BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS
Artificial Intelligence (BITS F444/ CS F407)
I Semester 2019-20
Programming Assignment-4
Coding Details
(November 1, 2019)

Instruction: Type the details precisely and neatly

1. ID 2015HS120600P
   Name RACHIT RATHORE

2. Mention the names of Submitted files :
   a. ConstraintCheck.py
   b. ConstraintGraphGenerator.py
   c. DomainGraphGenerator.py
   d. GlobalVariables.py
   e. MRV.py
   f. DFS_BT.py
   g. DFS_BT_AC3.py
   h. GUIDriver.py
   i. output.txt
   j. CodingDetails.pdf
   k. ConstraintGraph.jpg

3. Total number of submitted files: 11
4. Name of the folder : 2015HS120600P
5. Have you checked that all the files you are submitting have your name in the top?(yes/no) YES
6. Have you checked that all the files you are submitting are in the folder as specified in 4 (and no subfolder exists)?(yes/no). YES

7. Problem formulation
   a. List of variables (Specify all variables):
      For given problem - variables X = {‘N1’, ‘N2’, ‘N3’, ……. ‘N20’}

   b. Value domains of variables (Also list the variables against each value domain correspondingly)

D(‘N1’)= {2, 5, 7}, D(‘N2’)= {1, 4, 6, 2}, D(‘N3’)= {2, 5, 6, 1}, D(‘N4’)= {2, 4, 6, 8}, D(‘N5’)= {2, 6, 5}, D(‘N6’)= {1, 5, 3}, D(‘N7’)= {2, 4, 6, 1, 8}, D(‘N8’)= {1, 3, 4}, D(‘N9’)= {4, 1, 5, 8, 6}, D(‘N10’)= {8}, D(‘N11’)= {2, 3}, D(‘N12’)= {1, 2, 3, 4, 7}, D(‘N13’)={7, 1, 8}, D(‘N14’)= {5, 3, 6, 1}, D(‘N15’)= {2, 5}, D(‘N16’)= {2, 5, 1, 4}, D(‘N17’)= {1, 4, 5, 6}, D(‘N18’)= {5, 4}, D(‘N19’)= {1, 3, 6, 8}, D(‘N20’)= {6}

   c. Mention the constraints —>(as Adjacency list)

‘N1'--> 'N3', 'N5', 'N12', 'N2', 'N18', 'N19', 'N20', 'N8', 'N9', 'N13'
'N2'--> 'N8', 'N9', 'N12', 'N19', 'N3', 'N5', 'N18', 'N20', 'N1'
'N3'--> 'N5', 'N8', 'N9', 'N12', 'N18', 'N19', 'N4', 'N16', 'N7', 'N11', 'N14', 'N20', 'N2', 'N1', 'N10', 'N13', 'N17'
'N4'--> 'N3', 'N5', 'N16', 'N8', 'N9', 'N19', 'N10', 'N12', 'N20'
'N5'--> 'N3', 'N8', 'N9', 'N12', 'N18', 'N19', 'N4', 'N16', 'N7', 'N11', 'N14', 'N20', 'N2', 'N1', 'N10', 'N17'
'N6'-->
'N7'--> 'N3', 'N5', 'N11', 'N14', 'N20', 'N13', 'N8', 'N9', 'N19'
'N8'--> 'N3', 'N5', 'N9', 'N12', 'N18', 'N19', 'N2', 'N4', 'N16', 'N15', 'N10', 'N20', 'N13', 'N7', 'N1', ‘N11’
'N9'--> 'N3', 'N5', 'N8', 'N12', 'N18', 'N19', 'N2', 'N4', 'N16', 'N15', 'N10', 'N20', 'N13', 'N7', 'N1', 'N11', 'N17'
'N10'--> 'N3', 'N5', 'N8', 'N9', 'N18', 'N19', 'N20', 'N12', 'N4', 'N11', 'N17'
'N11'--> 'N3', 'N5', 'N7', 'N14', 'N20', 'N8', 'N18', 'N19', 'N9', 'N10', 'N17'
'N12'--> 'N3', 'N5', 'N8', 'N9', 'N18', 'N19', 'N2', 'N15', 'N20', 'N1', 'N10', 'N4'
'N13'--> 'N3', 'N8', 'N9', 'N7', 'N19', 'N20', 'N1'
'N14'--> 'N3', 'N5', 'N7', 'N11', 'N20'

'N15'--> 'N8', 'N9', 'N12', 'N16', 'N17', 'N18', 'N19', 'N20'
'N16'--> 'N3', 'N5', 'N4', 'N8', 'N9', 'N19', 'N15', 'N17', 'N18', 'N20'
'N17'--> 'N15', 'N16', 'N18', 'N19', 'N20', 'N3', 'N5', 'N11', 'N9', 'N10'
'N18'--> 'N3', 'N5', 'N8', 'N9', 'N12', 'N19', 'N15', 'N16', 'N17', 'N20', 'N2', 'N1', 'N10', 'N11'
'N19'--> 'N3', 'N5', 'N8', 'N9', 'N12', 'N18', 'N2', 'N4', 'N16', 'N15', 'N17', 'N20', 'N1', 'N10', 'N13', 'N7', 'N11'
'N20'--> 'N15', 'N16', 'N17', 'N18', 'N19', 'N3', 'N5', 'N7', 'N11', 'N14', 'N12', 'N2', 'N1', 'N8', 'N9', 'N10', 'N13', 'N4'

