Problem Description:

I have to select a randomly generated graph with random edges and select constraint for each edge from a constraint list, then run the Arc-Consistency algorithms (AC-1, AC-2, AC-3, AC-4) for increasing number of nodes and then measure the performances of each algorithm and plot a graph showing the comparison of four AC-algorithms. As a measure of performance, the following two will be the output graph:

- Domain size reduction v/s execution time
- Number of nodes v/s execution time

Input:

- A random graph (minimum 100 nodes, can be increased iteratively)
- Random constraint associated with each edge from a constraint list
- Random domain associated with each node

So, the problem is <X, D, C>

Where X = set of random variables , $\{X_1, \ldots, X_n\}$, and can be increased.

D = random domain, $\{D_1, \ldots, D_n\}$, associated with each variable.

C = set of constraint for each edge.

Domains are (minimum size of 5, maximum 35):

- Domains will be integers from 1 to 100

Constraints are (will be used randomly for each edge):

1. >=
2. <
3. !=
4. node1 | node2
5. node1 = $(node2)^2$

- If each value of domain of node1 < 80 and each value of domain of node2 > 10, then constraint number (3) will be used.
- If each value of domain of node1 < 15 and each value of domain of node2 > 20, then constraint number (4) will be used.
- If each value of domain of node1 < 15 and each value of domain of node2 > 15, then constraint number (5) will be used.
- For other domain range, constraint (1) and (2) will be used.

Output:

Running each AC-algorithm for increasingly number of nodes and capture the values needed and draw two graph comparing the four algorithms as mentioned above.