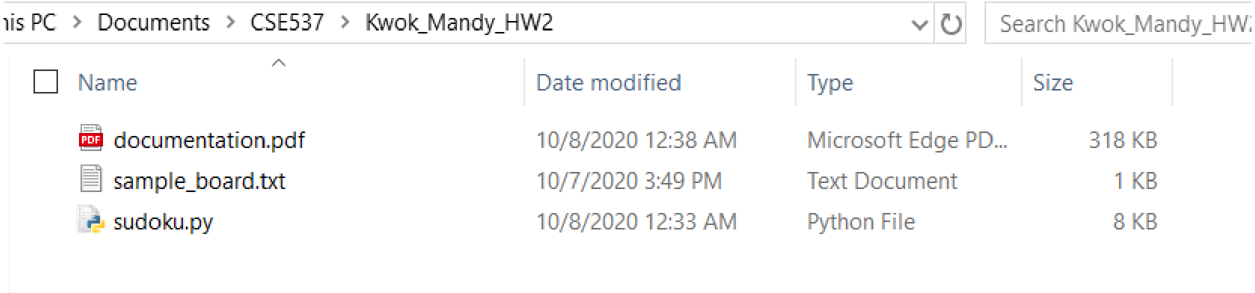
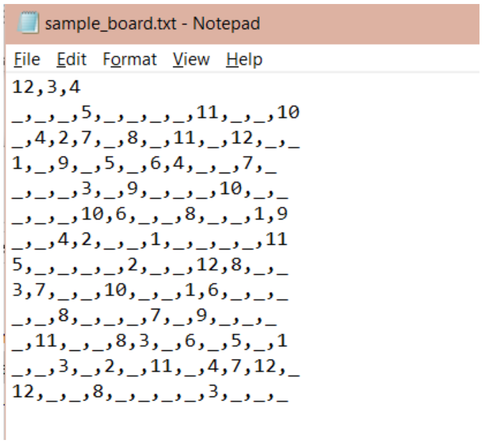
Mandy Kwok 10/9/20
111458972 CSE537 Artificial Intelligence

Assignment# 2

File Structure



Name	Date modified	Type	Size
documentation.pdf	10/8/2020 12:38 AM	Microsoft Edge PD...	318 KB
sample_board.txt	10/7/2020 3:49 PM	Text Document	1 KB
sudoku.py	10/8/2020 12:33 AM	Python File	8 KB



```
12,3,4
_,_,_,5,_,_,_,_,11,_,_,10
_,4,2,7,_,8,_,11,_,12,_,_
1,_,9,_,5,_,6,4,_,_,7,_
_,_,_,3,_,9,_,_,_,10,_,_
_,_,_,10,6,_,_,8,_,_,1,9
_,_,4,2,_,_,1,_,_,_,_,11
5,_,_,_,_,2,_,_,12,8,_,_
3,7,_,_,10,_,_,1,6,_,_,_
_,_,8,_,_,_,7,_,9,_,_,_
_,11,_,_,8,3,_,6,_,5,_,1
_,_,3,_,2,_,11,_,4,7,12,_
12,_,_,8,_,_,_,_,3,_,_,_
```

sudoku.py contains the python script file with all three implementations

sampleboard.txt contains one test set represented with numbers, comma, and underscore as specified in homework instruction

