paulpm / Paul Philip Mitchell

TDT4136 Assignment 5 – Constraint Satisfaction Problems

Comments

Included in this delivery is this report, a folder containing the four different boards and the sudoku.py script. The only changes from the original skeleton code is the implementation of the *backtrack, select_unassigned_variable, inference* and *revise* methods, as well as a *main* method and a few global variables. Explanation of the code is provided as comments in the source code.

I must say that I think you should continue providing skeleton code. I think many students, myself included, spend a lot of hours on implementation details in their respective programming languages, time that should rather be spent on logic and actual algorithms. Anyhow – great initiative providing skeleton code!

BOARD 1: EASY

```
Available boards are 0, 1, 2, 3. Please enter a value: 0
8 1 7 | 6 9 3 | 2 5 4
5 2 3 | 7 4 1 | 9 8 6
9 6 4 | 5 2 8 | 3 1 7
------+-------+------
6 7 1 | 8 5 9 | 4 2 3
3 4 8 | 2 6 7 | 1 9 5
2 9 5 | 1 3 4 | 7 6 8
------+-------+------
7 5 6 | 3 1 2 | 8 4 9
1 3 9 | 4 8 5 | 6 7 2
4 8 2 | 9 7 6 | 5 3 1
The backtrack function was called a total of 1 times.
The backtrack function failed a total of 0 times.

Process finished with exit code 0
```

Figure 1.1: Solution for the easy.txt board.

```
807690054
020041980
060020007
000809420
300207005
095104000
700010040
039480070
480076501
```

Figure 1.2: Original board.

As you can see, the backtrack function was called *one* time and failed *zero* times. This is a logically easy board for the program to solve.

BOARD 2: MEDIUM

```
Available boards are 0, 1, 2, 3. Please enter a value: 1
6 3 5 | 9 2 7 | 1 4 8
4 8 2 | 1 6 5 | 9 7 3
9 7 1 | 3 8 4 | 2 6 5
------+-------+------
5 2 9 | 7 1 6 | 3 8 4
8 4 6 | 2 9 3 | 5 1 7
7 1 3 | 5 4 8 | 6 9 2
------+-------+------
2 9 4 | 8 5 1 | 7 3 6
1 6 7 | 4 3 2 | 8 5 9
3 5 8 | 6 7 9 | 4 2 1
The backtrack function was called a total of 1 times.
The backtrack function failed a total of 0 times.

Process finished with exit code 0
```

Figure 2.2: Solution for the medium.txt board.

```
605900100
000100073
071300005
009010004
046293510
700040600
200001730
160002000
008009401
```

Figure 2.2: Original board.

The backtrack function was called only a single time for this board as well. In addition, like the previous board, the backtrack function does not fail a single time during execution.

BOARD 3: HARD

```
Available boards are 0, 1, 2, 3. Please enter a value: 2
8 9 2 | 3 5 1 | 7 6 4
1 3 4 | 8 7 6 | 5 2 9
5 7 6 | 4 9 2 | 3 1 8
------+-------+------
7 1 5 | 6 2 9 | 4 8 3
4 6 3 | 5 1 8 | 2 9 7
2 8 9 | 7 4 3 | 6 5 1
------+-------+------
3 5 1 | 2 8 4 | 9 7 6
9 4 7 | 1 6 5 | 8 3 2
6 2 8 | 9 3 7 | 1 4 5
The backtrack function was called a total of 5 times.
The backtrack function failed a total of 0 times.

Process finished with exit code 0
```

Figure 3.1: Solution for the hard.txt board.

```
090350700
000800029
000402008
710000000
463508297
000000051
300204000
940005000
008037040
```

Figure 3.2: Original board.

This board is noticeably more difficult. As you can see, for this particular execution the backtrack function was called *5 times* and failed *zero* times. The keywords here are *for this particular execution* – because there is some randomness involved (for instance in the *select_unassigned_variable* function), there will be different values for every execution. The second execution produced these values:

```
Available boards are 0, 1, 2, 3. Please enter a value: 2
8 9 2 | 3 5 1 | 7 6 4
1 3 4 | 8 7 6 | 5 2 9
5 7 6 | 4 9 2 | 3 1 8
------+-------+------
7 1 5 | 6 2 9 | 4 8 3
4 6 3 | 5 1 8 | 2 9 7
2 8 9 | 7 4 3 | 6 5 1
------+-------+------
3 5 1 | 2 8 4 | 9 7 6
9 4 7 | 1 6 5 | 8 3 2
6 2 8 | 9 3 7 | 1 4 5
The backtrack function was called a total of 8 times.
The backtrack function failed a total of 2 times.

Process finished with exit code 0
```

Figure 3.3: The second execution of the sudoku.py script for the "hard.txt" board.

As you can see, in this execution the backtrack function was called *8 times* and failed *2 times.*

BOARD 4: VERY HARD

```
Available boards are 0, 1, 2, 3. Please enter a value: 3
3 4 7 | 9 6 1 | 8 2 5
6 5 8 | 4 2 3 | 9 7 1
9 1 2 | 5 7 8 | 4 3 6
------+-------+------
5 3 4 | 2 9 7 | 6 1 8
7 6 9 | 1 8 4 | 2 5 3
2 8 1 | 3 5 6 | 7 9 4
------+-------+------
1 7 6 | 8 3 2 | 5 4 9
8 9 3 | 7 4 5 | 1 6 2
4 2 5 | 6 1 9 | 3 8 7
The backtrack function was called a total of 41 times.
The backtrack function failed a total of 31 times.

Process finished with exit code 0
```

Figure 4.1: Solution for the veryhard.txt board.

```
300001005
608023971
900070030
000200010
009000200
080006000
070030009
893740102
400600007
```

Figure 4.2: Original board.

Compared to the previous board, this configuration is even more difficult for the program to solve. The backtrack function is, in this execution, called a total of *41 times* and fails a total of *31 times*. However, because the same principles about randomness apply, the program will produce different values for each execution. As an example, the following is an image of the output for the second execution of the script on this boar

```
Available boards are 0, 1, 2, 3. Please enter a value: 3
3 4 7 | 9 6 1 | 8 2 5
6 5 8 | 4 2 3 | 9 7 1
9 1 2 | 5 7 8 | 4 3 6
------+-------+------
5 3 4 | 2 9 7 | 6 1 8
7 6 9 | 1 8 4 | 2 5 3
2 8 1 | 3 5 6 | 7 9 4
------+-------+------
1 7 6 | 8 3 2 | 5 4 9
8 9 3 | 7 4 5 | 1 6 2
4 2 5 | 6 1 9 | 3 8 7
The backtrack function was called a total of 29 times.
The backtrack function failed a total of 16 times.

Process finished with exit code 0
```

Figure 4.3: The second execution of the sudoku.py script for the "veryhard.txt" board.