30, October 2016

Assignment 5
Solving CSP

ALBERT Vincent
COOREVITS Sébastien

NTNU

Norwegian University of
Science and Technology

Table of contents

1. Sudoku solution

Here are our solutions for the 4 sudoku examples. For each, we checked that the solution was correct thanks to this website:
http://www.top-sudoku.com/sudoku/fr/rentrer-un-enonce-sudoku-solution.php

1.1 Easy

```
Solution for: sudoku easy

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

Backtrack function was called 1 time(s)
Backtrack function returned failure 0 time(s)
```

1.2 Medium

```
Solution for: sudoku medium

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

Backtrack function was called 3 time(s)
Backtrack function returned failure 0 time(s)
```

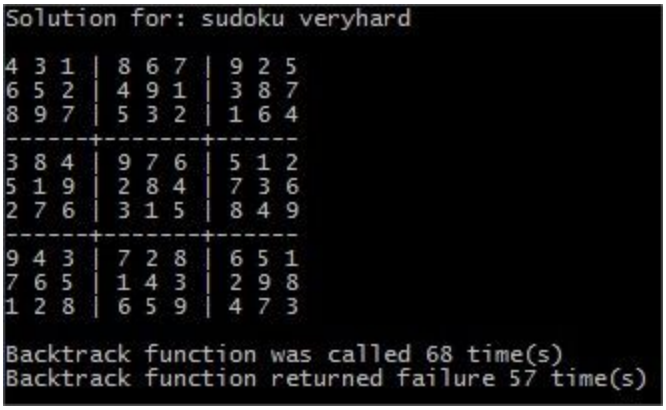
1.3 Hard

```
Solution for: sudoku hard

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

Backtrack function was called 12 time(s)
Backtrack function returned failure 4 time(s)
```

1.4 Very hard



2. Backtrack analysis

We can see that the tendency of the backtrack recursion evolution is to grow with the difficulty of the sudoku. It is normal because harder games provide less information regarding the constraints: there are fewer digits, and they are more dispersed. So we have more possible values in each domains. Because of that, the algorithm has to loop a lot and encounters more failures. And because of that, the part of failures also grows a lot.

Note: at the beginning, we made a mistake inside the revise() method. We put the condition to remove the value of i outside the two loops. So every time we went in this function, there was only maximum one value that was deleted. And the number of recursion and failures was enormous (10836 failures for 10892 calls for the very hard version), and to was the time the algorithm took. But we saw there was an error thanks to the map coloration problem. Indeed we knew since the guidance hour that the backtrack function shouldn't be called too many times, which wasn't the case here. So while we commented the code, we also searched for the problem and finally found it in this function. It is a good example on how a little error can affect the whole algorithm.