SimpleCipher - 转盘

- As part of a Day 1 Challenge, your new team at Amazon has created a basic alphabet-based encryption and has asked members to test the cipher. A simple cipher is built on the alphabet wheel which has uppercase English letters ['A'-'Z'] written on it.

```
public static String simpleCipher(String encrypted, int k){
        char[] encryptedArr = encrypted.toCharArray();
        for (int i = 0; i<encryptedArr.length; i++){
            char x = encryptedArr[i];
            // if the previous kth element is greater than 'A'
            if (x-k >= 65){
                encryptedArr[i] =  (char)(x-k);
            }
            // if ascii code of kth previous element if less than a, then add 26
            else{
                encryptedArr[i] = (char)(x-k+26);
            }
        }
        return new String(encryptedArr);
    }
```

MaxProfit

- For example, the product list is divided into n=6 sections and will select k=2 contiguous sections and those opposite to invest in. The profit estimates are profit = [1,5,1,3,7,-3] respectively.

```
public static int maxProfit(int k, List<Integer> profit){
        int n = profit.size();
        int maxSum = Integer.MIN_VALUE;
        int[] prefixSum = new int[2*n + 1];

        for(int i = 0; i<n; i++){
            prefixSum[i+1] = prefixSum[i] + profit.get(i);
        }

        for(int i=0; i<n; i++){
            prefixSum[i+n+1] = prefixSum[i+n] + profit.get(i);
        }

        for(int i=0; i<n; i++){
            int currSum = prefixSum[i+k] - prefixSum[i] + prefixSum[n/2+i+k] - prefixSum[n/2+i];
            maxSum =  Math.max(currSum, maxSum);
        }

        return maxSum;
    }
```

```
2. max profit
有参考这里https://www.1point3acres.com/bbs/thread-796903-1-1.html icebane的comment
用一个sliding window，分两半store
for (int i = 0; i < k; i++) {
        cur += profit;
        cur += profit[n / 2 + i];
    }
循环半个饼图
for (int i = 1; i < n / 2; i++) {
        cur = cur - profit[i - 1] + profit[i + k - 1];
        cur = cur - profit[n / 2 + i - 1] + profit[(n / 2 + i + k - 1) % n];
        res = Math.max(res, cur);
    }
```

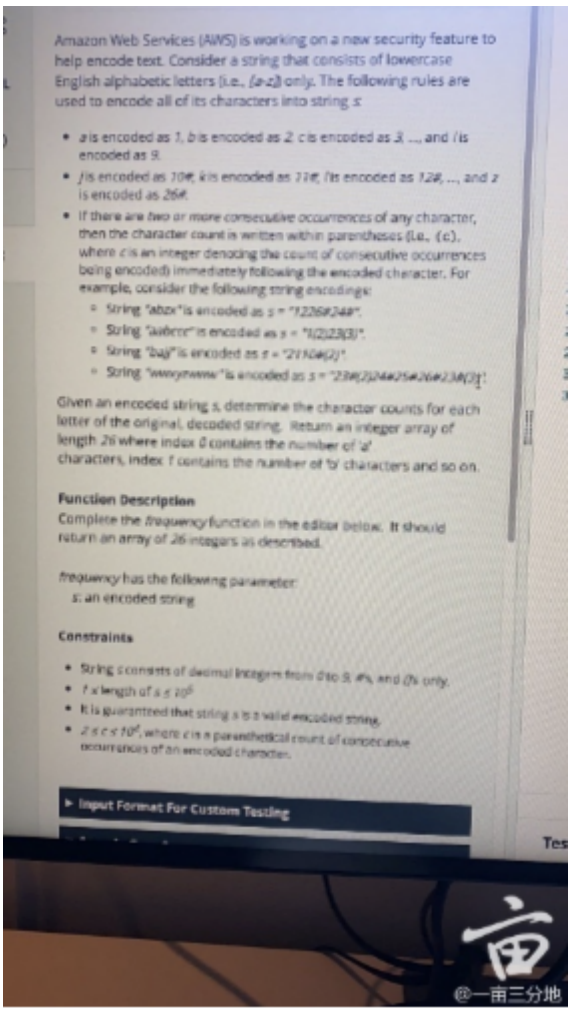
Investment Max Value

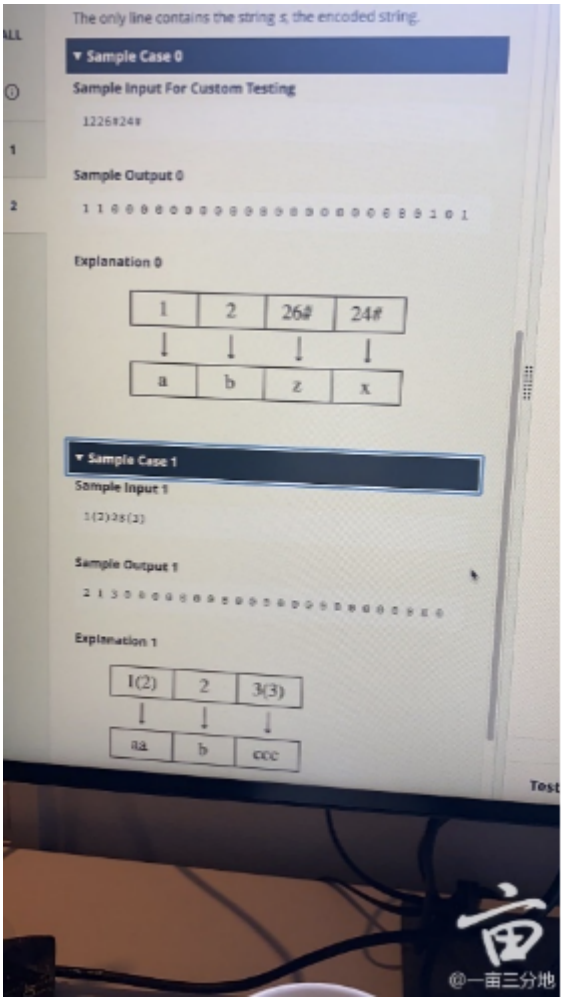
- For example, start with an array of 5 elements: investments = [0,0,0,0,0]*. The variable left and right represent the starting and ending indices, inclusive. Another varaible, contribution, is the new funds to invest per asset.

```
public static int maxValue(int n, int[][] rounds){
        int[] arr = new int[n];

        for(int i =0; i< rounds.length; i++){
            arr[rounds[i][0]] += rounds[i][2];
            if(rounds[i][1] < n-1){
                arr[rounds[i][1] + 1] -= rounds[i][2];
            }
        }
        for(int i=1; i<n; i++){
            arr[i] += arr[i-1];
        }
        int i;

        // Initialize maximum element
        int max = arr[0];

        // Traverse array elements from second and
        // compare every element with current max
        for (i = 1; i < arr.length; i++)
            if (arr[i] > max)
                max = arr[i];

        return max;

    }

    public static int maxValue2(int n, List<List<Integer>> rounds) {
        PriorityQueue<List<Integer>> pq = new PriorityQueue<List<Integer>>(
                (round1, round2) -> round1.get(0).compareTo(round2.get(0)));
        int maxValue = 0;
        int curValue = 0;
```

```
        rounds.sort((round1, round2) -> round1.get(0).compareTo(round2.get(0)));
        for (List<Integer> round : rounds) {
            curValue += round.get(2);
            // pop all intervals that right index < current round left index
            while (pq.size() > 0 && pq.peek().get(0) < round.get(0)) {
                List<Integer> r = pq.remove();
                curValue -= r.get(1);
            }
            maxValue = Math.max(maxValue, curValue);
            pq.add(round.subList(1, 3));
        }
        return maxValue;
    }

    public static int maxValue3(int n, int[][] rounds) {
        List<List<Integer>> roundList = new ArrayList<>();
        for(int i=0;i<rounds.length;i++) {
            List<Integer> round = new ArrayList<>();
            for (int j = 0; j < rounds[0].length; j++) {
                round.add(rounds[i][j]);
            }
            roundList.add(round);
        }
        PriorityQueue<List<Integer>> pq = new PriorityQueue<>(
                (round1, round2) -> round1.get(0).compareTo(round2.get(0)));
        int maxValue = 0;
        int curValue = 0;
        roundList.sort((round1, round2) -> round1.get(0).compareTo(round2.get(0)));
        for (List<Integer> round : roundList) {
            curValue += round.get(2);
            // pop all intervals that right index < current round left index
            while (pq.size() > 0 && pq.peek().get(0) < round.get(0)) {
                List<Integer> r = pq.remove();
                curValue -= r.get(1);
            }
            maxValue = Math.max(maxValue, curValue);
            pq.add(round.subList(1, 3));
        }
        return maxValue;
    }
```

