TSMC OA

1. Profit: Greedy (spend 20 min)

   Have budget `x` (integer), can buy and sell several items which are in a cost list. For the ith item, can earn profit `2^i` by spending `cost[i]`. What the max total profit you can earn from buying the items with given budget? Return `max_profit % (10 ** 9 + 7)` .

   Example:

   - cost = [10, 20, 40, 50, 60], x = 70
   - if buy 10 + 20 + 40, you can earn 2^0 + 2^1 + 2^2 = 1 + 2 + 4 = 7. (not the max)

   Solution:

   - Greedy: buying backwards since profit is determined by the index (power).
   - if buy 60 + 10, you can earn 2^4 + 2^0 = 17 (max)

   ```
   def f(cost, x):
       profit = 0
       for i in range(len(cost) - 1, -1, -1):
           if x >= cost[i]: # can buy
               profit += 2 ** i
               x -= cost[i]
               if x <= 0: break
       return profit % (10 ** 9 + 7)
   ```

2. Decrypt: Char manipulation (spend 20 min)

   There's a cypher cycle, to decrypt the encrypted string, move to the `k` th counter-clockwise character. Return the decrypted string.

```
def decrypt(encrypted, k):
    k %= 26
    decrypted = ""
    for char in encrypted:
        move_back_k_pos = ord(char) - k
        if move_back_k_pos < 65:
            decrypted += chr(move_back_k_pos + 26)
        else:
            decrypted += chr(move_back_k_pos)

    return decrypted
```

3. Array manipulation (spend 45 min)

   Given integer array `arr` and two pointers `p1` (start from 0) and `p2` (start from 1).

   - `segsize` = size of subarray
   - a subarray starts from p2 with a size of segsize `arr[p2:p2+segsize]` or all the remaining `arr[p2:]`
   - If number at p1 is greater than or equal to every single elements in subarray, p1 increment by 1, p2 increment by segsize. Continue this process until p1 is smaller than any of the elements in subarray or this process reached last element of arr.
   - Return the minimum segsize needed to reach to process the whole arr or -1 if can't find.

```
def f(arr):
    def compare(arrp1, subarr):
        for n in subarr:
            if arrp1 < n:
                return False
        return True

    def try_segsize(segsize):
        # init pointers
        p1, p2 = 0, 1

        # init subarray
        subarray = None
        if p2 + segsize <= len(arr):
            subarray = arr[p2:p2+segsize]
        else:
            subarray = arr[p2:]

        while p1 < len(arr) and p2 < len(arr) and compare(arr[p1], subarray):
            p1 += 1
            p2 += segsize

            # update subarray
            if p2 < len(arr) - 1: # have not reach the end
                if p2 + segsize <= len(arr):
                    subarray = arr[p2:p2+segsize]
                else:
                    subarray = arr[p2:]
            else: # reach the end
                return True
        return False


    for segsize in range(len(arr)): # try different segsize
        if try_segsize:
            return segsize
    return -1
```