable , which saves all the possible next steps and call the minmax().

minmax() actually is an overall function. It will do a min-max tree which regards the human last step as a center of 9*9 square. Call the evaluate() function will start the construction. In evaluate(), I will have two loops for every point in 9*9 square. Each time I will assume (assumption A) a computer next step possibility, then I do another two for loops to assume (assumption H) a human step after the computer one. At that time, I will calculate the payoff of that assumption A. If it’s smaller than the min payoff, I will put the A and this payoff into the tt. If it’s bigger than the mini payoff, I will not consider this value. In this way, I implement the 2 depth min-max tree.

In that way, I use the a-pruning the save space and time. In the evaluate() function, I use the maxvalue and boolean need as two flags for a-pruning method. If after the first level2 branch, I will compare each level2 node payoff with the maxvalue. If it’s smaller, then it’s not necessary for me to do the other nodes in this branch.

Then I will do another min-max tree which regards computer’s last step as the center.

After the evaluate(), I will have a hash table of the possible next steps and their mini payoff. Then I use the processvarious() to process the special situation.

In processvarious(), I will loop the each point in the board and increase the value of the point by defining 2-link 3-link 4-link situation. They will be given different increase. In that way, the the payoff and the increase value will be two elements which influence the next step.

Challenges
1, I did not consider the special situations like 2-link, 3link at first. It’s really difficult for me to decide which place to put the process functions for the special situations and how to add the weight to these points. As a result, I write another part function —— processvarious() to add weight for specifically needed points.
2, It’s hard to define the heuristic function and payoff value function for the point. I use the methods on the ppt to let win number computer may achieve minus win numbers human may achieve.
3, In the heuristic and payoff value function, it’s difficult for me to define the size of the area i need to calculate. It I need to do the whole board evaluation it will be too long time. I decide to define the evaluation size in 9*9 which has the evaluation points as the center. It could consider all the win possibility including that evaluation point.

Weaknesses
1, As I said in the challenges, the special situations like 2-link, 3-link are really difficult to include all. I may lose some situations so it may has the wrong step. Even for the 3 link situation, it may have some exceptions to increase or decrease the point values.
2, heuristic and payoff value functions are not perfect enough to value the weight of a point. I don't know if the size i define is small or perfect. That element may influence the moves a lot.