Frequency-decode string 0-9, 11#-26#






```
public static List<Integer> frequency(String s){
        int arr[] = new int[26];
        int n = s.length();
        int i = 0;
        while (i < n) {
            int c = 0;
            int count = 1;
            if (i + 2 < n && s.charAt(i + 2) == '#') {
                c = Integer.parseInt(s.substring(i, i + 2)) - 1;
                i += 3;
            } else {
                c = Character.getNumericValue(s.charAt(i)) - 1;
                i += 1;
            }
            if (i < n && s.charAt(i) == '(') {
                count = 0;
                i += 1;
                while (s.charAt(i) != ')') {
                    count *= 10;
                    count += Character.getNumericValue(s.charAt(i));
```

```
                i += 1;
            }
            i += 1;
        }
        arr[c] += count;
    }
    return Arrays.stream(arr).boxed().collect(Collectors.toList());
}
```

Coin Sequence- Flip coins

Amazon Shopping recently launched a new Coin Collection Album. Each page has a coin pasted on it, with either the head or tail side facing upward, represented by 'H'and 'T' respectively. A sequence of coins is called beautiful if all the head-facing coins are pasted before all the tail-facing coins. More formally, a beautiful sequence is a sequence of the form "HH.....TTT".

```
public static int flips(String s) {
    int tails = 0;
    int flips = 0;

    for (int i = 0; i < s.length(); i++) {
        Character c = s.charAt(i);
        if (c == 'T') {
            tails++;
        } else {
            flips++;
        }

        flips = Math.min(tails, flips);
    }
    return flips;
}
```

teamsize maxdiff 组队问题 **Count Maximum Teams**

Amazon is hosting a team hackathon.
Each team will have exactly teamSize developers.
A developer's skill level is denoted by skill[i].
The difference between the maximum and minimum skill levels within a team cannot exceed a threshold, maxDiff.

```
public static int teams(List<Integer> skill, int size, int diff) {
    Collections.sort(skill);
    int count = 0;

    int startIndex = 0;
    while (true) {
```

```
            if (startIndex + size - 1 >= skill.size()) {
                break;
            } else if (skill.get(startIndex + size - 1) - skill.get(startIndex) <= diff) {
                count += 1;
                startIndex += size;
            } else {
                startIndex += 1;
            }
        }

        return count;
    }
```

```
public class Main {
    public static void main(String[] args) {
        int teams = getTeams(new int[] { 3, 4, 3, 1, 6, 5 }, 3, 2);
        System.out.println(teams);
    }

    static int getTeams(int[] skills, int teamSize, int maxDiff) {
        Arrays.sort(skills);

        int n = teamSize - 1;
        int count = 0;
        for (int i = 0; i < skills.length - n; i++) {
            int start = skills[i];
            int end = skills[i + n];
            if (end - start <= maxDiff) {
                count++;
                i += n;
            }
        }
        return count;
    }
}
```

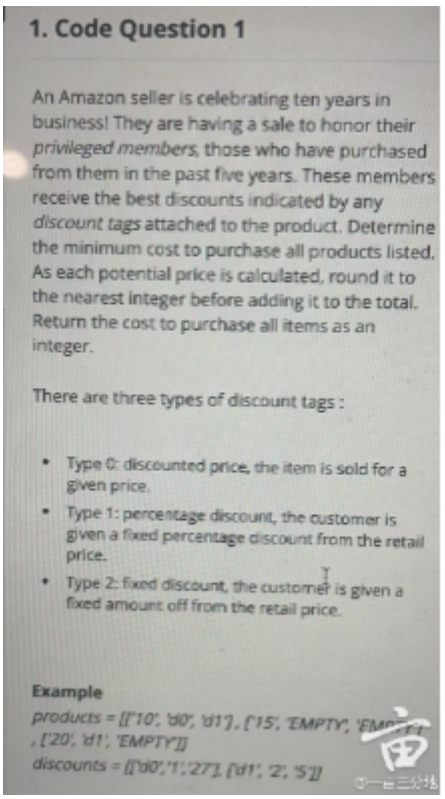
Find Valid Discount Coupons

At Amazon's annual sale. employees are tasked with generating valid discount coupons for loyal customers. However, there are some used/invalid coupons in the mix and the challenge in this task is to determine whether a given discount coupon is valid or not.

```
public static int valid(String discount) {
        Stack<Character> stack = new Stack<Character>();
        for (int i = 0; i < discount.length(); i++) {
            if (stack.size() > 0 && stack.peek() == discount.charAt(i)) {
                stack.pop();
            } else {
                stack.push(discount.charAt(i));
            }
        }
        if (stack.size() == 0) {
```

```
            return 1;
        } else {
            return 0;
        }
    }

    public static List<Integer> coupons(List<String> discounts) {
        List<Integer> results = new ArrayList<Integer>();
        for (String discount : discounts) {
            results.add(valid(discount));
        }
        return results;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int discountsLength = Integer.parseInt(scanner.nextLine());
        List<String> discounts = new ArrayList<>();
        for (int i = 0; i < discountsLength; i++) {
            discounts.add(scanner.nextLine());
        }
        scanner.close();
        List<Integer> res = coupons(discounts);
        System.out.println(res.stream().map(String::valueOf).collect(Collectors.joining(" ")));
    }
```

找三种不同类型**discount**（**shopping discount** 购物车账单）**Find Lowest Price**



1. Code Question 1

An Amazon seller is celebrating ten years in business! They are having a sale to honor their *privileged members*, those who have purchased from them in the past five years. These members receive the best discounts indicated by any *discount tags* attached to the product. Determine the minimum cost to purchase all products listed. As each potential price is calculated, round it to the nearest integer before adding it to the total. Return the cost to purchase all items as an integer.

There are three types of discount tags :

- Type 0: discounted price, the item is sold for a given price.
- Type 1: percentage discount, the customer is given a fixed percentage discount from the retail price.
- Type 2: fixed discount, the customer is given a fixed amount off from the retail price.

Example

products = [['10', 'd0', 'd1'], ['15', 'EMPTY', 'EMPTY'], ['20', 'd1', 'EMPTY']]

discounts = [['d0','1','27'], ['d1', '2', '5']]

```
public static int findLowestPrice(List<List<String>> products, List<List<String>> discounts) {
    int sum = 0;
    HashMap<String, List<String>> discountMap = new HashMap<String, List<String>>();
    for (List<String> discount : discounts) {
        discountMap.put(discount.get(0), discount.subList(1, discount.size()));
    }
    for (List<String> product : products) {
        int price = Integer.parseInt(product.get(0));
        double minPrice = price;
        for (String tag : product.subList(1, product.size())) {
            if (tag.equals("EMPTY")) {
                continue;
            }
            List<String> data = discountMap.get(tag);
            String kind = data.get(0);
            int value = Integer.parseInt(data.get(1));
            if (kind.equals("0")) {
                minPrice = Math.min(minPrice, value);
            } else if (kind.equals("1")) {
                minPrice = Math.min(minPrice, price * (1 - 0.01 * value));
            } else if (kind.equals("2")) {
                minPrice = Math.min(minPrice, price - value);
            }
        }
        sum += Math.round(minPrice);
    }
```

```
    return sum;
  }
```

Good Segment - Bad number

You've been nominated to participate in a programming challenge as part of your Day 1 Orientation at Amazon. Given an array of bad numbers and a range of integers, determine the longest segment of integers within the range that does not include any bad numbers.

```
public static int goodSegment(List<Integer> badNumbers, int lower, int upper) {
    int maxLength = 0;
    Collections.sort(badNumbers);
    int n = badNumbers.size();

    // use binary search to find first bad number >= lower
    int l = 0;
    int r = n - 1;
    int mid = 0;
    while (l <= r) {
      mid = (l + r) / 2;
      if (badNumbers.get(mid) < lower) {
        l = mid + 1;
      } else if (badNumbers.get(mid) > lower) {
        r = mid - 1;
      } else {
        break;
      }
    }
    int i = mid;
    int left = lower;
    while (i < n && left <= upper) {
      int right = Math.min(badNumbers.get(i) - 1, upper);
      maxLength = Math.max(maxLength, right - left + 1);
      left = badNumbers.get(i) + 1;
      i++;
    }
    maxLength = Math.max(maxLength, upper - left + 1);
    return maxLength;
  }
```

Optimal Utilization

Given 2 lists a and b. Each element is a pair of integers where the first integer represents the unique id and the second integer represents a value. Your task is to find an element from a and an element form b such that the sum of their values is less or equal to target and as close to target as possible. Return a list of ids of selected elements. If no pair is possible, return an empty list.

