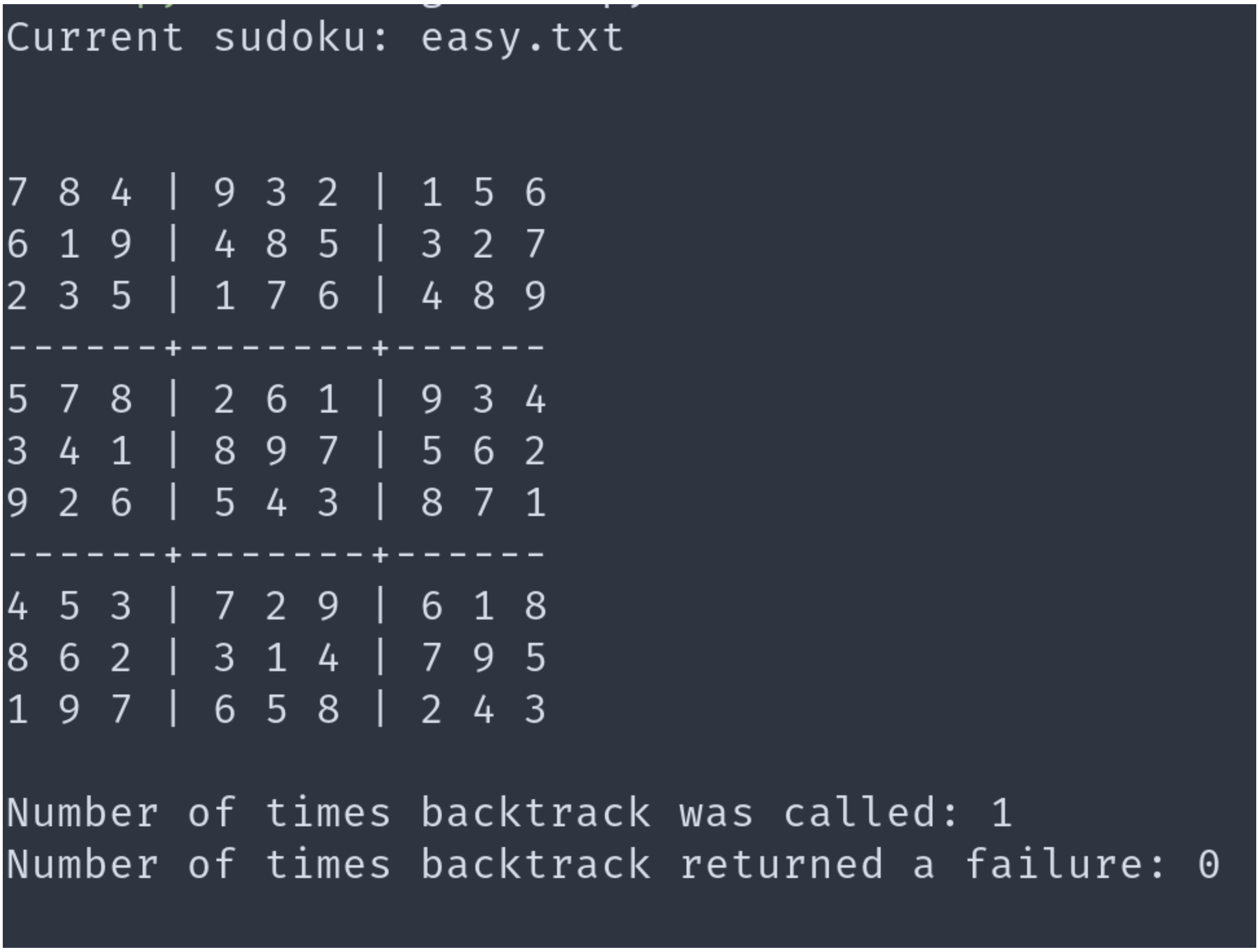
Exercise 4

Output from program (a and b)