Compilation

After unzipping the file, navigate into directory Kwok_Mandy_HW2

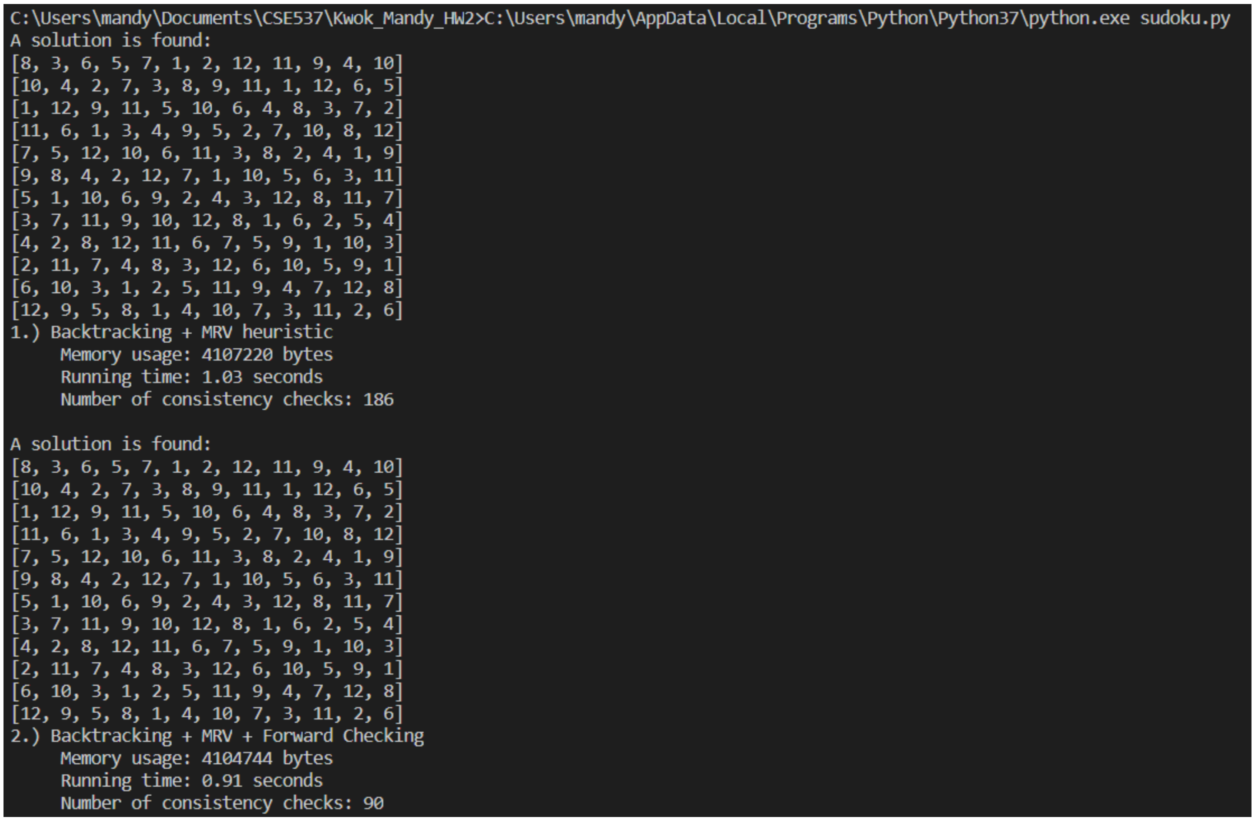
Open terminal in this directory

Install file dependencies if applicable (pip install guppy3, etc.)

Ensure the input file must called **sampleboard.txt** and placed within the same directory as **sudoku.py**

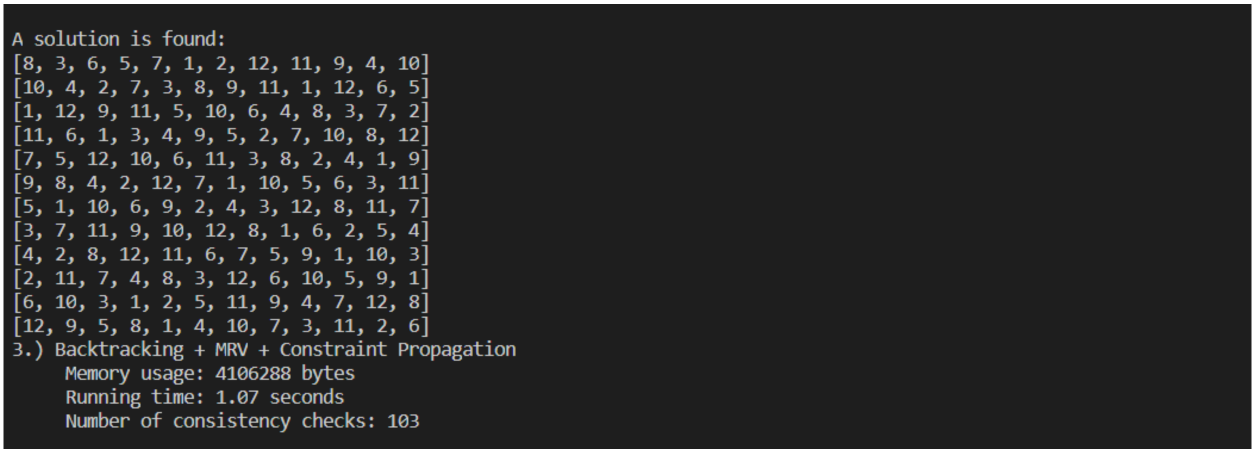
Run python with **sudoku.py** as argument