https://leetcode.com/discuss/interview-question/373202

https://leetcode.com/discuss/interview-question/1434609/Amazon-or-OA-August-2021-or-SDE-I-New-Grad-2022-(Reject)

```
private static List<int[]> getPairs(List<int[]> a, List<int[]> b,  int target) {
        // Time: O(a log a + b log b + (a+b))
        List<int[]> results = new ArrayList<>();
        // Sort based on it's values
        Collections.sort(a, (ai, bi) -> ai[1] - bi[1]);
        Collections.sort(b, (ai,bi) -> ai[1] - bi[1]);

        int left = a.size() - 1, right = 0;
        int max = Integer.MIN_VALUE;


        while (left >= 0 && right < b.size()) {

            // if sum > target, decrement left
            // else i'ts <= target. If the sum is closer to target than previous recorded max, then clear the results to add new ones.
            int sum = a.get(left)[1] + b.get(right)[1];
            if (sum > target) {
                left--;
            } else {
                // sum is less or equal to target
                if (sum > max) {
                    max = sum;
                    // closer sum found! remove previous results since this sum is closer
                    results.clear();

                }
                if (sum == max) // if the sum is the current max, add
                    results.add(new int[]{a.get(left)[0], b.get(right)[0]});
                right++; // increment to find closer to target

            }
        }

        return results;
    }

    private static List<List<Integer>> compute(int[][] a, int[][] b, int target) {
        List<List<Integer>> res = new ArrayList<>();
        TreeMap<Integer, List<List<Integer>>> tree = new TreeMap<>();

        for (int i=0; i<a.length; i++) {
            for (int j=0; j<b.length; j++) {
                int sum = a[i][1] + b[j][1];
                if (sum <= target) {
```

```
                List<List<Integer>> list = tree.computeIfAbsent(sum, (k) -> new ArrayList<>());
                list.add(Arrays.asList(a[i][0], b[j][0]));
            }
        }
    }

    return tree.floorEntry(target).getValue();
}
```

Minimum Difficulty of a Job Schedule (Beta Testing)
You want to schedule a list of jobs in d days. Jobs are dependent (i.e To work on the i-th job, you have to finish all the jobs j where 0 <= j < i).

You have to finish at least one task every day. The difficulty of a job schedule is the sum of difficulties of each day of the d days. The difficulty of a day is the maximum difficulty of a job done in that day.

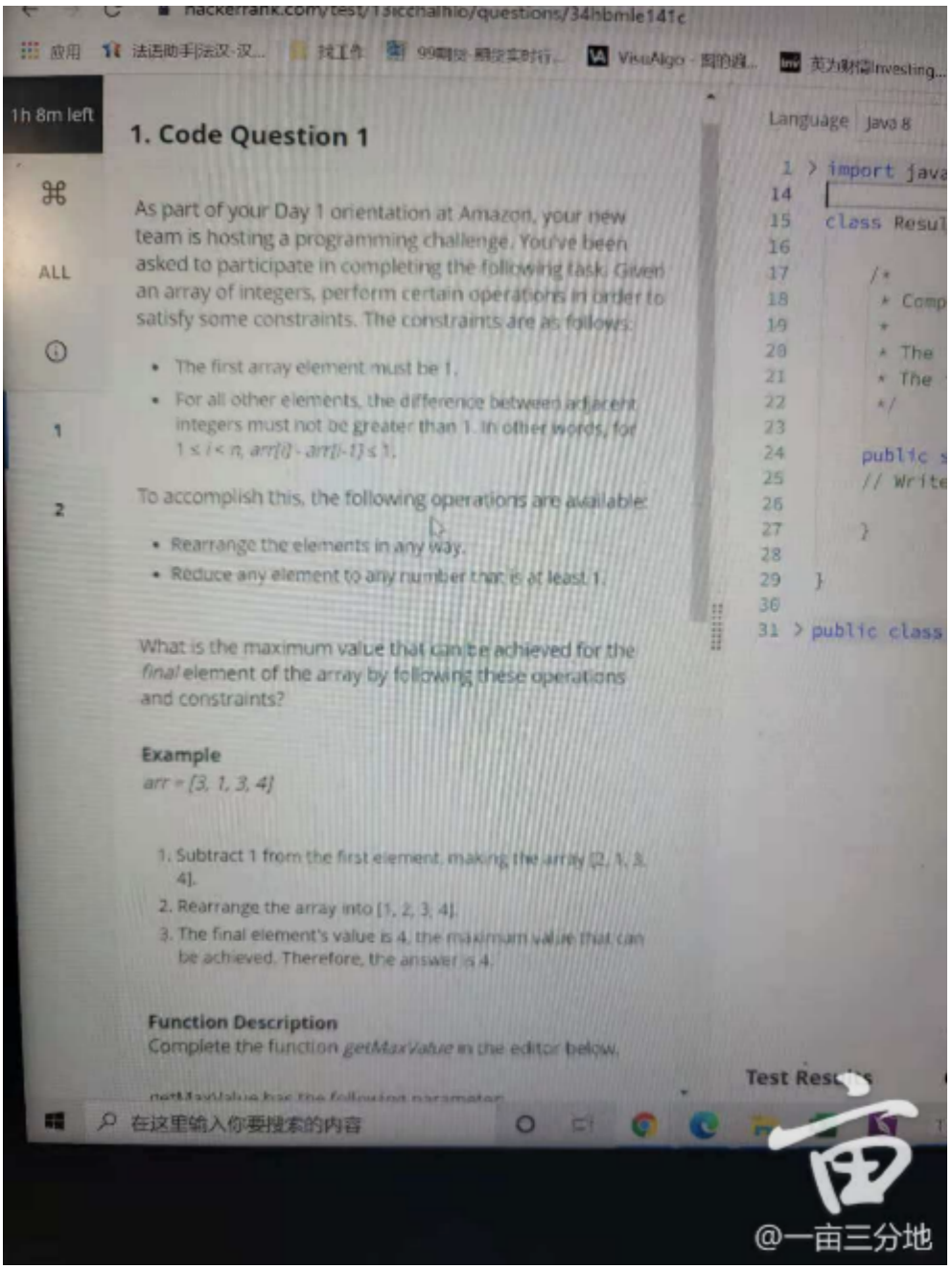
Given an array of integers jobDifficulty and an integer d. The difficulty of the i-th job is jobDifficulty[i].

Time O(nd)
Space O(n)

```
public int minDifficulty(int[] A, int D) {
    int n = A.length;
    if (n < D) return -1;
    int[] dp = new int[n], dp2 = new int[n], tmp;
    Arrays.fill(dp, 1000);
    Deque<Integer> stack = new ArrayDeque<Integer>();

    for (int d = 0; d < D; ++d) {
        stack.clear();
        for (int i = d; i < n; i++) {
            dp2[i] = i > 0 ? dp[i - 1] + A[i] : A[i];
            while (!stack.isEmpty() && A[stack.peek()] <= A[i]) {
                int j = stack.pop();
                dp2[i] = Math.min(dp2[i], dp2[j] - A[j] + A[i]);
            }
            if (!stack.isEmpty()) {
                dp2[i] = Math.min(dp2[i], dp2[stack.peek()]);
            }
            stack.push(i);
        }
        tmp = dp;
        dp = dp2;
        dp2 = tmp;
    }
```

```
        return dp[n - 1];
    }
```

Maximum Element After Decreasing and Rearranging



There are 2 types of operations that you can perform any number of times:
Decrease the value of any element of arr to a smaller positive integer.
Rearrange the elements of arr to be in any order.

Complexity
Time O(sort)
Space O(sort)

```
public int maximumElementAfterDecrementingAndRearranging(int[] A) {
```

```
        Arrays.sort(A);
        int pre = 0;
        for (int a: A)
            pre = Math.min(pre + 1, a);
        return pre;
    }
```

Count Binary Substrings- same number of 0 and 1

Give a binary string s, return the number of non-empty substrings that have the same number of 0's and 1's, and all the 0's and all the 1's in these substrings are grouped consecutively.

Substrings that occur multiple times are counted the number of times they occur.

Time Complexity: O(N), where NN is the length of s. Every loop is through O(N)O(N) items with O(1)O(1) work inside the for-block.

