Due Date: 27 JAN 2022 at 11:59 PM Eastern

PSet3

CS 4649/7649 Robot Intelligence: Planning
Instructor: Matthew Gombolay
By: Cody Houff

Worked with: Jeremy Collins, Alan Hesu, Dane Wang
Sources: https://ktiml.mff.cuni.cz/~bartak/constraints/propagation.html
Lecture Slides 6,7
https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/
https://stackoverflow.com/questions/36767278/why-is-the-complexity-of-arc-consistency-algorithm-ocd3
https://uweb.engr.arizona.edu/~ece566/class/lecture-notes/Module-II/Mod-II-Lec-7.pdf

Instructions:

- You may work with one or more classmates on this assignment. However, all work must be your own, original work (i.e., no copy+pasting code). You must list all people you worked with and sources you used on the document you submit for your homework
- Solutions to "by hand" problem must be enclosed by a box and be legible.
- You should submit your solutions to PSet 3 to Gradescope. The answers to problems 1-3 should be submitted as a PDF to the written portion of the assignment. The code should be submitted as a .py file to the code portion of the assignment. When you submit your code portion, Gradescope will inform you if you have submitted the correct file and your answers are in the correct format.

Due Date: 27 JAN 2022 at 11:59 PM Eastern

Problem 1

Prove that the complexity of AC-3 is $O(ek^3)$. You are welcome to look up sources online to help with the proof, but the proof must be detailed and derived using your own words – no copy+pasting! Lec6

From Lec 6, the basic summary of the Algorithm:

AC-3 (CSP)

Input: CSP = X,D, C

Output: CSP', the largest arc-consistent subset of CSP

1. FOR every Cij ∈ C # O(e) + ...

2. $Q \leftarrow Q \cup \{<xi, xj>, <xj, xi>\}$

3. ENDFOR

4. While $Q \neq \emptyset$ # Iterations of while loop determined by # of times line 7 is true (as well as e, k, and n)

5. Select and delete arc $<xi, xj>$ from Q

6. Revise (xi, xj) # O(k^2)

7. IF Revise(xi, xj) caused a change to Di #*O(ek)

8. $Q \leftarrow Q \cup <xk, xi> |k \neq i, k \neq j\}$

9. ENDIF

10. ENDWHILE

Complexity of AC-3? = $O(e$ + ek ∗ k^2) = O(ek^3)

Calculate the complexity of the method revise (method revises one arc between two domains).

For each value in one domain, it is looking at all the values of the 2nd domain. So the complexity of the revise method would be **k^2** (k is the maximum domain size).

Worst case scenario, how many times will be the revise be called? At first, all the arcs are in the queue. Every time the revise is called, one arc is removed from the queue. That would be **e** calls of the method.

But we are also adding arcs back to the queue. We are adding an arc back to the queue only if a value was deleted from the domain the arc is pointing to. One arc is pointing to one domain, so we can only add it as many times as the number of values in that domain. So worst case, we are adding every arc back to the queue **k** times.