easy.txt

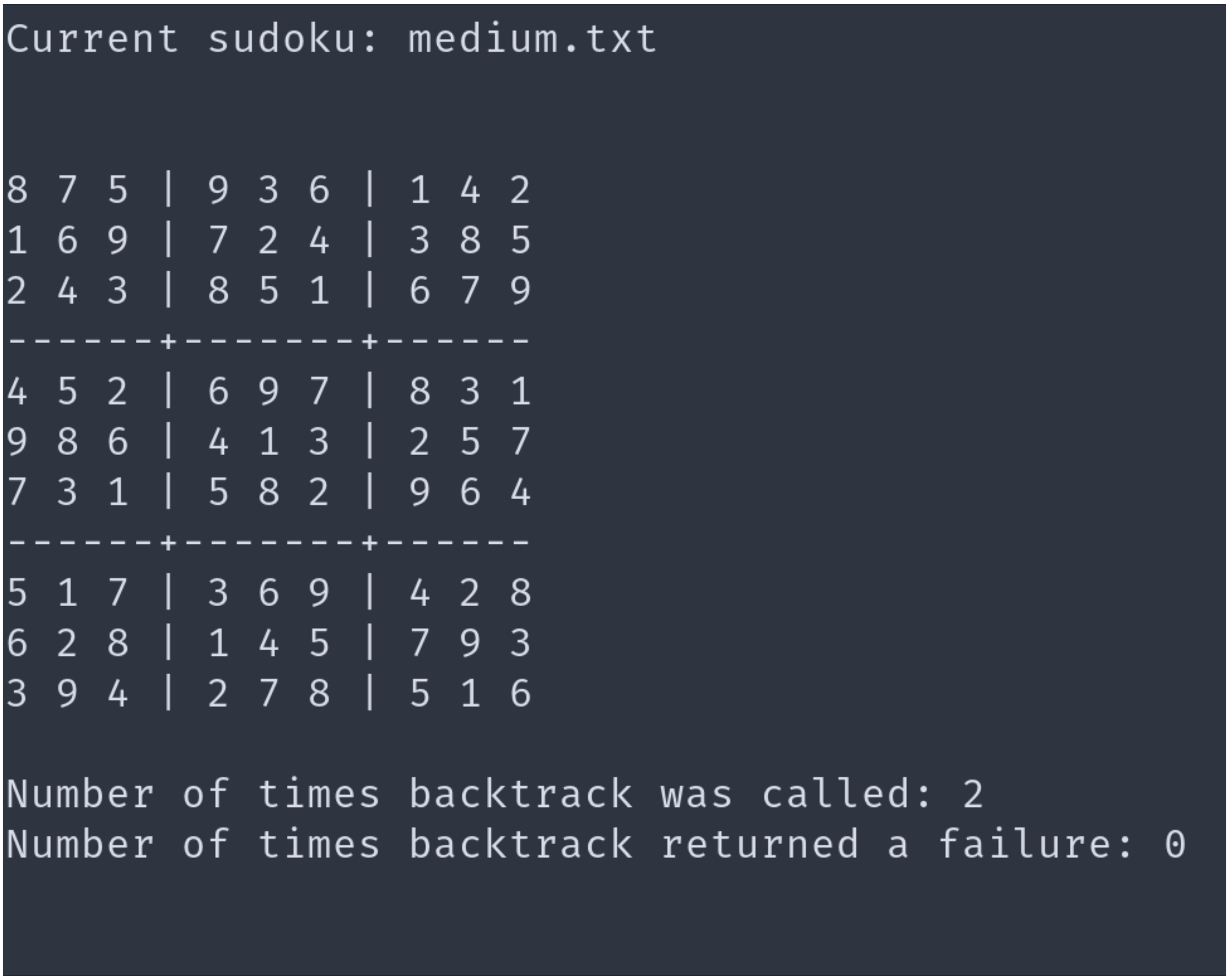


```
Current sudoku: easy.txt


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 times backtrack was called: 1
Number of times backtrack returned a failure: 0
```

