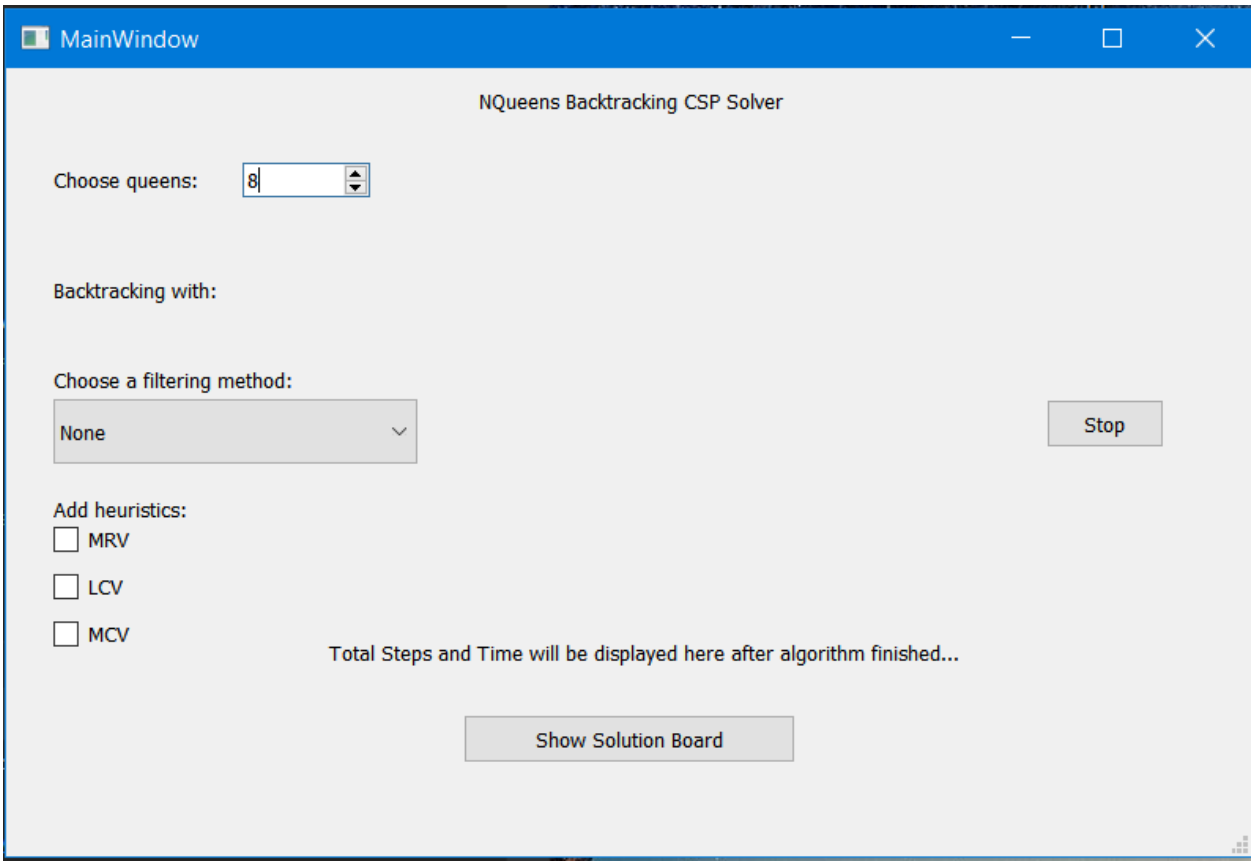
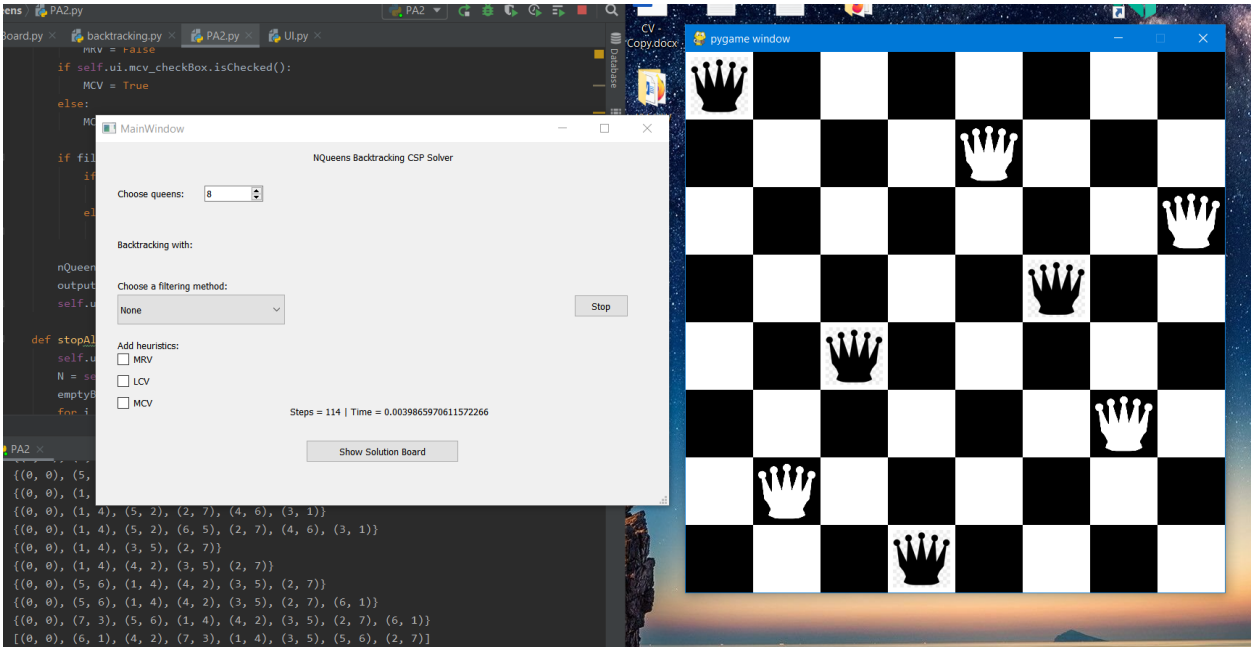
ICS 381 N Queens Programming Assignment 2

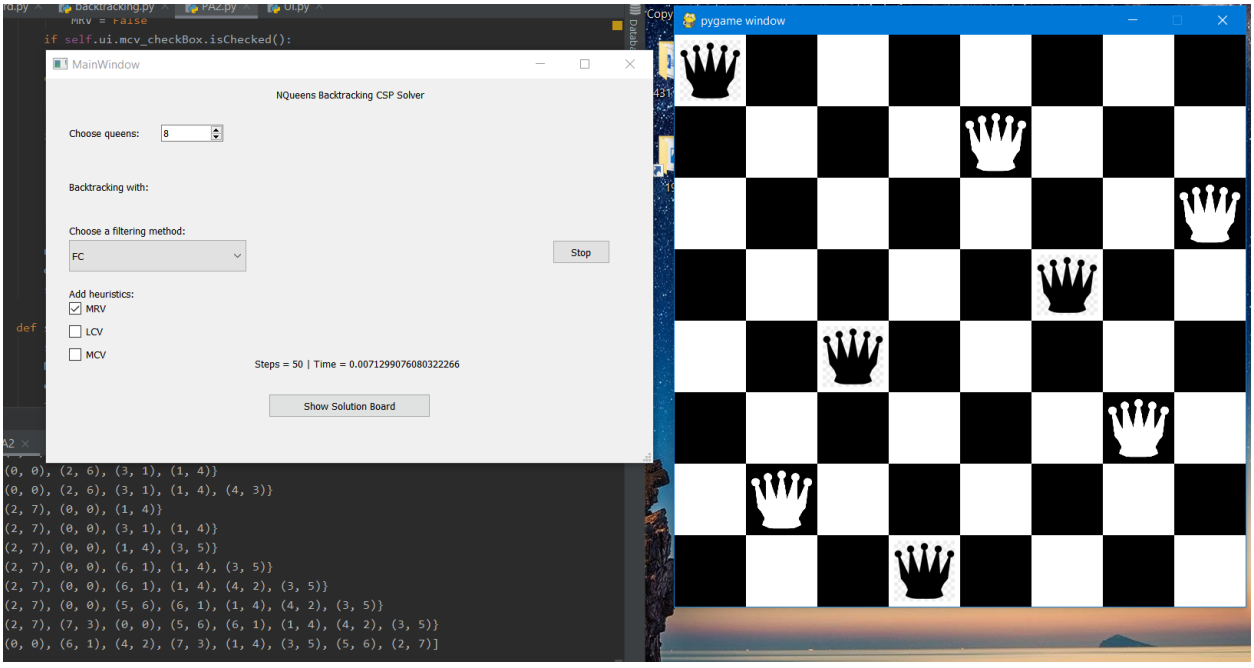
Farhan Mohammed Abdul Qadir 201771950