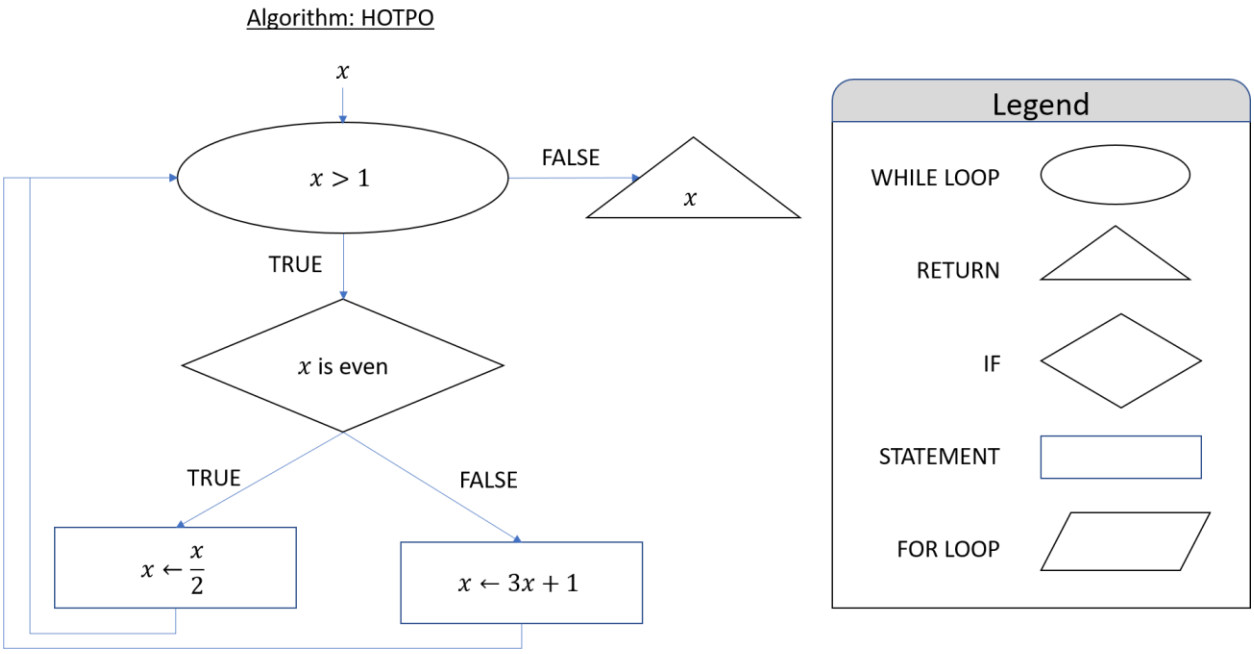
revise is called at maximum **e** + **ek** times (e is the number of arcs).

When we put everything from above, we get complexity of the whole algorithm as **$O(e+ek*k^2)$** which simplifies to **$O(ek^3)$.**

Due Date: 27 JAN 2022 at 11:59 PM Eastern

Problem 2

For this problem, you will draw block diagrams describing the flow of Backtrack Search with Forward Checking and Dynamic Variable Ordering. An example block diagram for HOTPO is shown below. Please follow the same conventions for your solutions.



Part A) Draw a block-diagram describing Backtrack Search with Forward Checking and Dynamic Variable Ordering, based upon slide 63 ("Procedure Dynamic-Var_Forward-Checking(x,D,C)").

Part B) Draw a block-diagram describing Backtrack Search with Forward Checking and Dynamic Variable Ordering, based upon slide 57 ("Procedure Select-Value-FC()"), which is a subroutine for the program you described in Part A.

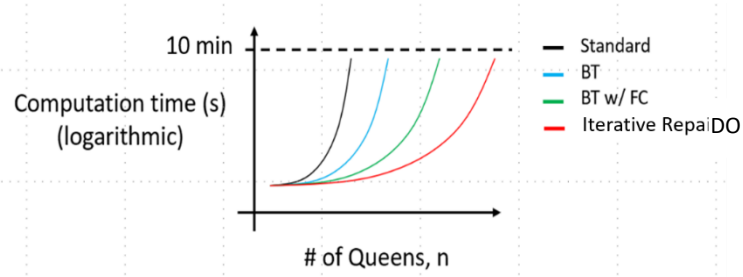
Due Date: 27 JAN 2022 at 11:59 PM Eastern

Problem 3

Implement the following algorithms to solve the n-queens problem, which is defined as the problem of placing n queens on a chess board of size n by n such that no queen can "take" another queen (i.e., no two queens should be on the same row, column or diagonal).

- Standard Search — function: standard()
- Backtrack Search — function: BT()
- Backtrack Search with Forward Checking (FC) — function: BT_w_FC()
- Iterative Repair (Min-Conflict Heuristic) — function: iterative_repair()

