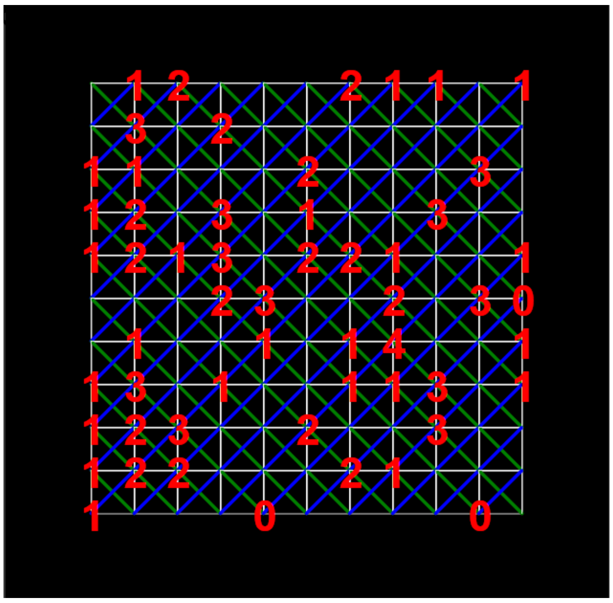
Homework 2

Ege Aktemur

April 2023

1. CSP Model

For every cell I create 2 booleans (left (green in the image)/right(blue in the image)). And I add constraint that both of them can not be equal.



Then I enumerate all of the numbered grid corners and add constrain that sum all of the lines that touches the corner must be summed to the number.
Example for the image for grid[0][1] (line_0_0_R + line_0_1_L) = 1

CSP vs A*

I think that answer of this question is that this is very liable on the both implementation and the case. First of all I want to emphasize that implementation of the A* would be hard because it is really hard to find a good admissible function for this problem. Secondly there are 2 implementation of the CSP solution first one is finding all cycles in the maze and restrict them using constraints. Second one is my algorithm which is creating new solutions while there is a cycle. I think first one is efficient for small and hard puzzles but performs bad on big mazes since it creates many unnecessary constraints. I can not compare A* to both since it is very based on the heuristic function and I could not found. To conclude I think that my solution would perform best in an average case.

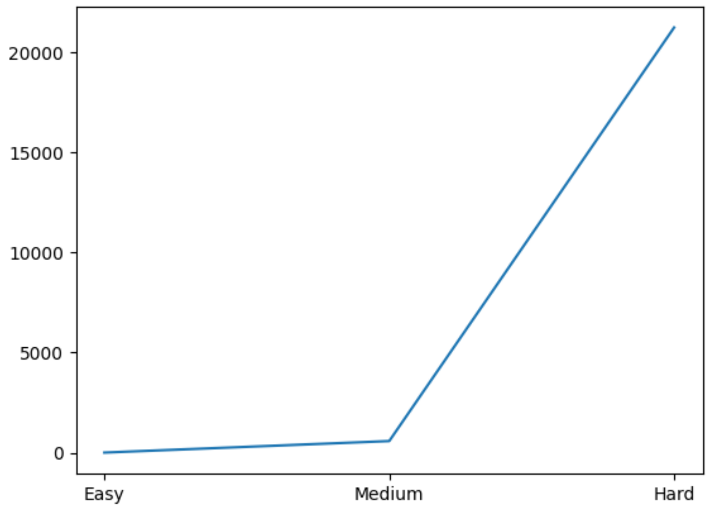
2. Implementation

Implementation is given in the Python files.

3. Evaluation

I created tests based on their size. I selected easy tests as puzzles sized 5x5, normal paths as 10x10, and hard tests as 12x12. I ran easy tests 1000 times, normal tests 100 times, and hard ones 10 times.

Variables	Constraints	Branches	Time (ms)	Average Time (ms)
50	4	0	4.38	
50	6	5	6.04	
50	6	10	6.53	5.16
50	14	51	5.84	
50	6	5	2.99	
200	28	4932	370.61	
200	56	6433	476.43	
200	58	15458	1708.22	580.32
200	28	1840	175.99	
200	42	1812	170.36	
288	68	55804	4823.96	
288	54	447743	36106.63	
288	52	2696	346.96	21237.62
288	72	3167090	45393.08	
288	64	200948	19517.48	



As Expected, these results indicate that the CSP-based method's performance decreases exponentially as the complexity of the puzzle increases. The time required to solve easy puzzles is much shorter than for middle or hard puzzles. As the difficulty level increases, the search space expands, leading to a more significant number of possible variable assignments to explore. The search process becomes more computationally demanding and takes longer to find a solution that meets all constraints. The scalability of the CSP-based method for solving the Slant puzzle can be considered limited, as the increase in time taken for solving harder puzzles is substantial. For easy puzzles, the method is quite efficient, but as the complexity of the puzzles increases, the method's performance degrades. This suggests that the method might struggle even more with larger and more complex puzzles. Lastly as a comment my algorithm recreates a solution while no solution found therefore order of the tried solution is very important as it can be seen in the difference of second and third hard puzzles.