# TDT4136 - Exercise 5

Per-Christian Berg
Filip Egge

## Task 2)

**Easy**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 8 | 5 | 9 | 6 | 3 | 2 | 7 | 1 | 4 |
| 1 | 2 | 6 | 7 | 4 | 9 | 5 | 3 | 8 |
| 7 | 3 | 4 | 1 | 8 | 5 | 6 | 9 | 2 |
| 6 | 7 | 5 | 8 | 2 | 1 | 3 | 4 | 9 |
| 9 | 4 | 2 | 5 | 6 | 3 | 1 | 8 | 7 |
| 3 | 1 | 8 | 9 | 7 | 4 | 2 | 5 | 6 |
| 2 | 9 | 3 | 4 | 1 | 7 | 8 | 6 | 5 |
| 5 | 8 | 1 | 2 | 9 | 6 | 4 | 7 | 3 |
| 4 | 6 | 7 | 3 | 5 | 8 | 9 | 2 | 1 |

**Medium**

| 6 | 4 | 9 | 5 | 8 | 7 | 2 | 1 | 3 |
|---|---|---|---|---|---|---|---|---|
| 3 | 8 | 7 | 2 | 4 | 1 | 9 | 6 | 5 |
| 5 | 2 | 1 | 9 | 6 | 3 | 4 | 7 | 8 |
| 9 | 1 | 3 | 7 | 2 | 5 | 8 | 4 | 6 |
| 2 | 6 | 8 | 1 | 9 | 4 | 5 | 3 | 7 |
| 7 | 5 | 4 | 6 | 3 | 8 | 1 | 2 | 9 |
| 1 | 9 | 2 | 3 | 5 | 6 | 7 | 8 | 4 |
| 4 | 7 | 6 | 8 | 1 | 9 | 3 | 5 | 2 |
| 8 | 3 | 5 | 4 | 7 | 2 | 6 | 9 | 1 |

**Hard**

| 8 | 1 | 5 | 7 | 4 | 2 | 3 | 9 | 6 |
|---|---|---|---|---|---|---|---|---|
| 9 | 3 | 7 | 1 | 6 | 8 | 5 | 4 | 2 |
| 2 | 4 | 6 | 5 | 3 | 9 | 1 | 7 | 8 |
| 3 | 8 | 4 | 6 | 5 | 7 | 2 | 1 | 9 |
| 5 | 7 | 9 | 2 | 1 | 4 | 8 | 6 | 3 |
| 1 | 6 | 2 | 9 | 8 | 3 | 4 | 5 | 7 |
| 7 | 5 | 3 | 4 | 2 | 6 | 9 | 8 | 1 |
| 6 | 2 | 1 | 8 | 9 | 5 | 7 | 3 | 4 |
| 4 | 9 | 8 | 3 | 7 | 1 | 6 | 2 | 5 |

**Very hard**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 6 | 9 | 5 | 7 | 2 | 1 | 8 | 4 |
| 4 | 5 | 1 | 3 | 6 | 8 | 7 | 9 | 2 |
| 7 | 8 | 2 | 4 | 9 | 1 | 6 | 3 | 5 |
| 9 | 4 | 5 | 2 | 1 | 3 | 8 | 7 | 6 |
| 6 | 2 | 7 | 9 | 8 | 5 | 3 | 4 | 1 |
| 1 | 3 | 8 | 7 | 4 | 6 | 2 | 5 | 9 |
| 8 | 9 | 4 | 6 | 2 | 7 | 5 | 1 | 3 |
| 2 | 7 | 3 | 1 | 5 | 9 | 4 | 6 | 8 |
| 5 | 1 | 6 | 8 | 3 | 4 | 9 | 2 | 7 |

## Task3)

**Easy**
Number of Backtrack 1
Number of Fails 0

Since this board has difficulty “easy” means that the board have a lot of completed squares. Therefore it is unnecessary to backtrack because the AC3 algorithm can solve the problem by itself. It has enough input or number to find all the constraints and by time delete variables in the domain that are redundant. The inference function is called by the backtracking_search function, and solves the board before the backtrack function is called again. We measure how many times the Backtrack function is called at the start of the function. This is the reason it will always run at least one time. The function call never goes further so it will have zero failures.

**Medium**
Number of Backtrack 1
Number of Fails 0

Same argument as above

**Hard**
Number of Backtrack 3
Number of Fails 0

In this solution we see that number of backtracks is 3. As mention before harder difficulties implies more empty squares, unfulfilled by numbers. So for this particular board its not enough to solve the puzzle only by AC3 algorithm. We implemented the backtracking function by selecting the unassigned variable with the fewest possible solutions. Because of that it will not take many guesses to come up with a solution. We clearly see that it guess right all the time therefore no failures.

**Veryhard**
Number of Backtrack 6
Number of Fails 3

We see on this board that taking the fewest possible solutions at first instance is not enough. When the board has no more options that can be solved by using the other numbers around it, the algorithm starts guessing on the allowed numbers for that square. And continues the solving the board until it can not go any further, or the board is solved. It will then backtrack and choose a different path.