Targeted Algorithm Analysis

The heuristic function of the game is *board\_score.* the time taken by that algorithm is varies depending upon the size of the board, number of pawns, bishop, knight, rook and king. But most of the time taken in terms of Big-O complexity is constant.

In term of the Big-O complexity, the function np.where is seem to take the most time, since it is written in C, so I assume np.where goes through whole array once. And np.where is used check where is the piece on the board, so the time complexity of np.where for the *board* array is,

Heigh of the board \* Width of the board

And in case when the king is captured the np.where functions is called twice, so the time complexity in that case would be,

2 \* (Heigh of the board \* Width of the board)

But constant terms are ignored, so the time complex will be:

Heigh of the board \* Width of the board

But since the heigh of the board is also constant, so the time complexity of np.where functions is,

8\* Width of the board

= Width of the board

And the next mosts time taking part of the code is going through each of the piece, and since there are maximum pawn in the game which can be converted into Queens (as per the assumption that pawns will convert into Queens only by AI), the time complexity of that will be 9 (maximum number of queens possible).