Space Complexity: O(N), the space used by groups.

```
public int countBinarySubstrings(String s) {
        int[] groups = new int[s.length()];
        int t = 0;
        groups[0] = 1;
        for (int i = 1; i < s.length(); i++) {
            if (s.charAt(i-1) != s.charAt(i)) {
                groups[++t] = 1;
            } else {
                groups[t]++;
            }
        }

        int ans = 0;
        for (int i = 1; i <= t; i++) {
            ans += Math.min(groups[i-1], groups[i]);
        }
        return ans;
    }
```

Caesar Cipher Encrpytion

You are given a list of string, group them if they are same after using Ceaser Cipher Encrpytion.

Definition of "same", "abc" can right shift 1, get "bcd", here you can shift as many time as you want, the string will be considered as same.

Caesar Cipher

Julius Caesar protected his confidential information by encrypting it using a cipher. Caesar's cipher shifts each letter by a number of letters. If the shift takes you past the end of the alphabet, just rotate back to the front of the alphabet. In the case of a rotation by 3, w, x, y and z would map to z, a, b and c.

```
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        scanner.nextLine();
        String input = scanner.nextLine();
        int k = scanner.nextInt();

        StringBuilder builder = new StringBuilder(input.length());
        for (int i = 0; i < n; i++) {
            char temp = input.charAt(i);
            boolean upperCase = Character.isUpperCase(temp);
            if (Character.isLetter(temp)) {
                temp += k%26;
                if (!Character.isLetter(temp) || (upperCase && !Character.isUpperCase(temp))) {
                    temp -= 26;
                }
            }
            builder.append(temp);
        }

        System.out.println(builder.toString());
    }
```

24 Game

You are given an integer array cards of length 4. You have four cards, each containing a number in the range [1, 9]. You should arrange the numbers on these cards in a mathematical expression using the operators ['+', '-', '*', '/'] and the parentheses '(' and ')' to get the value 24.

```
public boolean judgePoint24(int[] nums) {
        List<Double> list = new ArrayList<>();
        for (int i : nums) {
            list.add((double) i);
        }
        return dfs(list);
    }

    private boolean dfs(List<Double> list) {
        if (list.size() == 1) {
            if (Math.abs(list.get(0)- 24.0) < 0.001) {
                return true;
            }
```

```
            return false;
        }

        for(int i = 0; i < list.size(); i++) {
            for(int j = i + 1; j < list.size(); j++) {
                for (double c : generatePossibleResults(list.get(i), list.get(j))) {
                    List<Double> nextRound = new ArrayList<>();
                    nextRound.add(c);
                    for(int k = 0; k < list.size(); k++) {
                        if(k == j || k == i) continue;
                        nextRound.add(list.get(k));
                    }
                    if(dfs(nextRound)) return true;
                }
            }
        }
        return false;

    }

    private List<Double> generatePossibleResults(double a, double b) {
        List<Double> res = new ArrayList<>();
        res.add(a + b);
        res.add(a - b);
        res.add(b - a);
        res.add(a * b);
        res.add(a / b);
        res.add(b / a);
        return res;
    }
```

Swim in water 2D grid, 类似雨水问题，此时的雨水随时间变化
You have to reach a location as early as you can and the weather forecast predicts that it will rain heavily today. The surface in front of you has a very unique terrain. Assume it is a N X Nsurface, each square grid[i][j] represents the height (distance between surface and base) at that point(i, j).Also the boundary of this field is very high.

You have arranged a boat which is very fast. Your boat can cover an infinite distance in zero time. You start att=0 and the rain starts as soon as you start your journey (at t=0). At timet minutes, the height of the water everywhere is t units.

```
public int swimInWater(vector<vector<int>>& grid) {
        int n=grid.size();
        int m=grid[0].size();
        priority_queue< pair<int, pair<int,int>>, vector<pair<int, pair<int,int>>>,
greater<pair<int, pair<int,int>>>>pq;
        vector<vector<int>>dist(n,vector<int>(m,INT_MAX));
```

```
        dist[0][0]=0;
        pq.push({grid[0][0],{0,0}});
        int row[] = {1, -1, 0, 0};
        int col[] = {0, 0, 1, -1};

        while(!pq.empty()){
            auto p=pq.top();
            pq.pop();
            int t=p.first;

            int x=p.second.first;
            int y=p.second.second;

            for(int i=0;i<4;i++){
                int u=x+row[i];
                int v=y+col[i];
                if(u>=0 && v>=0 && u<n && v<m){
                    if(max(grid[u][v],t) < dist[u][v]){
                        dist[u][v]=max(grid[u][v],t);
                        pq.push({dist[u][v],{u,v}});
                    }

                }
            }
        }
        return dist[n-1][m-1];
    }
};
```

CalculateValues

Let's define a function as calculateValue(X) for allX>0 as follows:

• Write the number X in base 10 representation • Split the number X into contiguous subsequences such that the number subsequences and each subsequence is made up of identical digits

• For each subsequence, Let's say that the most significant digit it contains is the i-th digit, where i=0 corresponds to the least significant digit of X.

For example, if X = 12233322211 can be split into the following subsequences - "1", "22", "333", "222", "11", where i = 7 for the sequence "333" and i = 1 for
the subsequence "11"

• The value for each subsequence can be calculated as d x 10^i, where d is the digit repeated in the
sequence

• calculateValues(X) is calculated as the sum of all these values.

https://leetcode.com/discuss/interview-question/1397739/Amazon-OA-2022-New-Grad
https://discuss.codechef.com/t/encoding-editorial/33866/5

K Closest Points to Origin

Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k closest points to the origin (0, 0).

The distance between two points on the X-Y plane is the Euclidean distance (i.e., √(x1 - x2)2 + (y1 - y2)2).

You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in).

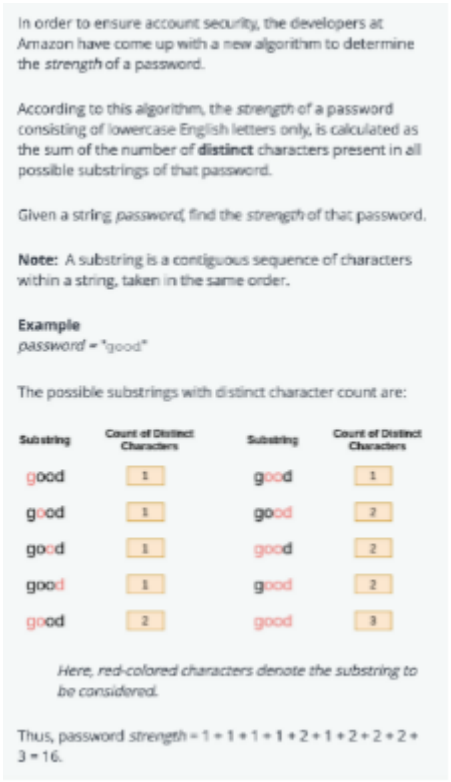
Time complexity: $O(N \cdot logN)$ for the sorting of points.

While sorting methods vary between different languages, most have a worst-case or average time complexity of $O(N \cdot logN)$.

Space complexity:O(logN) to O(N) for the extra space required by the sorting process.

```
public int[][] kClosest(int[][] points, int k) {
        // Sort the array with a custom lambda comparator function
        Arrays.sort(points, (a, b) -> squaredDistance(a) - squaredDistance(b));

        // Return the first k elements of the sorted array
        return Arrays.copyOf(points, k);
    }

    private int squaredDistance(int[] point) {
        // Calculate and return the squared Euclidean distance
        return point[0] * point[0] + point[1] * point[1];
    }
```

Find Password Strength



In order to ensure account security, the developers at Amazon have come up with a new algorithm to determine the *strength* of a password.

According to this algorithm, the *strength* of a password consisting of lowercase English letters only, is calculated as the sum of the number of **distinct** characters present in all possible substrings of that password.

Given a string *password*, find the *strength* of that password.

Note: A substring is a contiguous sequence of characters within a string, taken in the same order.

Example
password = "good"

The possible substrings with distinct character count are:

Substring	Count of Distinct Characters	Substring	Count of Distinct Characters
good	1	good	1
good	1	good	2
good	1	good	2
good	1	good	2
good	2	good	3

Here, red-colored characters denote the substring to be considered.

