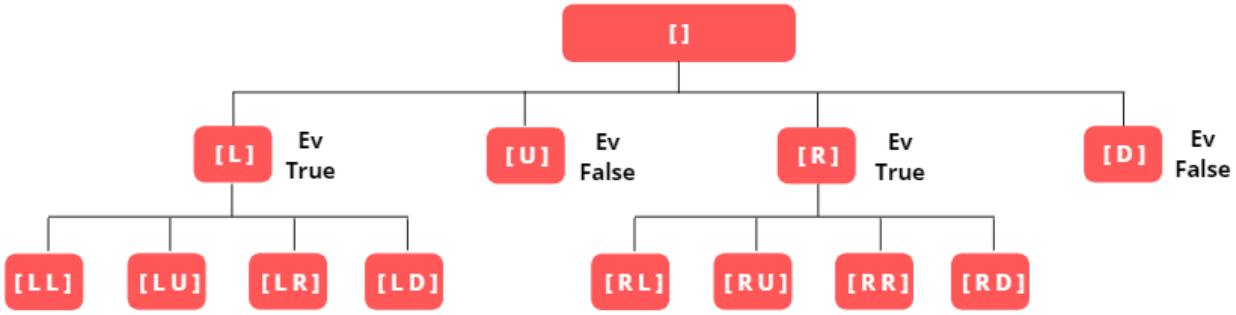
Alberto Rodero for Damavis Challenge

V1: Basic working implementation with recursive approach

V2: Optimization of evaluation function and trimmed approach

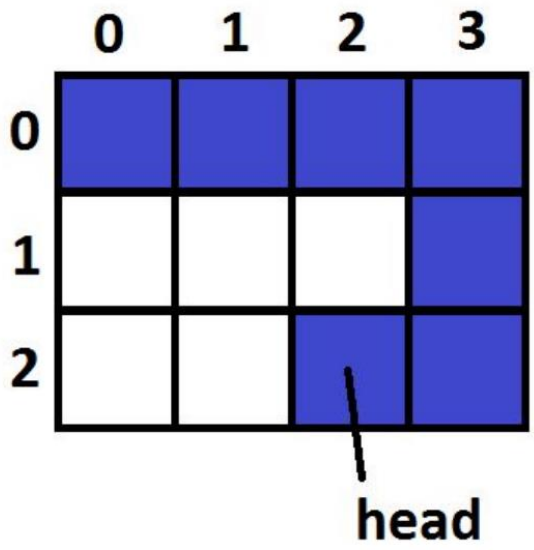
Function used:

- nOfPossibleSolutions: Initial call that returns the number of possible solutions, the solutions themselves and the number of evaluations for a certain depth, board and snake. Uses a recursive function
- nOfPossibleSolutionsRecursive: Based on a tree approach. Depth determines height of the tree, each node has 4 branches (one for each movement) and all leaves are either solutions or discarded solutions.

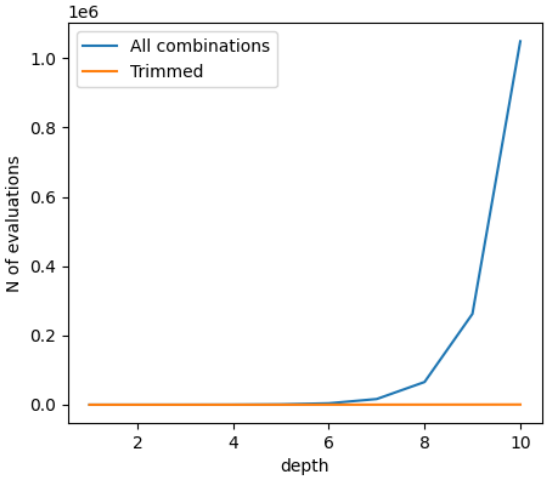


In this example we are not evaluating branches after movements U and D, we are trimming the number of evaluations a minimum of 50% of all possible combinations. In this way with a depth of 2 we are trimming at least 25% of all evaluations, for the snake is facing in one direction, so it can not move in the opposite direction it is facing.

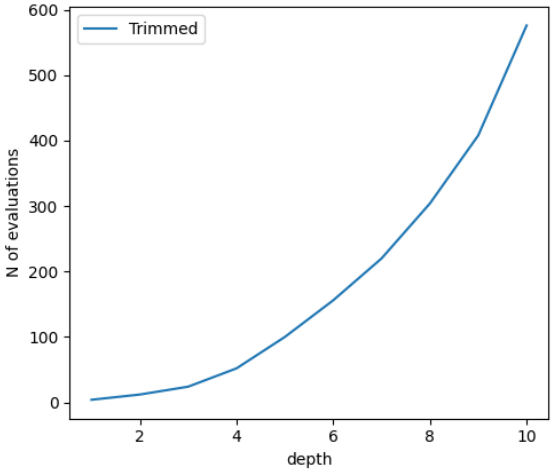
In this example it would be inefficient to evaluate all combinations starting with R, for it would be a collision with the snake body.
Although the function is using this concept it doesn't use a tree structure, for it is not computationally efficient.



The initial approach of all combinations follows a time cost order of O(n) a^n where a is the number of possible movements (constant) and d is depth (variable), 4^d.
The trimmed approach also follows an exponential time cost order with a much gentler slope.



In this graph we can see the comparison of the 2 approaches. From this perspective the trimmed approach looks almost linear because of the efficiency compared to the all combinations approach.



In this isolated graph we can see that it follow an exponential time cost order, but while in depth=10 the combination approach reaches more than 1 million evaluations the trimmed approach doesn't even reach the 600 evaluations.

- EvaluationFunction: Returns weather a set of movement is valid or not. Used by nOfPossibleSolutions