In chess, queens can move any number of squares vertically, horizontally, or diagonally. The *n-queens puzzle* is the problem of placing `n` queens on an `n × n` chessboard so that no two queens can attack each other.

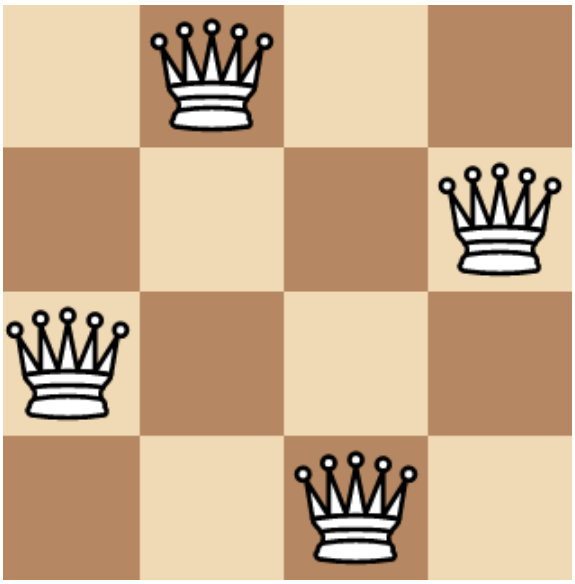
Given an integer `n`, print all possible distinct solutions to the *n-queens puzzle*. Each solution contains distinct board configurations of the placement of the `n` queens, where the solutions are arrays that contain permutations of `[1, 2, 3, .. n]`. The number in the `i`th position of the results array indicates that the `i`th column queen is placed in the row with that number. In your solution, the board configurations should be returned in lexicographical order.

Example

- For `n = 1`, the output should be `nQueens(n) = [[1]]`;

- For `n = 4`, the output should be
  `nQueens(n) = [[2, 4, 1, 3],`
- `[3, 1, 4, 2]]`
-

  This diagram of the second permutation, `[3, 1, 4, 2]`, will help you visualize its configuration:



The element in the `1st` position of the array, `3`, indicates that the queen for column `1` is placed in row `3`. Since the element in the `2nd` position of the array is `1`, the queen for column `2` is placed in row `1`. The element in the `3rd` position of the array is `4`, meaning that the queen for column `3` is placed in row `4`, and the element in the `4th` position of the array is `2`, meaning that the queen for column `4` is placed in row `2`.

Input/Output

- **[execution time limit] 4 seconds (js)**

- **[input] integer n**
  The size of the board.
  Guaranteed constraints:
  `1 ≤ n ≤ 10`.

- **[output] array.array.integer**
  All possible distinct board configurations of the placement of the `n` queens, ordered lexicographically.