Sample output:



```
C:\Users\mandy\Documents\CSE537\Kwok_Mandy_HW2>C:\Users\mandy\AppData\Local\Programs\Python\Python37\python.exe sudoku.py
A solution is found:
[8, 3, 6, 5, 7, 1, 2, 12, 11, 9, 4, 10]
[10, 4, 2, 7, 3, 8, 9, 11, 1, 12, 6, 5]
[1, 12, 9, 11, 5, 10, 6, 4, 8, 3, 7, 2]
[11, 6, 1, 3, 4, 9, 5, 2, 7, 10, 8, 12]
[7, 5, 12, 10, 6, 11, 3, 8, 2, 4, 1, 9]
[9, 8, 4, 2, 12, 7, 1, 10, 5, 6, 3, 11]
[5, 1, 10, 6, 9, 2, 4, 3, 12, 8, 11, 7]
[3, 7, 11, 9, 10, 12, 8, 1, 6, 2, 5, 4]
[4, 2, 8, 12, 11, 6, 7, 5, 9, 1, 10, 3]
[2, 11, 7, 4, 8, 3, 12, 6, 10, 5, 9, 1]
[6, 10, 3, 1, 2, 5, 11, 9, 4, 7, 12, 8]
[12, 9, 5, 8, 1, 4, 10, 7, 3, 11, 2, 6]
1.) Backtracking + MRV heuristic
     Memory usage: 4107220 bytes
     Running time: 1.03 seconds
     Number of consistency checks: 186

A solution is found:
[8, 3, 6, 5, 7, 1, 2, 12, 11, 9, 4, 10]
[10, 4, 2, 7, 3, 8, 9, 11, 1, 12, 6, 5]
[1, 12, 9, 11, 5, 10, 6, 4, 8, 3, 7, 2]
[11, 6, 1, 3, 4, 9, 5, 2, 7, 10, 8, 12]
[7, 5, 12, 10, 6, 11, 3, 8, 2, 4, 1, 9]
[9, 8, 4, 2, 12, 7, 1, 10, 5, 6, 3, 11]
[5, 1, 10, 6, 9, 2, 4, 3, 12, 8, 11, 7]
[3, 7, 11, 9, 10, 12, 8, 1, 6, 2, 5, 4]
[4, 2, 8, 12, 11, 6, 7, 5, 9, 1, 10, 3]
[2, 11, 7, 4, 8, 3, 12, 6, 10, 5, 9, 1]
[6, 10, 3, 1, 2, 5, 11, 9, 4, 7, 12, 8]
[12, 9, 5, 8, 1, 4, 10, 7, 3, 11, 2, 6]
2.) Backtracking + MRV + Forward Checking
     Memory usage: 4104744 bytes
     Running time: 0.91 seconds
     Number of consistency checks: 90
```



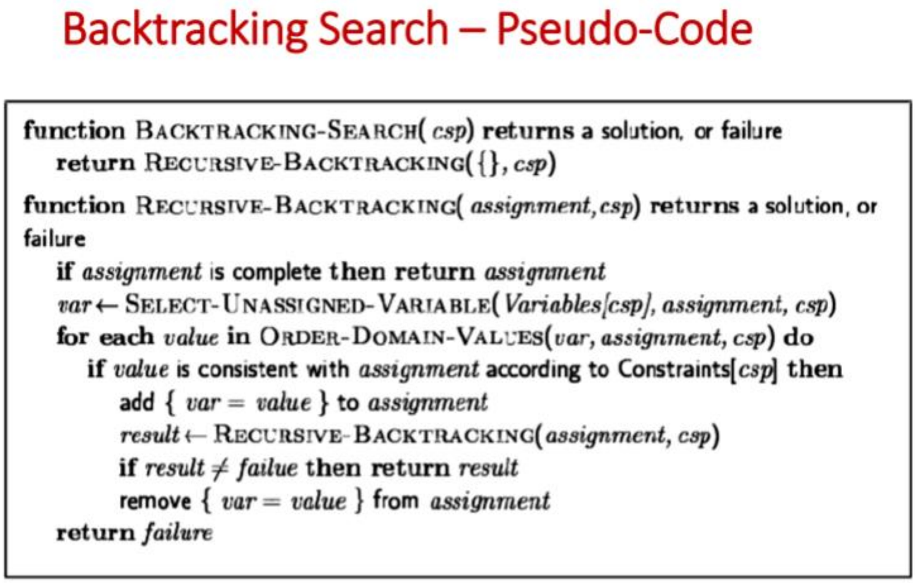
```

A solution is found:
[8, 3, 6, 5, 7, 1, 2, 12, 11, 9, 4, 10]
[10, 4, 2, 7, 3, 8, 9, 11, 1, 12, 6, 5]
[1, 12, 9, 11, 5, 10, 6, 4, 8, 3, 7, 2]
[11, 6, 1, 3, 4, 9, 5, 2, 7, 10, 8, 12]
[7, 5, 12, 10, 6, 11, 3, 8, 2, 4, 1, 9]
[9, 8, 4, 2, 12, 7, 1, 10, 5, 6, 3, 11]
[5, 1, 10, 6, 9, 2, 4, 3, 12, 8, 11, 7]
[3, 7, 11, 9, 10, 12, 8, 1, 6, 2, 5, 4]
[4, 2, 8, 12, 11, 6, 7, 5, 9, 1, 10, 3]
[2, 11, 7, 4, 8, 3, 12, 6, 10, 5, 9, 1]
[6, 10, 3, 1, 2, 5, 11, 9, 4, 7, 12, 8]
[12, 9, 5, 8, 1, 4, 10, 7, 3, 11, 2, 6]
3.) Backtracking + MRV + Constraint Propagation
     Memory usage: 4106288 bytes
     Running time: 1.07 seconds
     Number of consistency checks: 103
```