Thus, password *strength* = 1 + 1 + 1 + 1 + 2 + 1 + 2 + 2 + 2 + 3 = 16.

```
# Time: O(N)
# Space: O(1) [for variable character language of size M it would be O(M)]
def passwordStrength(password):
    last = {chr(97 + i): -1 for i in range(26)}
    ret = 0
    for i,ch in enumerate(password):
        left = i - last[ch]
        right = len(password) - i
        ret += left * right
        last[ch] = i

    return ret
```

https://leetcode.com/discuss/interview-question/1526418/Count-strength-of-pa%20%20ssword-or-amazon

解法：

```
public long passwordStrength(String s) {
        int n = s.length();
        int[] lastIndex = new int[26];
        Arrays.fill(last, -1);
        int[] dp = new int[n];
        int ans = 0;
        for (int i = 0; i < s.length(); i++) {
            int index = s.charAt(i) -'a';
            if(i == 0)
              dp[i] = 1;
            else{
              dp[i] = dp[i - 1] + i + 1 - (lastIndex[index] + 1);
            }
            ans += dp[i];
            last[index] = i;
        }
        return ans;
    }
```

Grid Climbing - Number of Connections

A grid with mrows and n columns is used to form a cluster of nodes. If a point in the grid has a value of 1, then it represents a node. Each node in the cluster has a level associated with it. A node located in the throw of the grid is a level inode. Here are the rules for creating a cluster: Every node at level /connects to all the nodes at level k where > i and kis the first level after level i that contains at least 1 node. • When i reaches the last level in the grid, no more connections are possible. . Given such a grid, find the number of connections present in the cluster.

Example: gridOfNodes = [[1, 1, 1], [0, 1, 0], [0, 0, 0],[1,1, 0]] .

```
int gridCluster(int[][] grid) {
        int ans = 0;
        int m = grid.length;
        ArrayList<Integer> arr = new ArrayList<>();

        for(int[] row: grid) {
            int sum = Arrays.stream(row).sum();
            if(sum != 0)
                arr.add(sum);
        }

        for(int i = 0; i < arr.size() - 1; i++)
            ans += (arr.get(i) * arr.get(i + 1));


        return ans;
    }
```

OR

```
int prev = 0;
int connections = 0;

for(int [] row: grid){
    int sum = 0;
    for(int n:row){
        sum +=n;
    }

    if(sum > 0){
        connections += prev *sum;
        prev = sum;
    }
}

return connections;
```

Sliding Window Maximum

You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

```
public int[] maxSlidingWindow(int[] nums, int k) {
    int n = nums.length;
    if (n * k == 0) return new int[0];
    if (k == 1) return nums;

    int [] left = new int[n];
    left[0] = nums[0];
    int [] right = new int[n];
    right[n - 1] = nums[n - 1];
    for (int i = 1; i < n; i++) {
      // from left to right
      if (i % k == 0) left[i] = nums[i];  // block_start
      else left[i] = Math.max(left[i - 1], nums[i]);

      // from right to left
      int j = n - i - 1;
      if ((j + 1) % k == 0) right[j] = nums[j];  // block_end
      else right[j] = Math.max(right[j + 1], nums[j]);
    }

    int [] output = new int[n - k + 1];
    for (int i = 0; i < n - k + 1; i++)
      output[i] = Math.max(left[i + k - 1], right[i]);
```

```
    return output;
  }
```

Count of Smaller Numbers After Self

You are given an integer array nums and you have to return a new counts array. The counts array has the property where counts[i] is the number of smaller elements to the right of nums[i].

Time Complexity: O(Nlog(N)). We need to perform a merge sort which takes O(Nlog(N)) time. All other operations take at most O(N)O(N) time.

Space Complexity: O(N), since we need a constant number of arrays of size O(N).

```
public List<Integer> countSmaller(int[] nums) {
        int n = nums.length;
        int[] result = new int[n];
        int[] indices = new int[n]; // record the index. we are going to sort this array
        for (int i = 0; i < n; i++) {
            indices[i] = i;
        }
        // sort indices with their corresponding values in nums, i.e., nums[indices[i]]
        mergeSort(indices, 0, n, result, nums);
        // change int[] to List to return
        List<Integer> resultToReturn = new ArrayList<Integer>(n);
        for (int i : result) {
            resultToReturn.add(i);
        }
        return resultToReturn;
    }

    private void mergeSort(int[] indices, int left, int right, int[] result, int[] nums) {
        if (right - left <= 1) {
            return;
        }
        int mid = (left + right) / 2;
        mergeSort(indices, left, mid, result, nums);
        mergeSort(indices, mid, right, result, nums);
        merge(indices, left, right, mid, result, nums);
    }

    private void merge(int[] indices, int left, int right, int mid, int[] result, int[] nums) {
        // merge [left, mid) and [mid, right)
        int i = left; // current index for the left array
        int j = mid; // current index for the right array
        // use temp to temporarily store sorted array
        List<Integer> temp = new ArrayList<Integer>(right - left);
        while (i < mid && j < right) {
            if (nums[indices[i]] <= nums[indices[j]]) {
```

```
            // j - mid numbers jump to the left side of indices[i]
            result[indices[i]] += j - mid;
            temp.add(indices[i]);
            i++;
        } else {
            temp.add(indices[j]);
            j++;
        }
    }
    // when one of the subarrays is empty
    while (i < mid) {
        // j - mid numbers jump to the left side of indices[i]
        result[indices[i]] += j - mid;
        temp.add(indices[i]);
        i++;
    }
    while (j < right) {
        temp.add(indices[j]);
        j++;
    }
    // restore from temp
    for (int k = left; k < right; k++) {
        indices[k] = temp.get(k - left);
    }
  }
```

Fill the truck - Maximum Units on a Truck

```
public int maximumUnits(int[][] boxTypes, int truckSize) {
    int ans = 0;
    int boxesLeft = truckSize;
    ArrayList<Map.Entry<Integer, Integer>> unitsBoxes = new ArrayList<>();
    for (int i = 0; i < boxTypes.length; i++)
        unitsBoxes.add(Map.entry(boxTypes[i][1], boxTypes[i][0]));
    unitsBoxes.sort(Comparator.comparing(e -> -e.getKey()));
    for (Map.Entry<Integer, Integer> e : unitsBoxes) {
        int units = e.getKey();
        int box = Math.min(boxesLeft, e.getValue());
        ans += box * units;
        boxesLeft -= box;
        if (boxesLeft == 0)
            break;
    }
    return ans;
 }
```

Maximum quality sent via channel

1h 37m left

ALL

1

2

3

4

1. Code Question 1

Amazon's AWS provides fast and efficient server solutions. The developers want to stress-test the quality of the servers' channels. They must ensure the following:

- Each of the packets must be sent via a single channel.
- Each of the channels must transfer at least one packet.

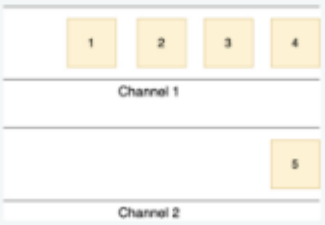
The *quality* of the transfer for a channel is defined by the median of the sizes of all the data packets sent through that channel.

Note: The median of an array is the middle element if the array is sorted in non-decreasing order. If the number of elements in the array is even, the median is the average of the two middle elements.

Find the maximum possible sum of the qualities of all channels. If the answer is a floating-point value, round it to the next higher integer.

Example
packets = [1, 2, 3, 4, 5]
channels = 2

At least one packet has to go through each of the *2* channels. One maximal solution is to transfer packets (1, 2, 3, 4) through channel 1 and packet (5) through channel 2.



Function Description
Complete the function *maximumQuality* in the editor below.

maximumQuality has the following parameter(s):
int packets[n]: the packet sizes
int channels: the number of channels

Returns
long int: the maximum sum possible

Constraints

- $1 \le len(packets) \le 5 * 10^5$
- $1 \le packets[i] \le 10^9$
- $1 \le channels \le len(packets)$

▸ **Input Format For Custom Testing**

▾ **Sample Case 0**

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
5       →   packets[] size n = 5
2       →   packets = [2, 2, 1, 5, 3]
2
1
5
3
2       →   channels = 2
```

Sample Output

```
7
```

Explanation
One solution is to send packet (5) through one channel and (2, 2, 1, 3) through the other. The sum of quality is 5 + (2 + 2)/2 = 7.

▸ **Sample Case 1**

▸ **Sample Case 2**