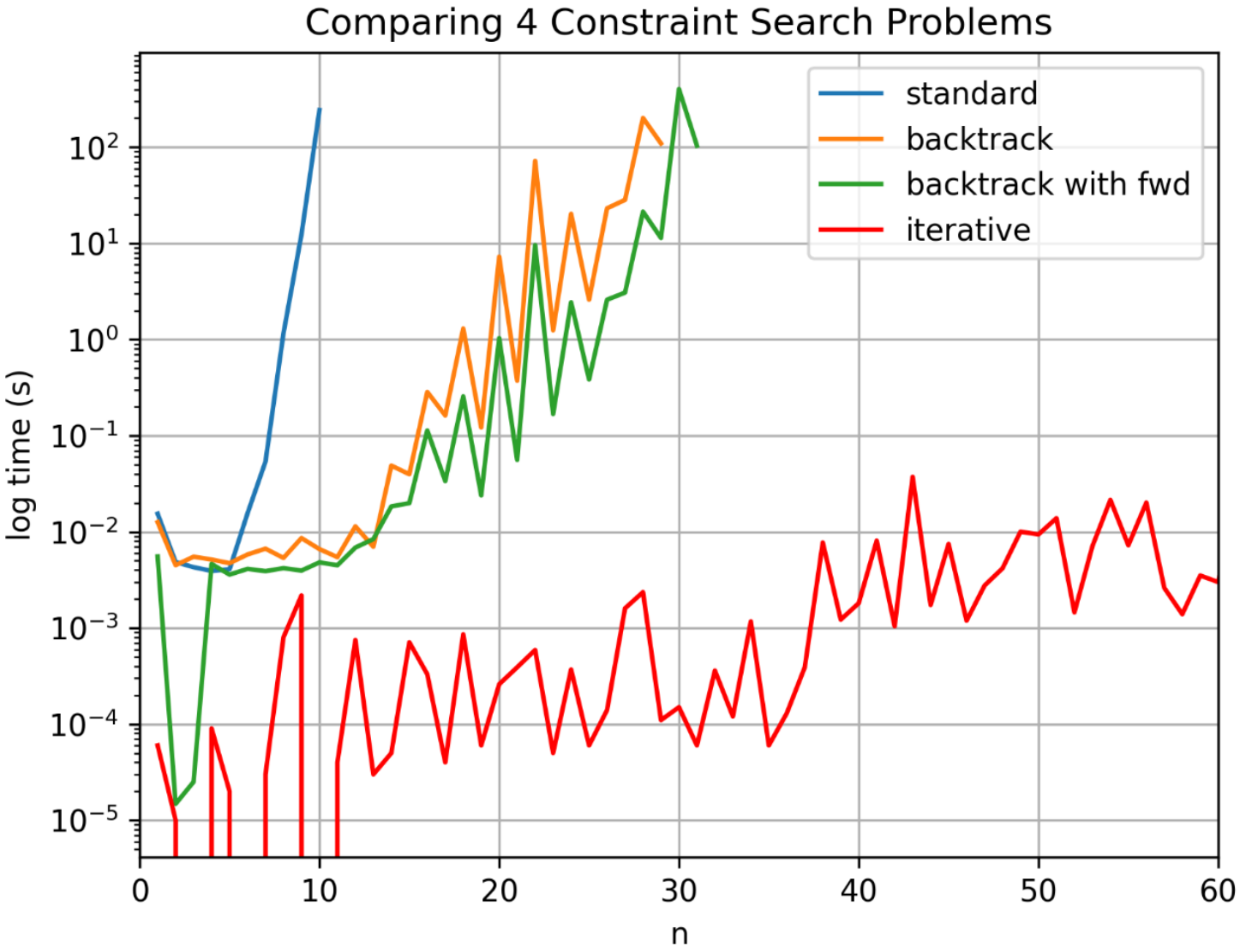
Generate a plot like the one below by trying n = {1,2,…} for each algorithm until the search method can no longer find a solution within 10 minutes of computation time. Include the plot in your PDF submission.



