CS210
Lab 0
Aha!! of Algorithms

Srinibas Swain (srinibas@iiitg.ac.in)

Welcome to Lab 0 of CS210. This entire lab is designed to warm start your coding neurons. Feel free to use a programming language of your choice, it can be Python as well.

Objectives

The **objectives of this Lab** are:

- To discuss the use of backtracking in algorithms.
- To implement a backtracking algorithm for the n-Queens problem

Useful Topics:

For this workshop, you may find it useful to review some of the following concepts:

- Recursion
- Backtracking

Task 1:

The N–*Queens* Problem: The N–Queens Problem is the problem of placing N queens on an $N \times N$ board so that no two queens are in the same row, column or diagonal.

Task A:

Write a function named `getPositions` that takes as input a list L representing a partial solution to the n-Queens problem and a positive integer n representing the dimension of the $n \times n$ chessboard. The input list contains the positions of the Queens currently placed on the board (or is empty if no Queens have been placed). If the board is represented by list L, then the entry $L[i]$ gives the row position of the Queen in column i. You may assume that the partial solution gives the positions of the Queens in the first i columns and that the other columns are empty.

Your program should return a list containing possible row positions for a Queen to be placed in the next column on the board. (The index of the next column can be obtained by the length of L).

For example:
If your list is $[5, 3]$ and $n = 6$, your program should return the list $[0, 1]$, that is, the next Queen could be placed at row 0 column 2, or row 1 column 2, of the board.

Some further examples:
If your list is empty and $n = 5$, your program should return the list $[0, 1, 2, 3, 4]$.
If your list is $[3, 1]$ and $n = 4$, your program should return an empty list.

Task B:

In order to test your program in Task A, modify the program in Task A so that it prints a table representing the chessboard. It should have Q in entries where a queen is placed, X in entries where a Queen may be placed in the next column and 0 in all other entries.

For example:
If your list is $[5, 3]$ and $n = 6$, your program should print:

```
0 0 X 0 0 0
0 0 X 0 0 0
0 0 0 0 0 0
0 Q 0 0 0 0
0 0 0 0 0 0
Q 0 0 0 0 0
```

If your program `getPositions` is correctly implemented, a Queen placed on a position marked ‘X’ cannot “attack” any of the positions marked ‘Q’, so you can easily check if ‘X’ marks a valid position. What else should you check for?

Task 2:

Useful material: In this task you may find “Implementing Backtracking” useful.

Task A:

The following is an algorithm `nQueens` that takes as input a list *partialSolution* and a positive integer n and prints all solutions to the n-Queens problem using backtracking.

Write a program that takes as input a positive integer n and prints all possible solutions to the n-Queens problem. Your program should use *backtracking.*

For example, your program may output:

```
Enter value for n: 4
Solutions
------------
[1, 3, 0, 2]
[2, 0, 3, 1]
------------
```

and

```
Enter value for n: 2
Solutions
------------
------------
```

Or you might prefer to print them as tables. For example:

```
Enter value for n: 4
Solutions
-------------------------
0 0 Q 0
Q 0 0 0
0 0 0 Q
```

```
0 Q 0 0
-------------------------
0 Q 0 0
0 0 0 Q
Q 0 0 0
0 0 Q 0
-------------------------
```