```
def median(ordered_list):
    if len(ordered_list)==1 : return ordered_list[0]
    if len(ordered_list) % 2 == 1:
        return ordered_list[int((len(ordered_list)+1)/2)]
    else:
        return math.ceil(ordered_list[len(ordered_list)/2]+ordered_list[len(lordered_list)/2+1])

quality = 0
if len(packets)< channels: return None
packets.sort()
for i in range(channels):
    quality += packets.pop()

return quality + median(packets)
```

解法：

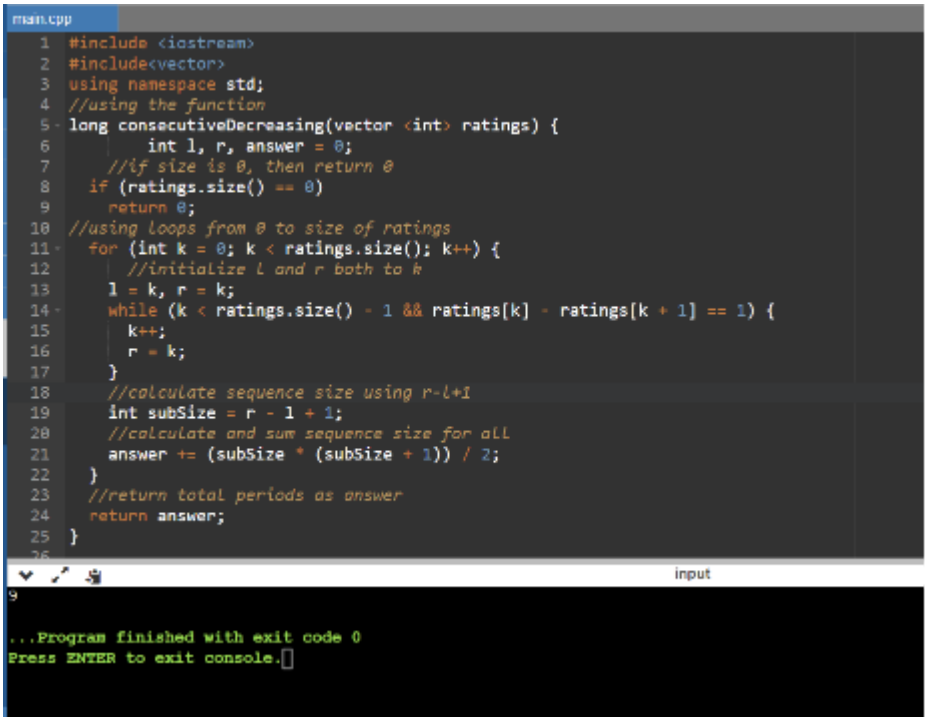
```
public long maxMedian(List<Integer> packets, int channel){
  Collections.sort(packet);
  long res = 0;
  while(channel > 1){
    res += packets.get(packets.size() - 1);
    packets.remove(packet.size() - 1);
  }
  int mid = packets.size() / 2;
  if(packets.size() % 2 != 0){
    res += (packets.get(mid));
  }
  else{
    res += (packets.get(mid - 1) + packets.get(mid) + 1) / 2;
  }
  return res;
}
```

Decreasing subarray（rating）

Amazon Shopping recently launched a new item whose daily customer ratings for n days is represented by the array ratings. They monitor these ratings to identify products that are not performing well. Find the number of groups that can be formed consisting of 1 or more consecutive days such that the rating continuously decreases over the days.
The rating is consecutively decreasing if it has the form: r r - 1, r- 2... and so on, where ris the rating on the first day of the group being considered. Two groups are considered different if they contain the ratings of different consecutive days.
Example
ratings = [4, 3, 5, 4, 3]
There are 9 periods in which the rating consecutively decreases.
5 one day periods: [4], [3], [5], [4], [3]
3 two day periods: [4, 3], [5, 4], [4, 3]
1 three day period: [5, 4, 3]
Return 9
Function Description
Complete the function countDecreasingRatings in the editor.
countDecreasingRatings contains one parameter:
int ratings[n]: customer ratings for n days
Returns
long. the number of valid grofips of ratings
Constraints
1 ≤ n≤ 105
0 < ratings[i] ≤ 109

Complete code in c++:

```
#include <iostream>
#include<vector>
using namespace std;
//using the function
long consecutiveDecreasing(vector <int> ratings) {

    int l, r, answer = 0;
   //if size is 0, then return 0
  if (ratings.size() == 0)
   return 0;

 //using loops from 0 to size of ratings
  for (int k = 0; k < ratings.size(); k++) {
     //initialize l and r both to k
   l = k, r = k;
   while (k < ratings.size() - 1 && ratings[k] - ratings[k + 1] == 1) {
    k++;
    r = k;
   }

   //calculate sequence size using r-l+1
   int subSize = r - l + 1;
   //calculate and sum sequence size for all
   answer += (subSize * (subSize + 1)) / 2;
  }
  //return total periods as answer
  return answer;
}
int main()
{
   //initializing the vector ratings with ( 4,3,5,4,3 )
   vector<int> ratings{ 4,3,5,4,3 };

   //call the function consecutiveDecreasing with parameter vector ratings
   cout<<consecutiveDecreasing(ratings);

   return 0;
}
```




```
def periodsOfDecreaseRating2(self, ratings):
    """
    two pointer O(n) appoach
    """
    res = 0
    i = 0
    m = len(ratings)

    for j in range(m):
        if j > 0 and ratings[j] < ratings[j - 1]:
            res += j - i + 1
        else:
            res += 1
            i = j
    return res
```

题目：**Subarray Common Prefix**

https://www.1point3acres.com/bbs/thread-824239-1-1.html
https://leetcode.com/discuss/interview-question/851513/roblox-new-grad-2021-oa-hackerrank

2， List<Integer> commonPrefix(List<String> inputs){}
给一个 string 的 list inputs，对于每一个 string， 在每个可能的地方进行切分，rightmost 的是 suffix，leftmost 是 prefix，比较切分后的 substring 和 original string 的 common prefix 的长度，计算每一个 string 的长度的和，最后返回 a list of sums.

例子，abcabcd,
第一次切分为 "", "abcabcd"，切分后 substring "abcabcd" 和 original string "abcabcd" 的 common prefix 的长度为 7
第二次切分为 " a ", " bcabcd "，切分后 substring "bcabcd" 和 original string "abcabcd" 的 common prefix 的长度为 0
第三次切分为 " ab ", " cabcd "，切分后 substring "cabcd" 和 original string "abcabcd" 的 common prefix 的长度为 0
第四次切分为 " abc ", " abcd "，切分后 substring "abcd" 和 original string "abcabcd" 的 common prefix 的长度为 3
第五次切分为 " abca ", " bcd "，切分后 substring "bcd" 和 original string "abcabcd" 的 common prefix 的长度为 0
第六次切分为 " abcab ", " cd "，切分后 substring "cd" 和 original string "abcabcd" 的 common prefix 的长度为 0
第六次切分为 " abcabc ", " d "，切分后 substring "d" 和 original string "abcabcd" 的 common prefix 的长度为 0
第七次切分为 " abcabd ", " "，切分后 substring "" 和 original string "abcabcd" 的 common prefix 的长度为 0

对于单个 string " abcabcd "，sum = 10;

答案1:

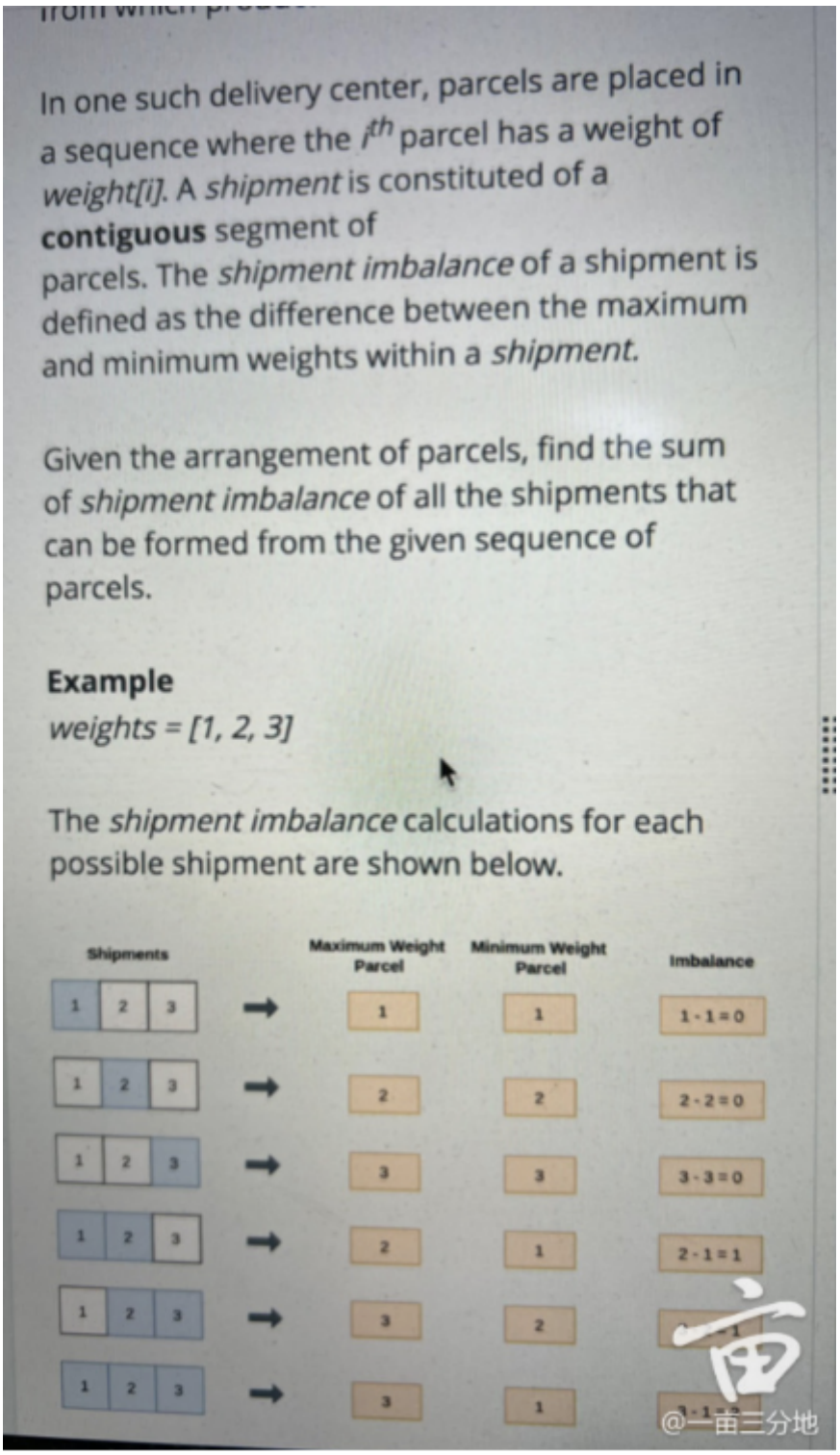
Q3 can be solved in O(n) using the Z algorithm. After the algorithm has completed the generated Z array will hold all common prefix lengths apart from n for the base case of removing nothing. So if we add those values together we get our result.