medium.txt

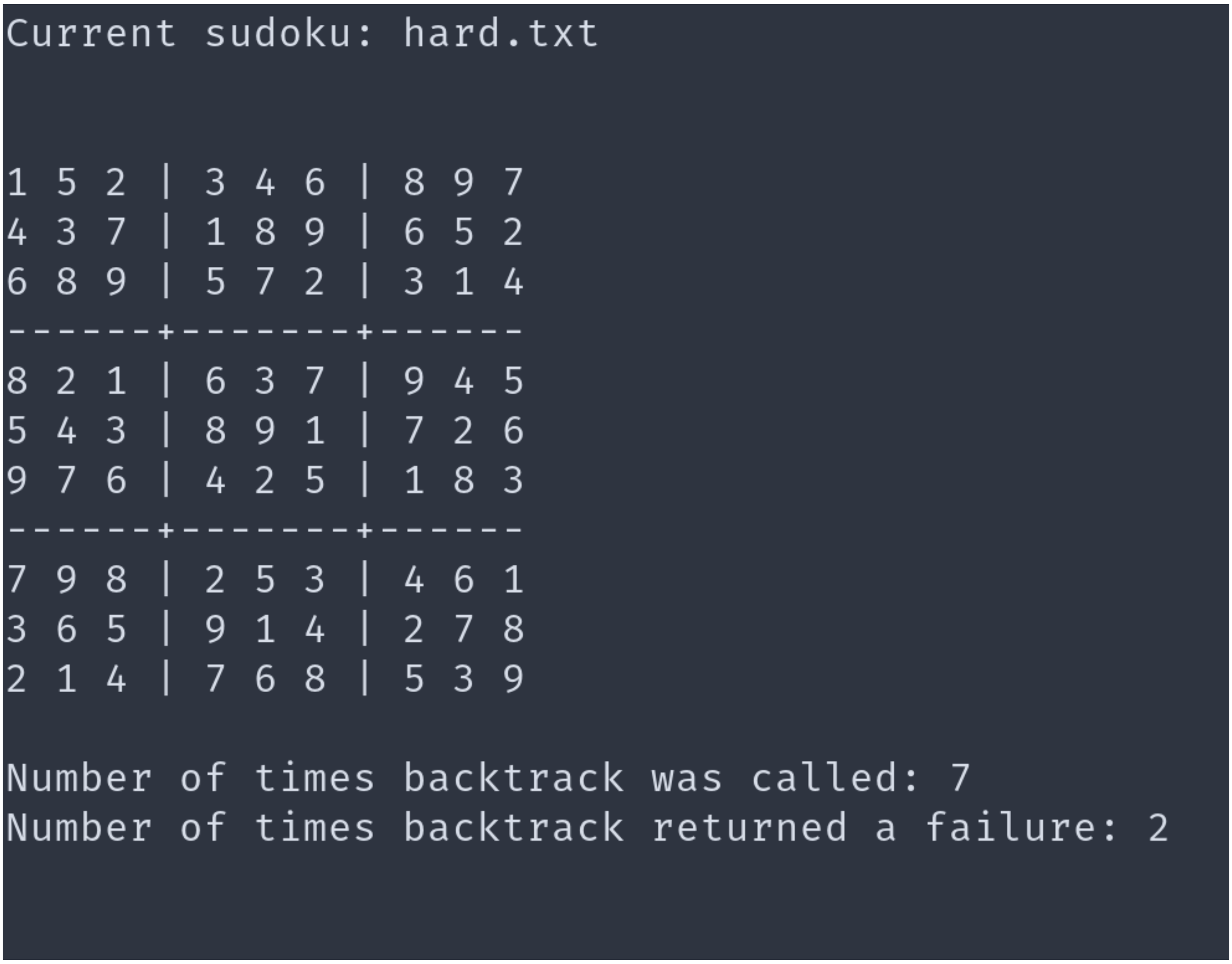


```
Current sudoku: medium.txt


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 times backtrack was called: 2
Number of times backtrack returned a failure: 0
```

