TD4136 - Introduction to Artificial Intelligence

Assignment 4: Solving Constraint Satisfaction Problems

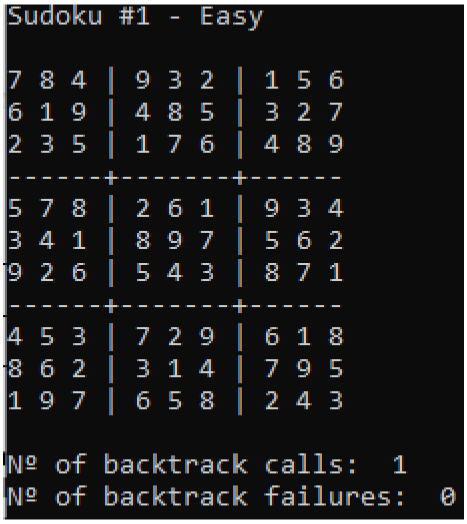
Pablo Zalla Quintanilla and Marc Soler Colomer
pabloaz - 567362 / marcso - 566522

Autumn 2021-2022
29/10/21

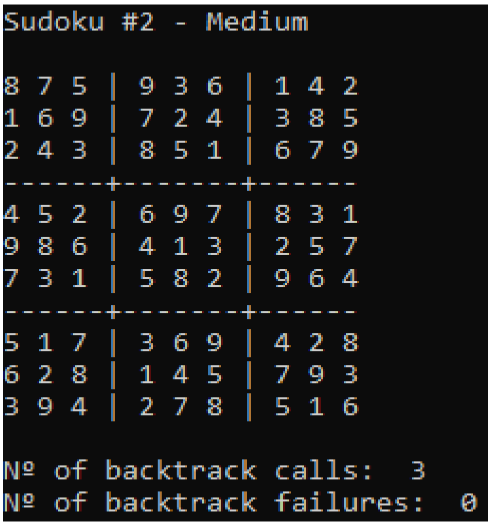
NTNU

Norwegian University of Science and Technology

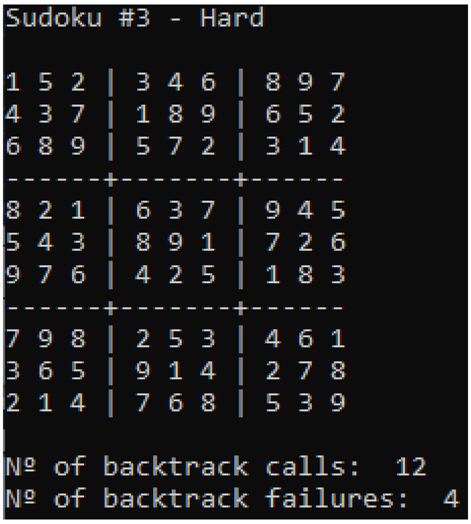
(a) Your program's solution for each of the four boards shown above.



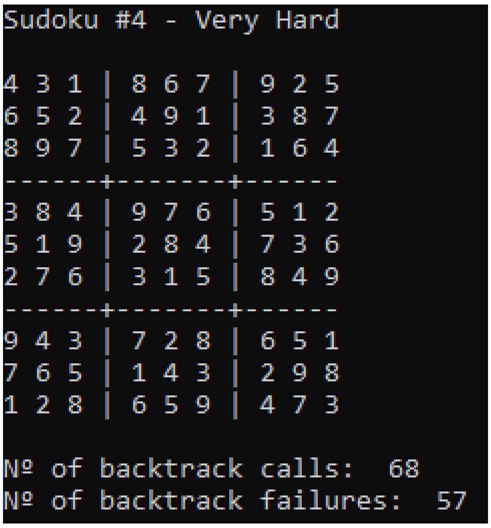
```
Sudoku #1 - Easy

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

Nº of backtrack calls:  1
Nº of backtrack failures:  0
```



```
Sudoku #2 - 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

Nº of backtrack calls:  3
Nº of backtrack failures:  0
```



```
Sudoku #3 - 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

Nº of backtrack calls:  12
Nº of backtrack failures:  4
```



```
Sudoku #4 - Very Hard

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

Nº of backtrack calls:  68
Nº of backtrack failures:  57
```

(b) The number of times your BACKTRACK function was called, and the number of times your BACKTRACK function returned failure, for each of the four boards shown above

Sudoku #1 - Easy:

Nº of backtrack calls: 1
Nº of backtrack failures: 0

Sudoku #2 - Medium:

Nº of backtrack calls: 3
Nº of backtrack failures: 0

Sudoku #3 - Hard:

Nº of backtrack calls: 12
Nº of backtrack failures: 4

Sudoku #4 - Very hard:

Nº of backtrack calls: 68
Nº of backtrack failures: 57

(c) Brief comments about the results in (b) for each of the four boards.

Once we check and analyze the results, we can see that the first and second sudoku's solutions are appropriate since the number of calls to our backtrack function is very low and we do not get any failure. But, if we take a look at the third and fourth one, we can see that the number of calls to the backtrack funcion and the number of failures, especially in the last one, has a notorious increase compared to the first and the second sudokus, as the difficulty of them has considerably increased.

This is because we did not implement any heuristic to order the cells and values that we choose, since the statement indicates that: "*it is sufficient to have SELECT-UNASSIGNED-VARIABLE return any unassigned variable, and to have ORDER-DOMAIN-VALUES return any ordering of the possible values for the given variable*"

We have simply tried with the first value, as we saw in the book, because it was the easiest implementation, but we are convinced that if we implement heuristic first, for the cell selection and then, for the value selection, we can get a better result.

For the cell, we would implement "Minimum-remaining-values heuristic (MRV)" in order to select the variable involved in the largest number of constraints, the "fail-first" value. We would do that because it is the most restricted variable and the most likely to cause failure soon. So we would make a list of the cells remaining and order them incrementally by the number of constraints they have, and choose from beginning to end.

For the value inside the cell, we would implement "Least-costing-value heuristic (LCV)" in order to select the value that rules out the fewest choices for the neighboring variables in the constraint graph, which means that we would try to choose the value to affect as little as possible to the next possible neighbour values, the "fail-last" value. So we would check the neighbours values and choose the one that restricts the less the future neighbours values.

These heuristics, that we saw in the book, represent what we would do in real life to solve a sudoku, firstly choose a cell that we consider that it can be easy to solve, and then search for possible values, considering the possible neighbours values.