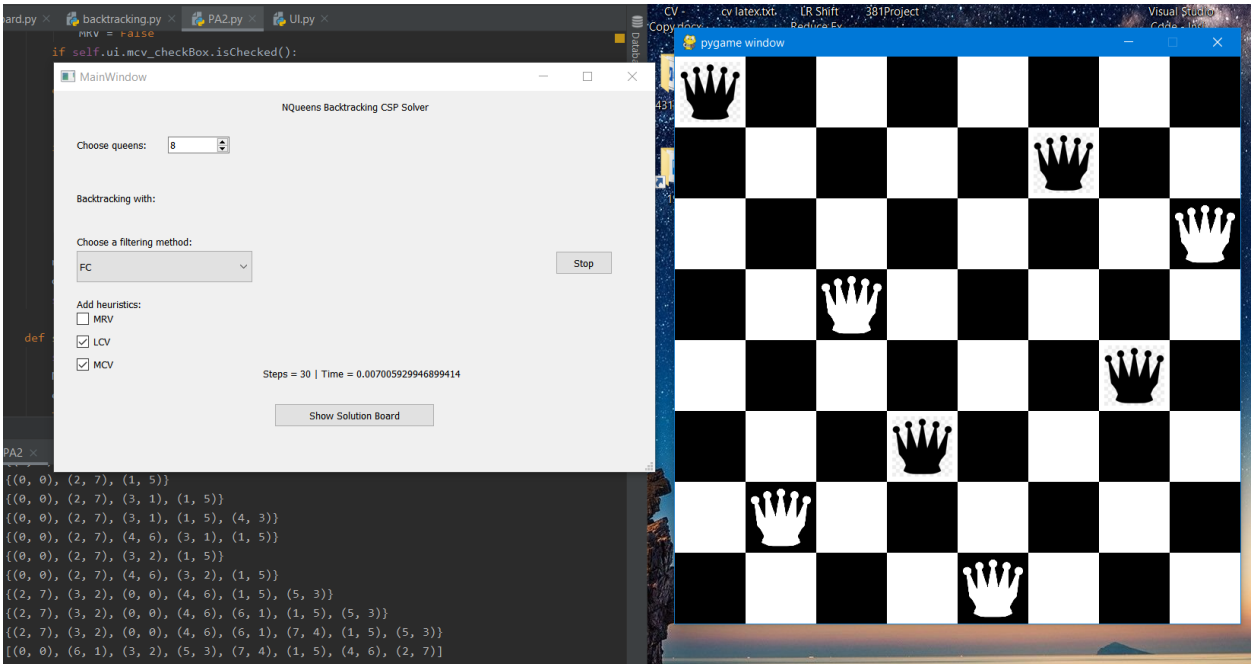
Sample Outputs

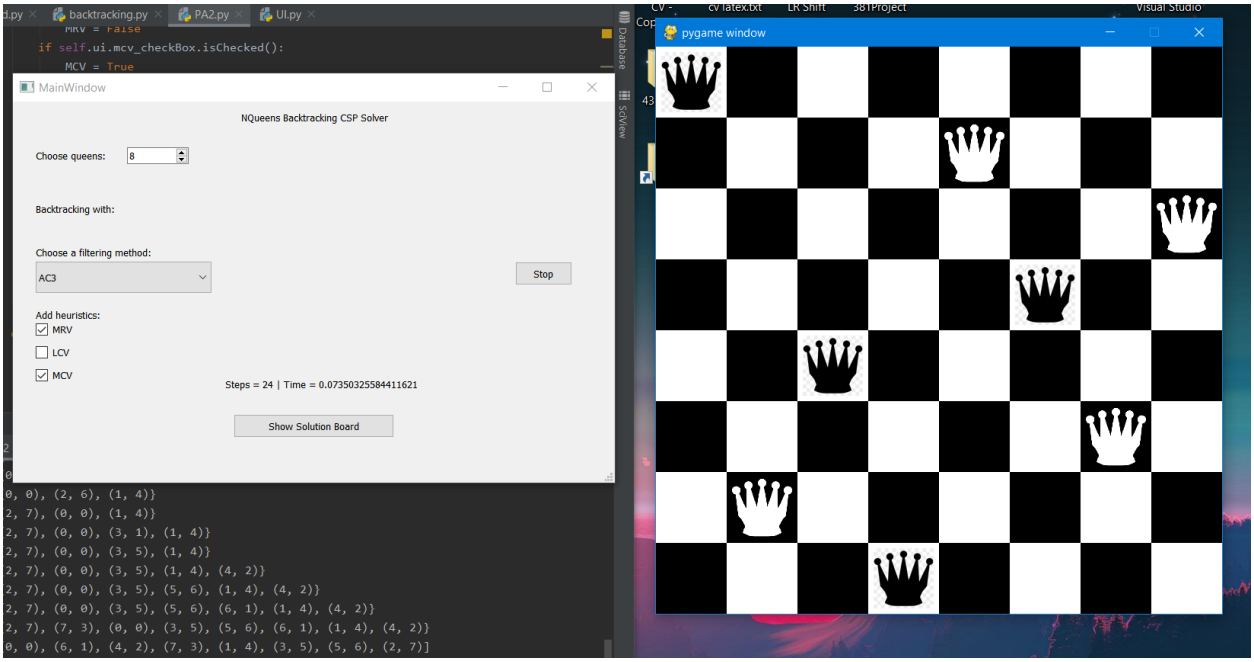
For several different scenarios, the parameters used for the sample outputs and the solution can be seen below. You can view the steps in the console and run the algorithm after choosing the inputs by clicking on the show solution button. I have not used threading so it is better to stick to lower values for the number of queens as the program will crash if other buttons are pressed while the algorithm is running. Note that the number of steps and time taken will be displayed on the main GUI window after the solution board window is closed.

