Assignment #5 TDT4136 - Introduction to Artificial Intelligence

Question 2

1)

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

Number of calls: 2
Number of failures: 0
Runtime: 0.062578 seconds
```

From these numbers we see that this problem is relatively easy (as we would hope). We only need to call the function twice. In reality, we do not need any backtracking (the number of failures is zero), so this problem could be solved by AC3 alone. This makes the runtime low.

2)

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

Number of calls: 10
Number of failures: 0
Runtime: 0.081233 seconds
```

The same comments as above can be applied here. No backtracking is performed, only applications of the AC3 algorithm.

3)
```
1 5 2 | 3 4 6 | 8 9 7
4 3 7 | 1 8 9 | 6 5 2
6 8 9 | 5 7 2 | 3 1 4
------+-------+------
8 2 1 | 6 3 7 | 9 4 5
5 4 3 | 8 9 1 | 7 2 6
9 7 6 | 4 2 5 | 1 8 3
------+-------+------
7 9 8 | 2 5 3 | 4 6 1
3 6 5 | 9 1 4 | 2 7 8
2 1 4 | 7 6 8 | 5 3 9

Number of calls: 28
Number of failures: 18
Runtime: 0.278831 seconds
```

Here we see some backtracking and therefore a slight increase in runtime. The number of total calls needed are still low, just slightly higher than the previous problem, but the number of times we have to backtrack is higher. We can think of this by formulating the tree consisting of partial assignment as nodes and value assignment to variables as the edges. In this way we see that we explore some 'unfruitful' branches (backtracking) before we reach a good one.

4)
```
4 3 1 | 8 6 7 | 9 2 5
6 5 2 | 4 9 1 | 3 8 7
8 9 7 | 5 3 2 | 1 6 4
------+-------+------
3 8 4 | 9 7 6 | 5 1 2
5 1 9 | 2 8 4 | 7 3 6
2 7 6 | 3 1 5 | 8 4 9
------+-------+------
9 4 3 | 7 2 8 | 6 5 1
7 6 5 | 1 4 3 | 2 9 8
1 2 8 | 6 5 9 | 4 7 3

Number of calls: 1780
Number of failures: 1758
Runtime: 14.601313 seconds
```

This is a more extreme version of the previous problem. We need a lot more calls to the function, but this could be due to the number of times we need to backtrack. Since backtracking is effectively a variant of DFS we could think that the algorithm explores the left-most branches first when the solution is to the far right. This leads to a lot of 'unfruitful' branches before we reach the solution.