Programming project

Artificial Intelligence

The project has 4 parts. In each part you are asked to write two programs. The programming language can be C, C++, JAVA, Python, or anything else approved by the instructor/TA. All programs are related to Morris-Variant-B.

Part I: MINIMAX (45%)

Write two programs that get as input two file names for input and output board positions, and the depth of the tree that needs to be searched. The programs print a board position after White plays its best move, as determined by a MINIMAX search tree of the given depth and the static estimation function given in the Morris-Variant-B handout. That board position should also be written into the output file. In addition, the programs prints the number of positions evaluated by the static estimation function and the MINIMAX estimate for that move. The board position is given by a list of 21 letters. See the Morris-Variant-B handout for additional information.

First program: MiniMaxOpening

The first program plays a move in the opening phase of the game. We request that you name it **MiniMax-Opening**.

For example, the input can be:

(you type:)

board1.txt board2.txt 2

(the program replies:)

Input position: xxxxxxxxxWxxxxxxBxxxx Output position: xxxxxxxxxWxxWxxxBxxxx
Positions evaluated by static estimation: 9.
MINIMAX estimate: 9987.

Here it is assumed that the file board1.txt exists and its content is:

xxxxxxxxxWxxxxxxBxxxx

The file board2.txt is created by the program, and its content is:

xxxxxxxxxWxxWxxxBxxxx

(The position and the numbers above may not be correct. They are given just to illustrate the format.)

Please use the move generator and the static estimation function for the opening phase. You are not asked to verify that the position is, indeed, an opening position. You may also assume that this game never goes into the midgame phase.

Second program: MiniMaxGame

The second program plays in the midgame/endgame phase. We request that you call it **MiniMaxGame**. For example, the input can be:

(you type:)
    board3.txt board4.txt 3
(the program replies:)
    Input position: xxxxxxxxxxWWxWWxBBBxx Output position: xxxxxxxxWWWxWWxBBBBxx.
    Positions evaluated by static estimation: 125.
    MINIMAX estimate: 9987.

Here it is assumed that the file board3.txt exists and its content is:

xxxxxxxxxxWWxWWxBBBxx

The file board4.txt is created by the program, and its content is:

xxxxxxxxWWWxWWxBBBBxx

(The position and the numbers above may not be correct. They are given just to illustrate the format.)

Part II: ALPHA-BETA (35%)

In this part you are asked to write two program that behave exactly the same as the program of Part I, but implement the ALPHA-BETA pruning algorithm instead of the MINIMAX. Notice that these programs should return the exact same estimate values as the programs of Part I; the main difference is in the number of nodes that were evaluated. We request that you call these programs **ABOpening** and **ABGame**.

Part III: PLAY A GAME FOR BLACK(10%)

Write the same programs as in Part I, but the computed move should be Black's move instead of White's move. We request that you call these programs **MiniMaxOpeningBlack** and **MiniMaxGameBlack**.

Part IV: STATIC ESTIMATION (10%)

Write an improved static estimation function. The new function should be better than the one which was suggested in the handout. Rewrite the programs of Part I with your improved static estimation function. We request that you call these programs **MiniMaxOpeningImproved** and **MiniMaxGameImproved**.

Due date: to be announced.

You must be present when your project is tested. A schedule of available time slots will be provided later.

Submit a documented source code. This should include the source for all eight programs. Make sure that this includes source code and binaries.

Show examples of the program output when applied to several positions. Give at least two cases in which alpha-beta produces savings over MINIMAX.

Show at least two examples where your evaluation function produced different moves than the standard evaluation function. Write a short (one or two paragraphs) explanation of why you believe your function to be an improvement over the function proposed by the instructor.