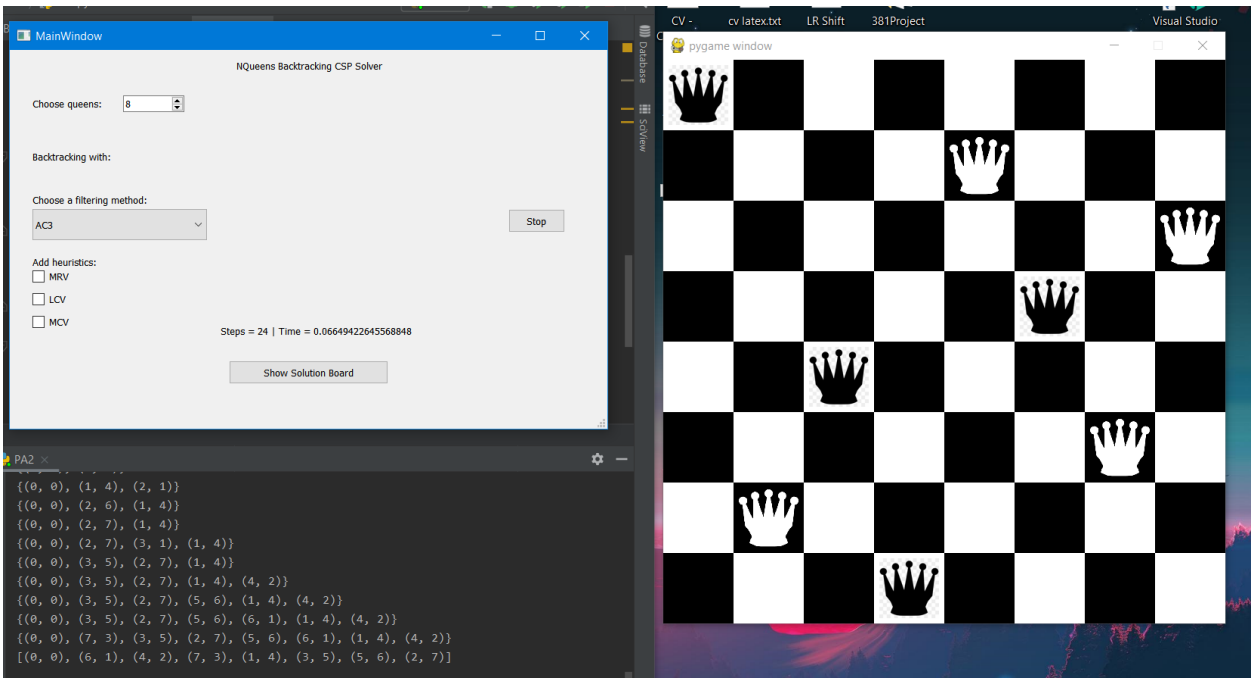


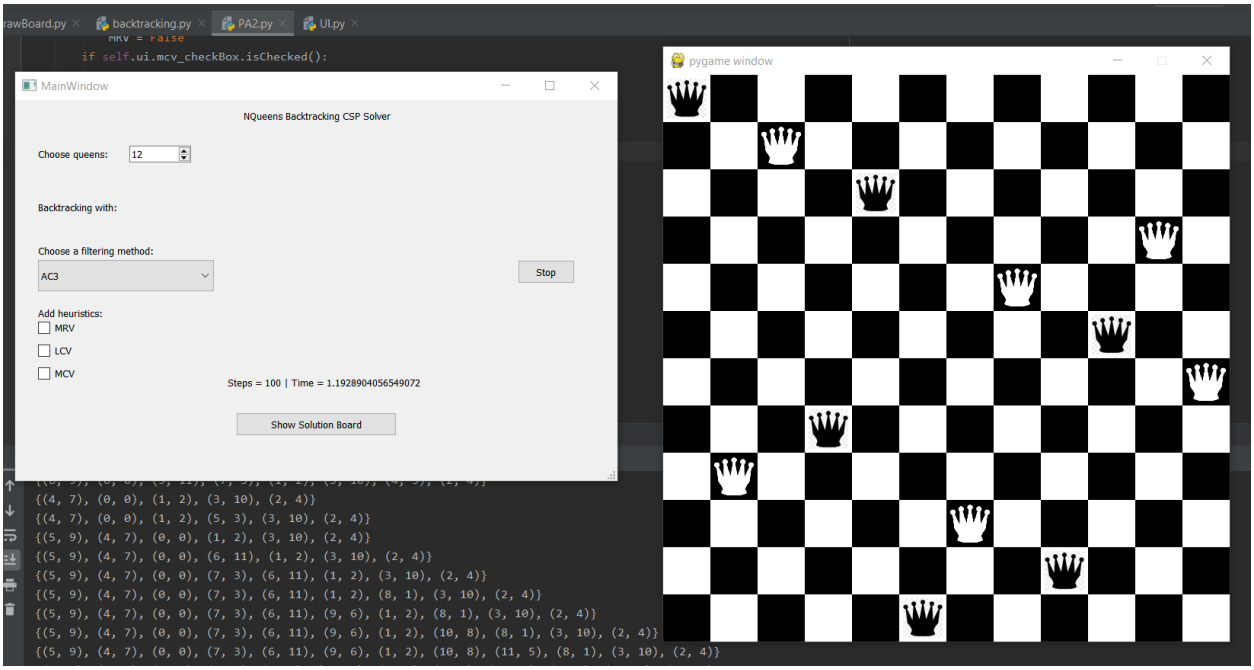

MainWindow
NQueens Backtracking CSP Solver
Choose queens:
8
Backtracking with:
Choose a filtering method:
None
Stop
Add heuristics:
MRV
LCV
MCV
Steps = 114 | Time = 0.0039865970611572266
Show Solution Board
pygame window

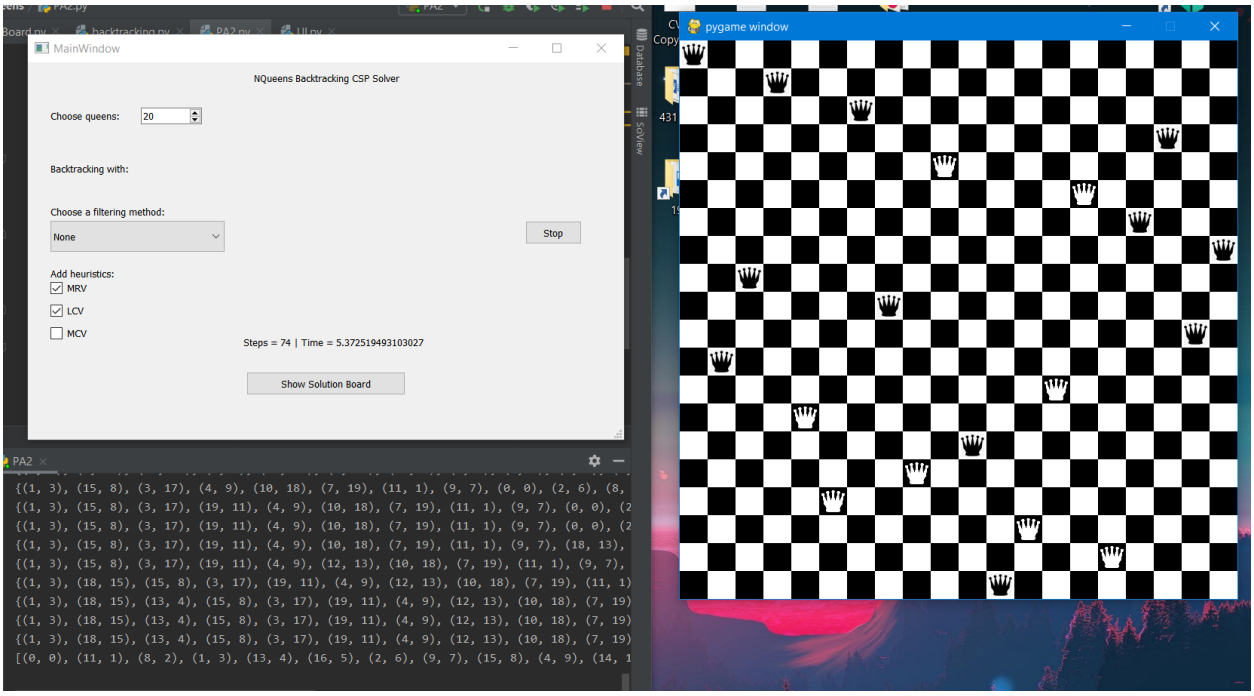

MainWindow
NQueens Backtracking CSP Solver
Choose queens:
8
Backtracking with:
Choose a filtering method:
FC
Stop
Add heuristics:
MRV
LCV
MCV
Steps = 50 | Time = 0.0071299076080322266
Show Solution Board
pygame window

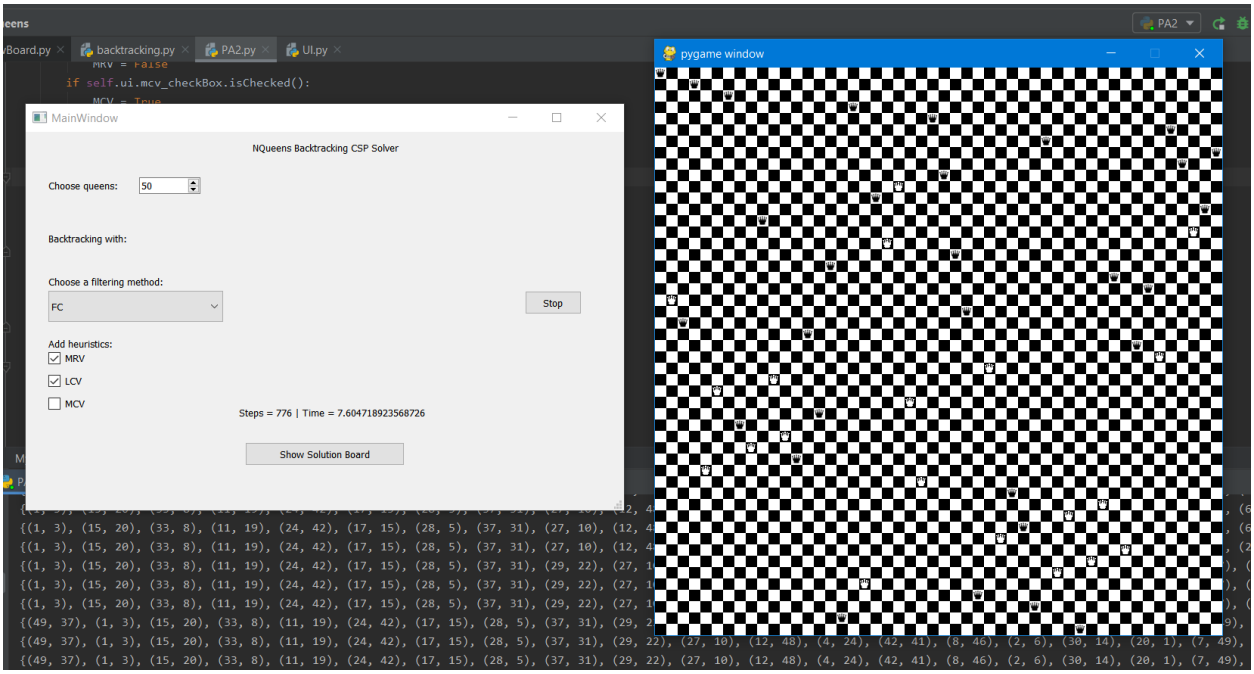

MainWindow
NQueens Backtracking CSP Solver
Choose queens:
8
Backtracking with:
Choose a filtering method:
FC
Stop
Add heuristics:
MRV
LCV
MCV
Steps = 30 | Time = 0.007005929946899414
Show Solution Board
pygame window
{(0, 0), (2, 7), (1, 5)}
{(0, 0), (2, 7), (3, 1), (1, 5)}
{(0, 0), (2, 7), (3, 1), (1, 5), (4, 3)}
{(0, 0), (2, 7), (4, 6), (3, 1), (1, 5)}
{(0, 0), (2, 7), (3, 2), (1, 5)}
{(0, 0), (2, 7), (4, 6), (3, 2), (1, 5)}
