Chris Swenson

CSC 180

10/21/17

Homework 2 - "Hi-YA!" AI

1. **Program Name:** Sockfish
2. A Pre-compiled executable "Sockfish.exe" is included in the same directory as this text-file.
   1. **Run:** Double-click Sockfish.exe
   2. *(Optional Run):* Double-click Sockfish\_d.bat (For specifying # of threads, TT memory size, and debug info)
   3. *(Compile):* To compile the program, go to “build/” and open “Sockfish.sln” with Microsoft Visual Studio. Then, set the build parameters to "Release" and "x64" and build.
   4. **Play:** Run the program and enter whether you want to go first with "y" or "n" (case-insensitive), then enter your move (case-insensitive, ex: "a1b1").
3. This program is written in C++.
4. Minimax (implemented as Negamax), Alpha-Beta, Iterative Deepening, Move Ordering (root move only), Transposition Table (using Zobrist Hashing), Multithreading, Bitboards & Magic Bitboards.
5. The AI typically searches up to 9 plies deep from the opening position, and up to 10 plies deep later on in the game. When a winning move is found, only plies up to the win are needed.
6. The AI has two factors in its terminal evaluation function.
   1. First, all piece types have a pre-set value. During terminal evaluation, the sum of the number of each piece multiplied by its piece type value is returned. White pieces have a positive value, while red pieces had a negative value. If there are the same number and type of pieces on both sides, the returned evaluation would be zero as the sides would cancel out.
   2. Second, all piece types have a pre-set value board. That is, for each piece, there is a board with different values for each tile. The sum of each piece multiplied by the value on the corresponding value board for the corresponding position of the piece is returned. Ninjas will receive a higher score on diagonal tiles that can attack the enemy king the closer they are to the enemy king. Samurais will receive a higher score the closer they are to the tile that will attack the enemy king.
7. I believe the AI is fairly strong as I cannot come close to beating it. Regardless of how strong the evaluation function may be, it can see 9 plies in the future, and therefore will not let me near its king unless I could see a perfect move 10 plies in the future. Although I have not played it against any other AI’s, I believe it will be able to outmaneuver any AI that can’t see 9 plies ahead. It doesn’t seem to play any obviously bad moves.
8. My program runs iterative deepening on all threads simultaneously, but each thread starts with a max search depth of one ply more than the previous thread. (**Lazy SMP**) This should theoretically increase the chance of one thread generating a transposition table entry of a searched branch that other branches can use to speed up their own searches.
9. There are no known fatal bugs in this version of the program (or illegal move bugs).
10. There are no known weak move bugs.
11. All requirements are satisfied by this program.