Report for Project Go

胡与兴

11510225

1. Preliminaries

The go project basically only uses the breadth first search algorithm, and that is enough to finish the entire code by call the algorithm for several times. As for coding software, I use sublime text with terminate to execute the program, and PyCharm for debugging.

2. Methodology

The given code already has the plotting method and that give us perfect picture of the go chessboard, which saved us a lot of time. For the first question, I only need to finish the is_alive method. I add the first chess point into a queue (I import the Queue [1] from queue), when the point is checked or is not the color of first point, I just ignore it, and add the four points around this point into the queue, when there is an empty point, which indicate the chess is alive, the loop will stop and send back the data. For the second question, I use the is_alive and go_judge as well, but make a little update: also make them transfer the not alive list, this is used for print the new chessboard, which needs me send the coordinate to remove the white chess. And for the last question, is pretty simple and just needs me to try all the points in order to find the right place to put the white chess. And after calculating, I use write to make sure all the result will send to a new txt file[2].

3. Empirical Verification

About verification, of course the only way to test that is to design a new chessboard and mark some points manually, this might not conclude all the situation, but there is no need to do that. I somehow find some chessboard on line but those board are far too complicated, and all of them has 19 lines which is far more than I need. And also, the performance is close to Ω(N^2), that is because I don't use much strategy to improve the performance, and there's no need to. The reason I said that is because N = the lines on the chessboard, and we all know

that N is constant on most of go chessboard, which makes the performance comes to O(c), which is pretty perfect. And all the experimental results are reasonable and meet my expectation, by using the inset python IDE[3]. That maybe because of the program I design use the strategy of exhaustion, cover all the points on the board.

4. References

[1] Docs.python.org. (2017). 18.5.8. Queues — Python 3.6.3 documentation. [online] Available at: https://docs.python.org/3/library/asyncio-queue.html [Accessed 13 Oct. 2017].

[2] Docs.python.org. (2017). 11. File and Directory Access— Python 3.6.3 documentation. [online] Available at: https://docs.python.org/3/library/filesys.html [Accessed 13 Oct. 2017].

[3] Docs.python.org. (2017). 27.3. The Python Debugger— Python 3.6.3 documentation. [online] Available at: https://docs.python.org/3/library/pdb.html [Accessed 13 Oct. 2017].