# Task1-A

AC3 – Can only solve easy sudokus, to solve harder sudokus you need more complex inferences, like the naked triples.

Depth-first- takes too long time to find correct solution. Can get stuck in infinite branch.

Backtracking – using heuristics to significantly speed up the search and finds a good solution for all sudokus.

Min-conflicts – Does not work for sudokus. Can get stuck in local minima.

# Task 1-B

Both first unasigned variable and min remaining values with no inference fails solving the Sudoku. These cases are similar to the depth first seach. The algorithm gets stuck in an infinite tree. Thus some inference is needed to be able to solve the hardest Sudoku.

When using the forward checking inference both FUV and MRV are able to solve the puzzle relatively quickly. MRV does however perform better because it will search the tree more effectively.

# **Task 2-A**

## Depth first: n = 25

The size of the board is n^2, so the search space grows exponentially. The depth first search don’t use any heuristic and thus choses nodes poorly => problem will be solved, but it will take a very long time for n>25.

## Backtracking search (MRV+FC): n=300

Uses the same approach as DFS but inference and heuristic will speed up the search process significantly.

## AC-3: n = 0

Can’t solve even simplest of queen problems since this CSP can’t be made arc consistent.

## Min conflicts n= 8000

Works well for this kind of problem, since each iteration minimizes the problem, and it doesn’t get the same tree structure that the other algorithms get.

# Task 2-b

Backtracking with MRV and FC works the best for this problem since they makes use of a heuristic and inference. MRV is what significantly reduce the time complexity of the problem, since it works similarly to min-conflicts algorithm. It will place the queens where the domain is the smallest i.e. where there are the least amount of conflicts.

# Task 2-c

This depends on the size of the board aswell how close to a solution the original board is. For example when we tried n = 100:

1. 40 steps
2. 105 steps
3. 3 steps

For example when it only required 3 steps to solve, the puzzle was probably almost solved from the beginning.

# Task 2-d

The heuristic MRV uses the domain of the nodes which only exists in CSPs. Thus this heuristic works for all CSPs. All CSP heuristics thus works for all other CSPs. For traditional problem solving, the heuristic is designed to specifically to solve the relevant problem.