**Optional Challenges**

**If you’ve played around with this solver for a little bit, you’ll quickly come to the realization that the larger the boardgame, the longer the solution takes. This can be narrowed down to two culprits:**

1. **The generation of potential boards**
2. **The play through of each board**

**There are many ways of speeding up both of these sections. This completely optional challenge is to figure out how to go about speeding up this code. NOTE! If you do find some way of speeding this up, and are able to adequately explain how it works (don’t just copy things from online...), then you will also get bonus points on this assignment. Keep in mind though, if your base code is slow due to the algorithm/approach you took, and you speed it up using some method like parallelization, there may still be point reductions (and no bonus points given). So make sure you focus on the base assignment first before taking on this optional challenge!**

1. Generation of potential board

A close up of text on a white background

Description automatically generated

In our code, we checked all the combinations of the possible positions for the blocks. To speed up the code, we can check the positions only when they are on the path of the Lazor.  For example, on the board above, we only need to consider the combinations that include the shaded grids (1, 1), (3, 1), (3, 3), (5, 3), (5, 5), (7, 5) and (7, 7). After the first reflection/refraction, we also only consider the grids that are on the path of the reflected/refracted Lazor. This algorithm will reduce the run time significantly.

1. The play through of each board

We used ‘set’ and ‘dict’ as our data structure in our code. Unlike ‘list’, ‘set’ does not allow multiple occurrences of the same element and store values in order. Therefore, it is more efficient to use ‘set’. The time complexity for ‘set’ and ‘dict’ are mostly O(1), whereas the time complexity for ‘list’ are mostly O(n). Tables below show the time complexity for ‘list’, ‘set’, and ‘dict’ [1].

A close up of text on a white background

Description automatically generated

Table 1. Time complexity for lists.

A close up of text on a white background

Description automatically generated

Table 2. Time complexity for sets.

A close up of text on a white background

Description automatically generated

Table 3. Time complexity for dicts.

Reference:

[1] <https://www.ics.uci.edu/~pattis/ICS-33/lectures/complexitypython.txt>