GAS STATION

Question: https://leetcode.com/problems/gas-station/

Solution:

```
class Solution {
    public int canCompleteCircuit(int[] gas, int[] cost) {


        int sumGas = 0, sumCost = 0, tank = 0, start = 0;
        for(int i = 0; i < gas.length; i++) {
            sumGas += gas[i];
            sumCost += cost[i];
            tank += gas[i] - cost[i];
            if(tank < 0) {
                start = i + 1;
                tank = 0;
            }
        }

        if(sumGas < sumCost) return -1;
        return start;
    }
}
```

Sequential Digits

Question: https://leetcode.com/problems/sequential-digits/

Solution:

https://leetcode.com/problems/sequential-digits/discuss/?currentPage=1&orderBy=hot&query=

N queen Problem

Question: https://leetcode.com/problems/n-queens-ii/

Solution:

```
class Solution {


        int totalChessBoard;

        char[][] board;


        private boolean isPossible(int n, int row, int col) {


                for(int i = row - 1; i >= 0; i--)

                        if(board[i][col] == 'Q')

          return false;


                int i = 1;

                while(row - i >= 0 && col - i >= 0) {

                        if(board[row - i][col - i] == 'Q')

          return false;

                        ++i;

                }


                i = 1;

                while(row - i >= 0 && col + i < n) {

                        if(board[row - i][col + i] == 'Q')

          return false;

                        ++i;

                }

                return true;

        }
        }



    private void nQueen(int n, int row) {

                if(n == row) {
                        ++totalChessBoard;
                        return;
                }

                for(int i = 0; i < n; i++)
                        if(isPossible(n, row, i)) {
                                board[row][i] = 'Q';
                                nQueen(n, row + 1);
                                board[row][i] = '.';
                        }
        }



    public int totalNQueens(int n) {
                board = new char[n][n];
                for(int i = 0; i < n; i++)
                        for(int j = 0; j < n; j++)
                                board[i][j] = '.';

                totalChessBoard = 0;
                nQueen(n, 0);
                return totalChessBoard;
        }
}
```

Minimum Jumps

Question: https://leetcode.com/problems/jump-game-ii/

Solution:

```
class Solution {
  public int jump(int[] nums) {


    int i = 0, maximumMove = 0, min = 0;
    while(i < nums.length && maximumMove < nums.length - 1) {
      int max = 0;
      while(i <= maximumMove) {
        max = Math.max(max, i + nums[i]);
        i++;
      }
      maximumMove = max;
      ++min;
    }


    return min;
  }
}
```