1. (7 points) In this assignment you will solve the N Queens problem. The problem is to place N Queens on an N X N chess board so that no 2 queens attack each other. In chess a queen attacks another queen if they are in the same row, column or diagonal. Your programs should take an integer N as an argument and return the number of solutions to the N-queens problem.

You will view this as depth first search on a tree, where the nodes correspond to partial configurations of queens on the first *i* rows. The root node has no queens. Its children only have queens in the first row. Their children have queens in the first two rows, and so on. Each node at level i can be represented as a list of columns that correspond to the positions of queens in rows 0 thru i-1. Implement the following two versions of the program for N=4 up to 8 (or to a maximum N it is feasible to run on your machine) and report the time taken, the number of nodes visited, and the number of solutions found for problems of different sizes on your machine in a table or a plot.

a) Exhaustive search: Search the tree exhaustively and report the number of solutions to the problem. Call this function E_Queens.

b) Backtracking search: Terminate the recursion under a node if any two queens already attack each other in that node. You must still return the number of all possible solutions. Call this function B_Queens.

Comment on the relative performance of the two versions.

2. (1 point). We have three containers of sizes *a* pints, *b* pints, and *c* pints where $a > b > c$. Initially the first container is empty and the others are full of water. You are allowed one type of action: pour water from one container to another until the first one is empty or the second one is full. We want to know if there is a sequence of actions that results in exactly 2 pints in one of the containers. Formulate this problem as a depth first search. What are the vertices and edges?

3. (1 point). You are given a directed graph. Give an $O(n+e)$ algorithm that determines whether or not it has a vertex *s* from which all other vertices are reachable.

4. (1 point). Design a linear-time algorithm which, given an undirected graph G and a particular edge $(u, v)$, determines if there is a cycle containing $(u, v)$.

**How to submit?**

Label the source file 'hw4.py' and the report report4.pdf. Submit the report on Canvas and the source file at the teach site: https://teach.engr.oregonstate.edu/