{(2, 7), (3, 2), (0, 0), (4, 6), (1, 5), (5, 3)}
{(2, 7), (3, 2), (0, 0), (4, 6), (6, 1), (1, 5), (5, 3)}
{(2, 7), (3, 2), (0, 0), (4, 6), (6, 1), (7, 4), (1, 5), (5, 3)}
[(0, 0), (6, 1), (3, 2), (5, 3), (7, 4), (1, 5), (4, 6), (2, 7)]


MainWindow
NQueens Backtracking CSP Solver
Choose queens:
8
Backtracking with:
Choose a filtering method:
AC3
Stop
Add heuristics:
MRV
LCV
MCV
Steps = 24 | Time = 0.07350325584411621
Show Solution Board
pygame window
0, 0), (2, 6), (1, 4)}
2, 7), (0, 0), (1, 4)}
2, 7), (0, 0), (3, 1), (1, 4)}
2, 7), (0, 0), (3, 5), (1, 4)}
2, 7), (0, 0), (3, 5), (1, 4), (4, 2)}
2, 7), (0, 0), (3, 5), (5, 6), (1, 4), (4, 2)}
2, 7), (0, 0), (3, 5), (5, 6), (6, 1), (1, 4), (4, 2)}
2, 7), (7, 3), (0, 0), (3, 5), (5, 6), (6, 1), (1, 4), (4, 2)}
0, 0), (6, 1), (4, 2), (7, 3), (1, 4), (3, 5), (5, 6), (2, 7)]


MainWindow
NQueens Backtracking CSP Solver
Choose queens:
8
Backtracking with:
Choose a filtering method:
AC3
Stop
Add heuristics:
MRV
LCV
MCV
Steps = 24 | Time = 0.0664942264556848
Show Solution Board
PA2
{(0, 0), (1, 4), (2, 1)}
{(0, 0), (2, 6), (1, 4)}