Mandy Kwok 10/9/20
111458972 CSE537 Artificial Intelligence

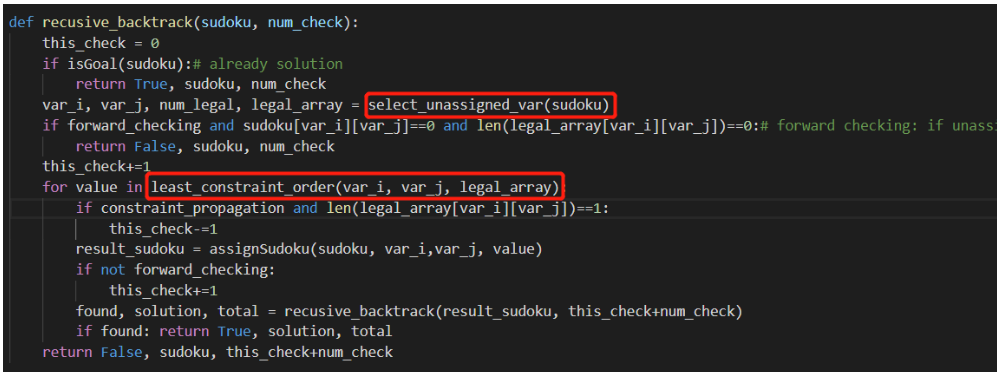
Trace of execution

Backtracking + MRV heuristics



Backtracking Search – Pseudo-Code

```
function BACKTRACKING-SEARCH( csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING({}, csp)
function RECURSIVE-BACKTRACKING( assignment, csp) returns a solution, or
failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE( Variables[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment according to Constraints[csp] then
            add { var = value } to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failue then return result
            remove { var = value } from assignment
    return failure
```



```python
def recusive_backtrack(sudoku, num_check):
    this_check = 0
    if isGoal(sudoku):# already solution
        return True, sudoku, num_check
    var_i, var_j, num_legal, legal_array = select_unassigned_var(sudoku)
    if forward_checking and sudoku[var_i][var_j]==0 and len(legal_array[var_i][var_j])==0:# forward checking: if unass
        return False, sudoku, num_check
    this_check+=1
    for value in least_constraint_order(var_i, var_j, legal_array):
        if constraint_propagation and len(legal_array[var_i][var_j])==1:
            this_check-=1
        result_sudoku = assignSudoku(sudoku, var_i,var_j, value)
        if not forward_checking:
            this_check+=1
        found, solution, total = recusive_backtrack(result_sudoku, this_check+num_check)
        if found: return True, solution, total
    return False, sudoku, this_check+num_check
```

I implemented the Backtracking with MRV heuristic algorithm by following the recursive approach from course slide. The variable named **csp** would be a sudoku represented in Python array with numbers[0,N-1]. The variable named **var** would be one of the element in a sudoku array.

As shown in the highlighted part from the second screenshot, I implemented function **select_unassigned_var()** that replaces SELECT_UNASSIGNED_VARIABLE() in pseudocode to choose the most constrained variable(variable with the fewest legal values) to be assigned next. The function **least_constraint_order()** is used as ORDER_DOMAIN_VALUES() in pseudocode to determine the order of values tried by choosing the least constraining value(rules out the fewest values in remaining variables). These functions contributed to the **MRV heuristic**.

Forward Checking

I implemented the idea to keep track of remaining legal values for unassigned variables using the function **getLegalValues()** that returns an array of legal values for all spaces in sudoku. The function will look at all columns, rows, and sub-grid to eliminate duplicate legal values. During the backtracking search, if an unassigned variable has no legal value then terminate search for that variable.

Constraint Propagation

I implemented the function **constraint_propagation_optimize()** to perform early detection for failures by rechecking inconsistencies or duplicate value in a row, column, or sub-grid from the array of legal values generated by **getLegalValues().** Since this is doing additional work compared to Forward Checking, the runtime and consistency checks will be greater than that of Forward Checking.

Statistics

Based on the multiple trials of output:

Number of Consistency Check: Backtracking+MRV > Backtracking+MRV+Constraint Propagation > Backtracking+MRV+Forward Checking

Runtime: Backtracking+MRV+Constraint Propagation > Backtracking+MRV > Backtracking+MRV+Forward Checking

Memory Usage: Backtracking+MRV > Backtracking+MRV+Constraint Propagation > Backtracking+MRV+Forward Checking