```
def common_prefix_length(s):
    n = len(s)
    z = [0] * n
    l = r = 0

    for i in range(1, n):
        if i <= r:
            z[i] = min(r - i + 1, z[i - l])
        while i + z[i] < n and s[z[i]] == s[i + z[i]]:
            z[i] += 1
        if i + z[i] - 1 > r:
            l, r = i, i + z[i] - 1

    return n + sum(z)


def test_common_prefix_length():
    assert common_prefix_length("abcabcd") == 10
    assert common_prefix_length("ababaa") == 11
    assert common_prefix_length("aa") == 3
```

答案2: **https://www.hackerrank.com/challenges/string-similarity/topics**

题目：**kth factor of n**
答案: **leetcode 1492 https://leetcode.com/problems/the-kth-factor-of-n/**

题目：**Delivery Shipment max and min subarray sum**


In one such delivery center, parcels are placed in a sequence where the i^{th} parcel has a weight of *weight[i]*. A *shipment* is constituted of a **contiguous** segment of parcels. The *shipment imbalance* of a shipment is defined as the difference between the maximum and minimum weights within a *shipment*.
Given the arrangement of parcels, find the sum of *shipment imbalance* of all the shipments that can be formed from the given sequence of parcels.
Example
weights = [1, 2, 3]
The *shipment imbalance* calculations for each possible shipment are shown below.
Shipments
Maximum Weight Parcel
Minimum Weight Parcel
Imbalance
1 2 3
1
1
1 - 1 = 0
1 2 3
2
2
2 - 2 = 0
1 2 3
3
3
3 - 3 = 0
1 2 3
2
1
2 - 1 = 1
1 2 3
3
2
1 2 3
3
1
@一亩三分地

方法 **1**, **2**:

```python
def sum_of_extremums(nums, use_maximum):
    compare = operator.le if use_maximum else operator.ge
    total = 0
    stack = []
    accumulated_sum = 0

    for index, num in enumerate(nums):
        while stack and compare(stack[-1][1], num):
            right, other_num = stack.pop()
            left = 0 if not stack else stack[-1][0] + 1
            accumulated_sum -= (right - left + 1) * other_num

        left = 0 if not stack else stack[-1][0] + 1
        accumulated_sum += (index - left + 1) * num
        total += accumulated_sum
        stack.append((index, num))

    return total

def sum_of_imbalances(weights):
    """
    >>> sum_of_imbalances([1,2,3])
    4
    """

    return sum_of_extremums(weights, True) - sum_of_extremums(weights, False)
```

```python
def sumOfSubArrayExtremes(weights, maxi):
    termEnds = float('-inf')
    compare = operator.gt
    if maxi:
        compare = operator.lt
        termEnds = float('inf')

    weights = [termEnds] + weights + [termEnds]
    stack = []
    resultSum = 0
    for idx in range(len(weights)):
        while stack and compare(weights[stack[-1]], weights[idx]):
            curr = stack.pop()
            resultSum += weights[curr] * (idx - curr) * (curr - stack[-1])

        stack.append(idx)
    return resultSum

def sumOfImbalances(weights):
    """
    >>> sumOfImbalances([1,2,3])
    4
    """
    return sumOfSubArrayExtremes(weights, maxi=True) - sumOfSubArrayExtremes(weights, maxi=False)
```

类似题目：**Count Number of distinct characters in all substrings**
答案**:**
https://leetcode.com/discuss/interview-question/1481915/Amazon-or-OA-or-Count-distinct-characters-in-all-substrings.

题目：**password strength 2 - substring with vowel and consonant**
https://leetcode.com/discuss/interview-question/1471459/Amazon-OA

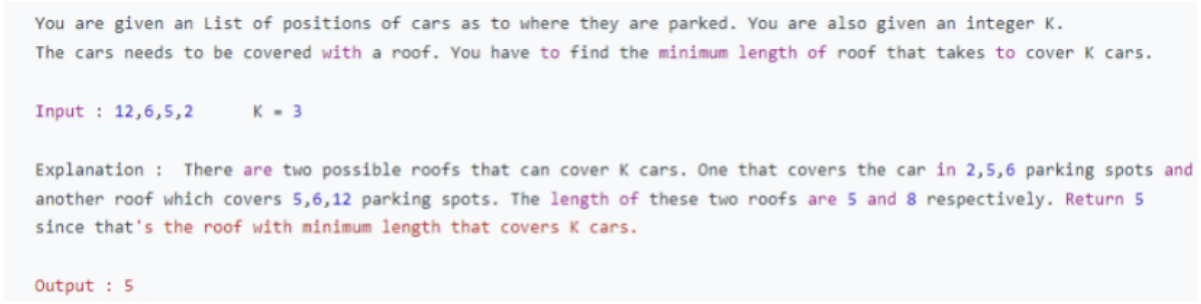
答案：https://leetcode.com/discuss/interview-question/1471459/Amazon-OA

题目： 反转（**swap**）**0-1 pairs**
https://leetcode.com/discuss/interview-question/1471459/Amazon-OA
答案：
https://leetcode.com/problems/flip-string-to-monotone-increasing/

题目： **drivers's rating find the contiuous sum * minimum rate over period**

答案：https://leetcode.com/discuss/interview-question/815454/amazon-oa-question-sde-1

题目：**Length of roof that covers k cars**



```
You are given an List of positions of cars as to where they are parked. You are also given an integer K.
The cars needs to be covered with a roof. You have to find the minimum length of roof that takes to cover K cars.

Input : 12,6,5,2      K = 3

Explanation :  There are two possible roofs that can cover K cars. One that covers the car in 2,5,6 parking spots and
another roof which covers 5,6,12 parking spots. The length of these two roofs are 5 and 8 respectively. Return 5
since that's the roof with minimum length that covers K cars.