{(0, 0), (2, 7), (1, 4)}
{(0, 0), (2, 7), (3, 1), (1, 4)}
{(0, 0), (3, 5), (2, 7), (1, 4)}
{(0, 0), (3, 5), (2, 7), (1, 4), (4, 2)}
{(0, 0), (3, 5), (2, 7), (5, 6), (1, 4), (4, 2)}
{(0, 0), (3, 5), (2, 7), (5, 6), (6, 1), (1, 4), (4, 2)}
{(0, 0), (7, 3), (3, 5), (2, 7), (5, 6), (6, 1), (1, 4), (4, 2)}
[(0, 0), (6, 1), (4, 2), (7, 3), (1, 4), (3, 5), (5, 6), (2, 7)]
CV -
cv latex.txt
LR Shift
381Project
Visual Studio
pygame window


rawBoard.py
backtracking.py
PA2.py
UI.py
if self.ui.mcv_checkBox.isChecked():
MainWindow
NQueens Backtracking CSP Solver
Choose queens:
12
Backtracking with:
Choose a filtering method:
AC3
Stop
Add heuristics:
MRV
LCV
MCV
Steps = 100 | Time = 1.1928904056549072
Show Solution Board
{(4, 7), (0, 0), (1, 2), (3, 10), (2, 4)}
{(4, 7), (0, 0), (1, 2), (5, 3), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (1, 2), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (6, 11), (1, 2), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (7, 3), (6, 11), (1, 2), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (7, 3), (6, 11), (1, 2), (8, 1), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (7, 3), (6, 11), (9, 6), (1, 2), (8, 1), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (7, 3), (6, 11), (9, 6), (1, 2), (10, 8), (8, 1), (3, 10), (2, 4)}
{(5, 9), (4, 7), (0, 0), (7, 3), (6, 11), (9, 6), (1, 2), (10, 8), (11, 5), (8, 1), (3, 10), (2, 4)}
pygame window





In general, forward checking seems to be faster than simple backtracking and AC3 but AC3 takes the least number of steps usually. LCV takes a lot of time and steps as we consider placing queen in all rows for every column then choose the best one. MCV does not really improve the functioning of the algorithm for the N queens problem as the number of constraints for each remaining variable is always the same, as every variable(column) in a chess board is constrained with all other columns since placing a queen in one column affects the positions of queen placements in all other columns. It can be verified that filtering and heuristics greatly improve the performance of the simple backtracking algorithm.