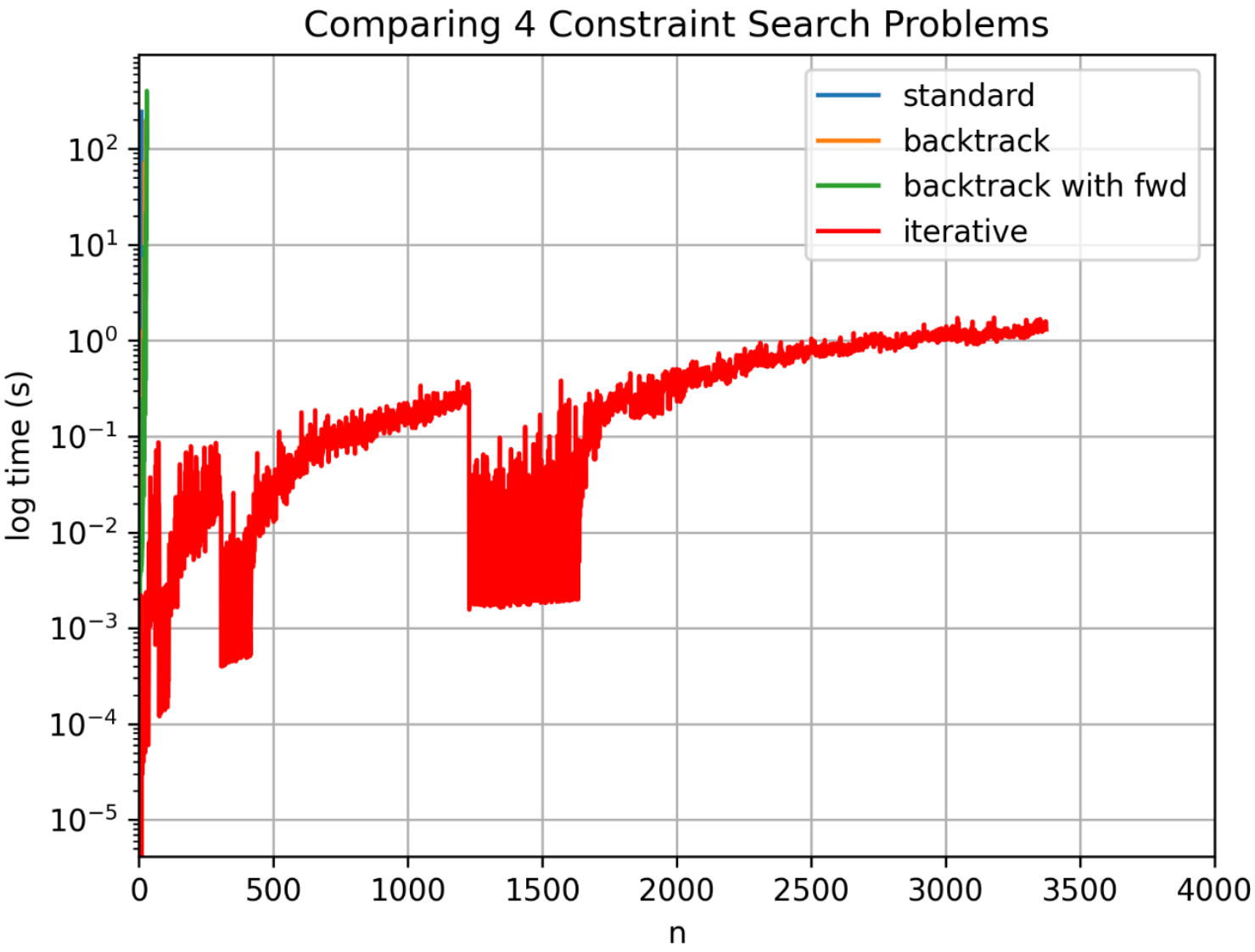
Notes:

- Use the statespace convention we described with columns for variables and rows for the domain values.
- For Standard Search, Backtrack Search, and Backtrack Search with FC, use a fixed ordering assigning queens (your variables) based upon their column index, e.g., the queen in column one would be assigned first, the queen in column two would be assigned second, and so forth.
- For iterative repair, use a random initialization as your greedy heuristic. If after performing hill climbing, you cannot find a solution, generate another random solution (i.e., there should be a loop that keeps trying to find a solution for up to 10 minutes of computation).
- You should have one Python function for each type of search, and the code should be clearly documented. You must use the file template provided (pset3.py). Further, when the grader runs the python code, the function should return the time taken for each # of queens up to 10 minutes in a list. For example, if your first five times are 1 min, 2 min, 3 min, 5 min, and 10 min for n = 1, 2, 3, 4, and 5 respectively, you would return [1,2,3,5,10].
- The 10 minute time-out is not a cumulative time but a time per value of n. So, for example, if your first four times are 1 min, 2 min, 3 min, and 5 min for n = 1, 2, 3, and 4, respectively, you should keep going. You should keep going until, for some n large enough, that you cannot find a solution in <= 10 minutes for that single n-Queens problem.

Due Date: 27 JAN 2022 at 11:59 PM Eastern



Due Date: 27 JAN 2022 at 11:59 PM Eastern



Due Date: 27 JAN 2022 at 11:59 PM Eastern

Graph Iterative with larger steps to see around what n value it hits 10 min