8. Data structure used

    a. Constraint graph node structure:
    Given above constraints, ex = 'N1'--> 'N3', 'N5', 'N12', 'N2', 'N18', 'N19', 'N20', 'N8', 'N9', 'N13'
    "N(i)'s" in front are nodes of graph.

    b. Constraint graph edge structure:
    Given above constraint, ex = 'N1'--> 'N3', 'N5', 'N12', 'N2', 'N18', 'N19', 'N20', 'N8', 'N9', 'N13'
    Tuples ('N1', 'N3'), ('N1', 'N5'), ... etc are edges (neighbours) of graph.
    Both the values in tuples cannot be equal.

    c. Constraint graph (Adjacency list/ adjacency matrix/ any other(specify)
    Constraint graph is implemented as Adjacency list => in Python it is implemented as DICTIONARY, with all the variables as 'key' and list of all its neighbours(variables it is constrained to) as 'value' pair.

    d. How are you maintaining value domains as you go with search process?
    Value domain are implemented as DICTIONARY, with all the variables as 'key' and list of its domain values as 'value' pair.

9. DFS + backtracking technique details

    a. Variable ordering used (List heuristics used):
        1. Default ordering - N1 to N20
        2. MRV ordering - Least no. of domain values of a variable appears first, if there's a tie between no. of domain values degree heuristic is used, i.e., one who has more no. of constraints or neighbours appears first.

    b. Node structure for DFS:
    Each node is a Dictionary of assignment values so far. Ex: one node -> {'N1': 2, 'N2': 4}
    then another node -> {'N1': 2, 'N2': 4, 'N3': 1} etc.

    c. Method for assignment of a value to a variable and backtracking:
    Recursive is used

    d. How is edge node of your adjacency list (constraint graph) useful in deciding upon which constraint module(or modules) to use for testing the violation of the constraints while you assign a value to a variable?
    When a variable is assigned a value, a separate function checks in constraint graph for all its neighbours, if the any of the neighbour have same value assigned then the function returns FALSE (i.e. constraint violated).

    e. Total number of nodes generated for assignment of values to all variables
    NO FULL ASSIGNMENT EXISTS, but nodes generated in traversing full tree = 26635

    f. Write the statistics here as asked

R1 = 26635 R2 = 216 bytes R3 = 18
R4 = 2.912 sec R5= 772

g. Code status (implemented fully/ partially/ not done) IMPLEMENTED FULLY

10. DFS+ Backtracking using constraint propagation:
    a. Explain the method for constraint propagation. How are you updating the value domains? What do you do with the value domains of the variables when you backtrack while performing DFS?
    - Recursion is used with AC3 algo called at every step.
    - At every step when a variable is assigned a value and its not violating any constraint, corresponding inconsistent domain values of its neighbours are reduced.
    - As every node here consist of its own assignment dictionary(assignments till this step) + domain dictionary(new reduced domain), so while backtracking assignment and domain of current node is used.

    b. Total number of nodes generated using the above technique
    NO FULL ASSIGNMENT EXISTS, but nodes generated in traversing full tree = 8545

    c. Write the statistics here as asked

    R6 = 8545 R7 = 0.6781 R8 = 0.7312 sec

    d. Code status (implemented fully/ partially/ not done): IMPLEMENTED FULLY

11. Comparative analysis

Fill in the following information

	DFS+BT	DFS+BT+Constraint propagation
Average number of nodes created	26635	8545
Average time taken (in secs)	2.89	0.73

12. Compilation Details:
    a. Code Compiles (Yes/ No): YES
    b. Mention the .py files that do not compile: NA
    c. Any specific function that does not compile: NA
    d. Ensured the compatibility of your code with the specified Python version(yes/no) YES
    e. Instructions for compilation of your files mentioning the multi file compilation process used by you (We may use the replica of these for compiling your files while evaluating your code)
    —> Put all the files in one folder and build & run GUIDriver.py file.

13. Driver Details: Does it take care of the options specified earlier(yes/no): YES

14. Execution status (describe in maximum 2 lines)
    Executes completely with all functions.

15. Declaration: I, RACHIT RATHORE (name) declare that I have put my genuine efforts in creating the python code for the given programming assignment and have submitted only the code developed by me. I have not copied any piece of code from any source. If the code is found plagiarized in any form or degree, I understand that a disciplinary action as per the institute rules will be taken against me and I will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.
    ID 2015HS120600P Name: RACHIT RATHORE

    Date: 01 November 2019
    ****
    Should not exceed four pages