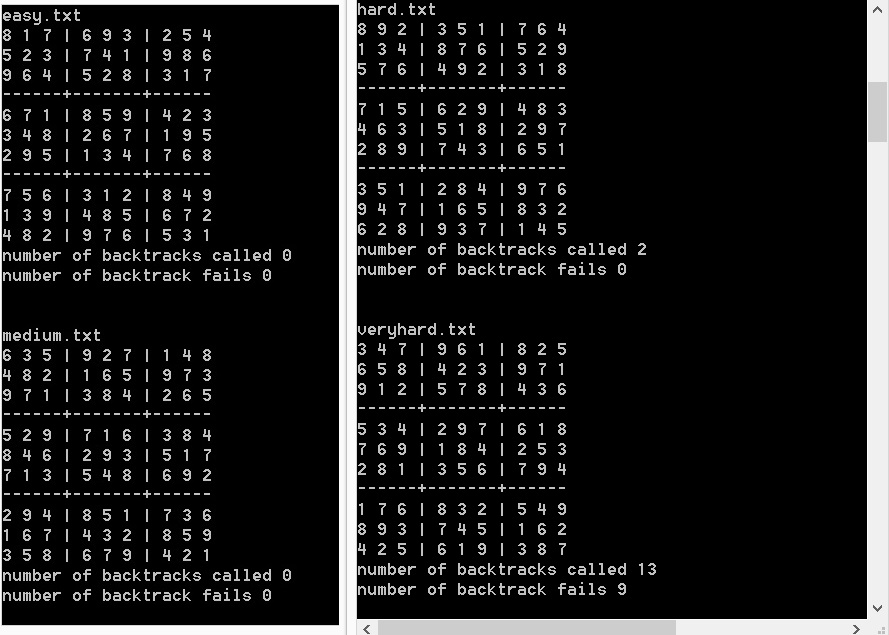
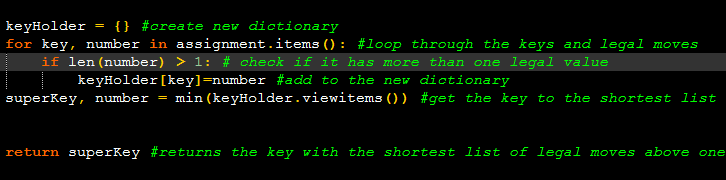
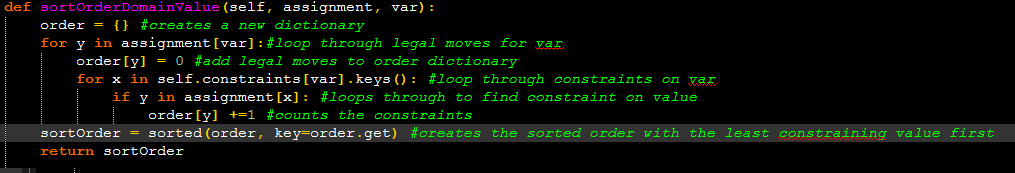
Solving Constraint Satisfaction Problems

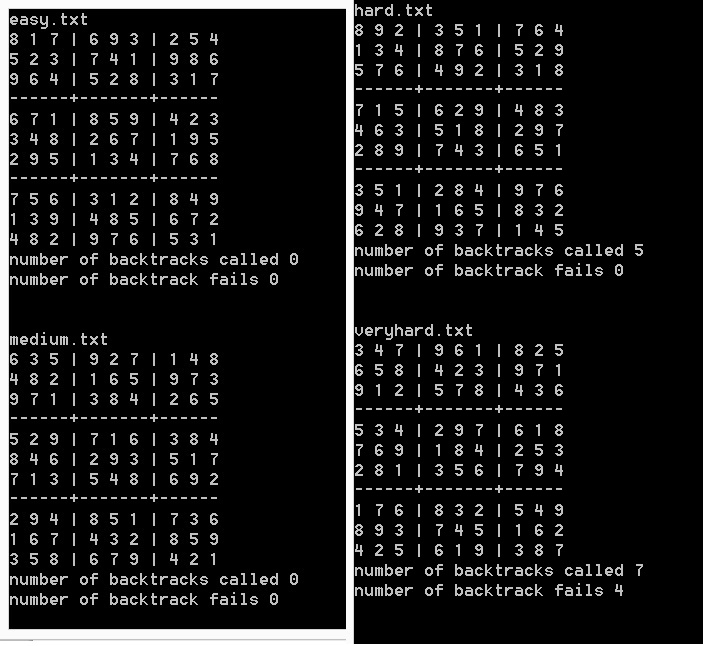
My first implementation of the code managed to solve both easy and medium without backtracks. For the hard it managed to do it with two backtracks and zero backtrack fails. For very hard it managed it in 13 backtracks where 9 failed. 

After reading through the book I was sure I could make some adjustments to make it even better.

I decided to take advantage of the hints in the book and change two functions, the select\_unassigned\_variable function and backtrack function.

First, the select\_unassigned\_variable function should return the name of one of the variables in assignment that hasn't been decided yet, that is whose list of legal values is greater than one. However, this is not necessarily the best choice as a list of legal values of length 2 will be faster to solve than one with 8 legal values. 

The second improvement that I implemented was a slight change to the backtrack function. Instead of looping for x in assignment[var] I chose to have a function return the sorted order with the least constraining value first to save time. 



The odd thing about these improvements is that for the hard level I get an increase in backtracks to 5, and for the very hard level I get a decent reduction of 6 backtracks, from 13 to 7, and 5 less backtrack fails from 9 to 4. I have seen others who have got less backtracks for both hard, and very hard but believe that the improvements are only for those levels as the main difference is which key with the shortest value above one that gets evaluated first.