WordokuSolver_Backtracking (n=9)

Time Complexity O(n^2dn) i.e., [n^2 variables * d (domain) * n(possibility to assign value)]

Space Complexity O(n^2)

	Total Clock Time (in sec)	**Search Clock time (in sec)**	**No of nodes generated**
Test Case 1 (input.txt)	0.124376058578491	0.124135494232178	7379
Test Case 2 (input1.txt)	0.892620325088501	0.891985177993774	51668
Test Case 3 (input2.txt)	0.340498685836792	0.340080976486206	20746
Test Case 4 (input3.txt)	0.135656118392944	0.135321855545044	7933
Test Case 5 (input4.txt)	0.008501291275024	0.008157968521118	386

WordokuSolver_minconflict (n=9)

Time Complexity O(itr*n^2) i.e., [maximum iteration (itr) *n^2(Goal test each iteration)]

Space Complexity O(n^2)

	Total Clock Time (in sec)	**Search Clock time (in sec)**	**No of nodes generated**
Test Case 1 (input.txt)	5.44241857528687	5.44093537330627	64539

Test Case 2 (input1.txt)	3.07246947288513	3.07098579406738	160150
Test Case 3 (input2.txt)	3.62516045570374	3.62370610237122	38797
Test Case 4 (input3.txt)	1.70624494552612	1.70461106300354	16022
Test Case 5 (input4.txt)	0.192917108535767	0.191366910934448	1422

Note:- Above time and space complexity is with respect to my program implementation

Analysis:-

CSP with backtracking is very fast and gives results for sure with respect to mini-conflict using CSP.

Mini-conflict advantage is that it is possible to get a result on the very first iteration as it requires a filled board and then modify the variables to get the best result. There is a possibility that a mini-conflict algorithm will get stuck in the local maxima. Thus it is very important to take some steps in the wrong direction in order to get to global maxima.

On an average, backtracking algorithm works faster compared to mini-conflict.

Behaviour expected:-
Yes, both backtracking and mini-conflict are able to solve any valid wordoku problem.
At the same time, mini-conflict takes longer time compared to CSP backtracking as it takes some extra time to get out of local maxima.