**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 \_\_2016A7PS0150P\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name \_\_Patel\_Parth\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Mention the names of Submitted files :
   1. utils.py
   2. algo.py
   3. my\_stats.py
   4. GUI.py
   5. testcases.txt
   6. constraint\_graph.jpg
   7. coding details PA4.docx
2. Total number of submitted files: \_\_7\_\_\_\_\_\_
3. Name of the folder :\_2016A7PS0150P\_\_\_\_\_\_\_
4. Have you checked that all the files you are submitting have your name in the top?(yes/no) Yes
5. 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
6. Problem formulation
   1. List of variables (Specify all variables):

1. GROUP\_CHOICES - List of lists to store which group wants to meet which laureates.

` 2. LAUREATE\_CHOICES - List of lists to store available slots/ domain for each laureate.

3. constraint\_graph - Adjacency list, implemented as list of Python sets.

4. assignment - List to store current partial assignment for laureates.

5. heuristic - Choice of heuristic function for variable ordering

6. variable\_order - List to store order in which laureates are assigned values

* 1. Value domains of variables (Also list the variables against each value domain correspondingly):

Here, variables are slots to be assigned to the laureates, and for each particular laureate, domain corresponds to the set of slots during which the laureate is available. For example, in the given problem, domain for N1 is {2, 5, 7}.

* 1. Mention the constraints:

For laureates ‘i’ and ‘j’ (s.t. ‘i’ not equal to ‘j’, 1<= i, j <=m) present in preference of same group, N(i) should not be equal to N(j).

1. Data structure used:
   1. Constraint graph node structure:

Constraint graph is list of sets. For each node, I have a corresponding set in the list to store all its neighbours.

* 1. Constraint graph edge structure:

Constraint graph is list of sets. For each edge (u, v) in the graph, I store u in set corresponding to v in the list, and v in set corresponding to u in the list.

* 1. Constraint graph (Adjacency list/ adjacency matrix/ any other(specify): Adjacency list
  2. How are you maintaining value domains as you go with search process?

I have a variable ‘LAUREATE\_CHOICES’ which is a list of lists to store value domains of each laureate. This is dynamically updated as search process goes on.

1. DFS + backtracking technique details
   1. Variable ordering used (List heuristics used):

1. Default order: N1 to N20

2. MRV (with degree as tie-break).

* 1. Node structure for DFS:

Node stores current partial assignment and maintain the values that have been tried and the ones that are yet to be tried for the next laureate (returned by variable ordering heuristic), checking consistency according to the constraint graph.

* 1. Method for assignment of a value to a variable and backtracking:

A list is used to store current variable assignment. A laureate is assigned a value by changing the corresponding value in the list, and the value is reset to zero while backtracking (if it violates the constraints).

* 1. 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?

To assign a value to a particular variable without violating the constraints, I check that none of its neighbours (as found from the constraint graph) should have the same value in current partial assignment.

* 1. Total number of nodes generated for assignment of values to all variables: 8545
  2. Write the statistics here as asked

R1 = 8545 R2 = 64 R3 = 18

R4 = 0.11493351936 secs. R5= 13

* 1. Code status (implemented fully/ partially/ not done): implemented fully

1. DFS+ Backtracking using constraint propagation:
   1. 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?

In AC3, constraint propagation is based on the idea of arc consistency. Arc refers to a directed arc in the constraint graph, say from variable X to variable Y. The arc is consistent if, for every value x of X, there is some value y of Y that is consistent with x. Otherwise, we delete from domain of X (implemented as a Python set) the values x that have no corresponding consistent value in the domain of Y. This will make the arc consistent. Arc consistency is applied until all the arcs in the constraint graph are consistent. AC3 is a preprocessing step to prune the domains before applying DFS+backtracking on the pruned domains.

* 1. Total number of nodes generated using the above technique: 370
  2. Write the statistics here as asked

R6 = 370 R7 = 0.956699824 R8 = 0.00945091248 secs.

* 1. Code status (implemented fully/ partially/ not done): implemented fully

1. Comparative analysis: Fill in the following information :

|  |  |  |
| --- | --- | --- |
|  | DFS+BT | DFS+BT+Constraint propagation |
| Average number of nodes created | 8545 | 370 |
| Average time taken | 0.11493351936 secs. | 0.00945091248 secs. |

1. Compilation Details:
   1. Code Compiles (Yes/ No):\_\_\_Yes\_\_\_\_
   2. Mention the .py files that do not compile:\_\_N/A\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Any specific function that does not compile:\_\_\_N/A\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Ensured the compatibility of your code with the specified Python version(yes/no)\_\_Yes\_\_\_\_\_\_\_
   5. 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) On ubuntu terminal, use the command: **python GUI.py**
2. Driver Details: Does it take care of the options specified earlier(yes/no):\_\_Yes\_\_\_\_\_
3. Execution status (describe in maximum 2 lines): All the submitted code works. The GUI is as specified - R1 to R9 are displayed, and buttons are provided to input group and laureate data, print group and laureate data to console, select heuristic for variable ordering and run DFS+BT and DFS+BT+AC3. Functionality is provided to view partial assignments after creation of every 100 nodes.
4. Declaration: I, \_\_Patel\_Parth\_\_\_\_\_\_\_\_\_\_\_ (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\_\_2016A7PS0150P\_\_\_\_\_\_\_\_\_\_\_\_\_ Name:\_Patel\_Parth\_\_\_\_\_\_\_\_\_

Date: \_1/11/2019\_\_\_

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Should not exceed four pages \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*