hard.txt

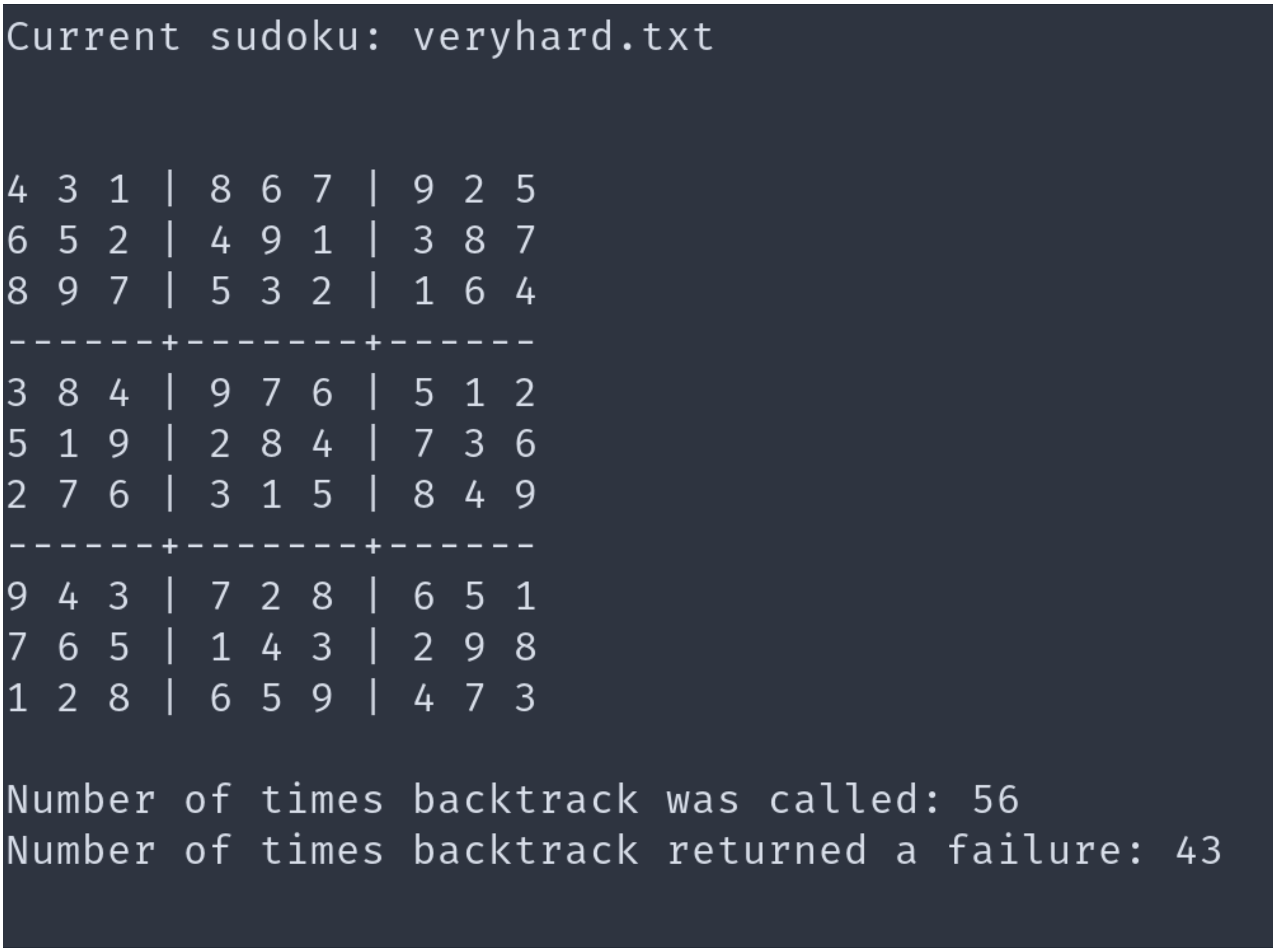


```
Current sudoku: hard.txt


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 times backtrack was called: 7
Number of times backtrack returned a failure: 2
```

veryhard.txt



```
Current sudoku: veryhard.txt


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 times backtrack was called: 56
Number of times backtrack returned a failure: 43
```

Reflection

In easy.txt backtrack is only called once, this is because the first inference call in `backtracking_search` solves the entire sudoku, since it's quite easy to infer all values when its easy.

For medium.txt we see that backtrack is called twice, meaning inference was called twice. Same as above, it's quite easy for a computer to infer values when we already have so many values.

For hard.txt we have a few more, but also here it's not a lot. A difference is that in this case, it actually tries values that gives a failure.

In veryhard.txt backtrack is called a lot more, and it also fails a lot more. This would probably be better if we had better domain value ordering, i.e choosing the value that rules out the fewest values for neighbors first. The inference could also be more optimized.

The difference between hard.txt and veryhard.txt is noticable. Perhaps for hard.txt we are "lucky" with the values we are choosing, resulting in few failures. Also with fewer values, it has to go through much more possibilities, as it's not linear.