Output : 5
```

答案：https://leetcode.com/discuss/interview-question/1317796/amazon-oa-2021-hackerrank
https://r24zeng.gitbook.io/amazon-prepare-2021/leetcode/26.-minimum-length-of-roof

题目：**Decode input string diagonally**
答案：https://leetcode.com/discuss/interview-question/1317796/amazon-oa-2021-hackerrank
https://r24zeng.gitbook.io/amazon-prepare-2021/leetcode/27.-decode-string

题目：**discount coupons**
答案：https://leetcode.com/discuss/interview-question/1518678/amazon-oa-sde2-seattle
https://leetcode.com/problems/valid-parentheses/
https://leetcode.com/discuss/interview-question/1566892/amazon-oa-valid-coupon

题目：Total number of Wheels

https://leetcode.com/discuss/interview-question/1365052/Amazon-OA

1. Code Question 1

Given an integer denoting a total number of wheels, help Amazon Logistics find the number of different ways to choose a fleet of vehicles from an infinite supply of two-wheeled and four-wheeled vehicles such that the group of chosen vehicles has that exact total number of wheels. Two ways of choosing vehicles are considered to be different if and only if they contain different numbers of two-wheeled or four-wheeled vehicles.

For example, if our array *wheels = [4,5,6]* our return array would be *res = [2, 0, 2]*. Case by case, we can have *1* four-wheel or *2* two-wheel to have *4* wheels. We cannot have *5* wheels. We can have *1* four-wheel and *1* two-wheel or *3* two-wheel vehicles in the final case.

Function Description

Complete the function *chooseFleets* in the editor below. The function should return an array of integers representing the answer for each *wheels[i]*.

chooseFleets has the following parameter(s):

wheels[wheels[0],...wheels[n-1]]: an array of integers

Constraints

- $1 \le n \le 10^5$
- $1 \le wheels[i] \le 10^6$

▸ Input Format for Custom Testing

▾ Sample Case 0

Sample Input 0

答案**1**:

Since the only options we have are two wheels and four wheels, the number of ways would only increase by 1 on next multiples of 4.

```
public int[] findVehicles(int[] vehicles) {
        for(int i =0; i<vehicles.length;i++) {
            if(vehicles[i]%2 != 0) {
                vehicles[i] = 0;
                continue;
            }
            int numOfWays = vehicles[i]/4 + 1;
            vehicles[i] = numOfWays;
        }
        return vehicles;
    }
```

答案**2**:

If the number is not a multiple of 2 then number of ways is 0. if it is, then the problems is to find the number of multiples of 4 in the range (for building 4 wheelers) + 1 for choosing all the vehicles to be 2 wheelers. Because number of ways depends like moving a pointer from the range choosing one part to be 2 wheelers and the other part to be 4 wheelers.

Time Complexity : O(N)

Space Complexity : O(1)

Java

Copy

```java
// "static void main" must be defined in a public class.

class Solution {
    public int[] findWays (int[] wheels) {
        for (int index = 0; index < wheels.length; index++) {
            wheels[index] = (wheels[index] % 2 != 0) ? 0: (wheels[index]/4) + 1;
        }
        return wheels;
    }
}

public class Main {
    public static void main(String[] args) {
        Solution solution = new Solution();
        int[] wheels = new int[]{0, 1, 2, 3, 4, 5, 6, 7, 8};
            System.out.println(
                IntStream.of(solution.findWays(wheels)).boxed().collect(Collectors.toList())
            );
    }
}
```

题目：**Demolition of Robot**

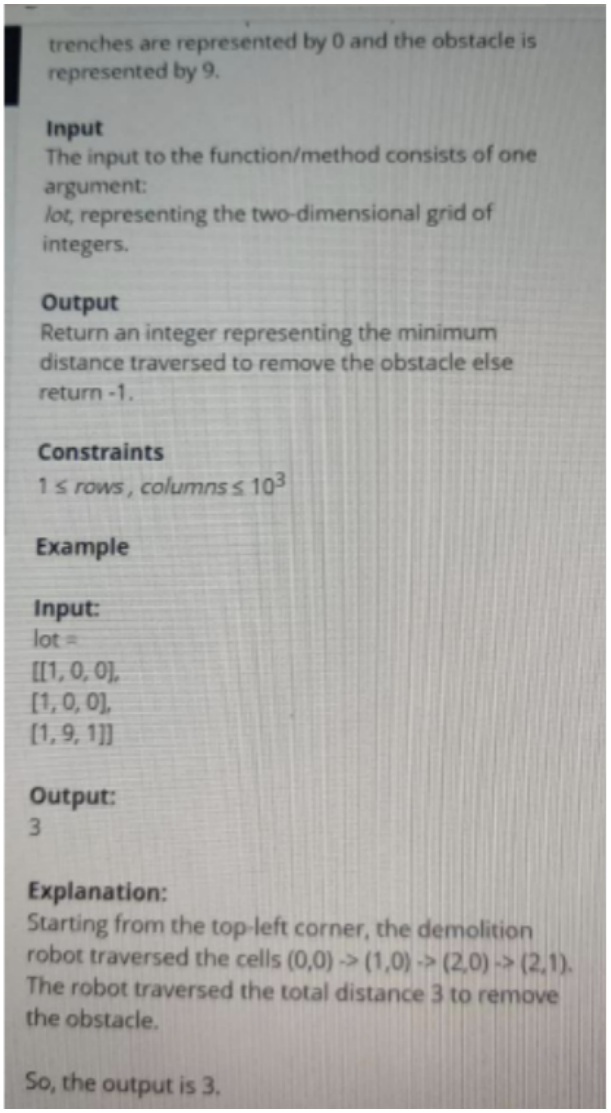
3. Demolition Robot

You are in charge of preparing a recently purchased lot for one of Amazon's new building. The lot is covered with trenches and has a single obstacle that needs to be taken down before the foundation can be prepared for the building. The demolition robot must remove the obstacle before progress can be made on the building.

Write an algorithm to determine the minimum distance required for the demolition robot to remove the obstacle.

Assumptions:
The lot is flat, except for trenches, and can be represented as a two-dimensional grid.
The demolition robot must start from the top-left corner of the lot, which is always flat, and can move one block up, down, left, or right at a time.
The demolition robot cannot enter trenches and cannot leave the lot.
The flat areas are represented as 1, areas with trenches are represented by 0 and the obstacle is represented by 9.

Input
The input to the function/method consists of one argument:
lot, representing the two-dimensional grid of integers.



trenches are represented by 0 and the obstacle is represented by 9.

Input
The input to the function/method consists of one argument:
lot, representing the two-dimensional grid of integers.

Output
Return an integer representing the minimum distance traversed to remove the obstacle else return -1.

Constraints
$1 \leq rows, columns \leq 10^3$

Example

Input:
lot =
[[1, 0, 0],
[1, 0, 0],
[1, 9, 1]]

Output:
3

Explanation:
Starting from the top-left corner, the demolition robot traversed the cells (0,0) -> (1,0) -> (2,0) -> (2,1). The robot traversed the total distance 3 to remove the obstacle.

So, the output is 3.

答案:
https://leetcode.com/discuss/interview-question/1257344/Amazon-OA-or-Demolition-of-Robot

题目：Validating String With RegEx
https://leetcode.com/discuss/interview-question/1473342/Amazon-OA

2. Validating Strings with RegEx

Given a list of strings made up of the characters 'a' and 'b', create a regular expression that will match strings that begin and end with the same letter.

Example
'*a*', '*aa*', and '*bababbb*' match.
'*ab*' and '*baba*' do not match.

Replacing the blank (i.e., "_____") with a regular expression that matches strings as described. Locked code in the editor prints *True* for each correct match and *False* for each incorrect match.

Constraints

- $1 \leq query \leq 10^3$
- $1 \leq |string| \leq 10^3$
- Each character $string[i] \in \{a,b\}$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function:

The first line contains an integer *query*, the number of strings to be tested.
Each of the next *query* lines contains a string to validate.

▼ Sample Case 0

Sample Input

```
STDIN       Function
-----       -----
5       →   number of strings to be tested, query = 5
a       →   query Strings = 'a', 'b', 'ab', 'ba',
'aba'
b
ab
```

答案**1**:

This should be the NG-OA demo/practice question, I got this 30/09/2021,
The solution should be: r'^[a-z]$|^([a-z]).*\1$'

答案**2**:

Simple and Intuitive answer : `( a | b | (a.*a) | (b.*b) )`
Notes:

- `.*` can match anything, including empty string.
- `|` stands for OR

答案**3**:

delicia11 ★ 1 October 2, 2021 1:31 PM

it should be r'^(.).*\1$|[ab]$'

0 Reply

tbag_njit ★ 0 September 24, 2021 2:35 PM

I think this should work : ^([ab]).*\1$

题目：**Turnstile**

9. Turnstile

A university has exactly one turnstile. It can be used either as an exit or an entrance. Unfortunately, sometimes many people want to pass through the turnstile and their directions can be different. The i^{th} person comes to the turnstile at *time[i]* and wants to either exit the university if *direction[i] = 1* or enter the university if *direction[i] = 0*. People form *2* queues, one to exit and one to enter. They are ordered by the time when they came to the turnstile and, if the times are equal, by their indices.

If some person wants to enter the university and another person wants to leave the university at the same moment, there are three cases:

- If in the previous second the turnstile was not used (maybe it was used before, but not at the previous second), then the person who wants to leave goes first.
- If in the previous second the turnstile was used as an exit, then the person who wants to leave goes first.
- If in the previous second the turnstile was used as an entrance, then the person who wants to enter goes first.

Passing through the turnstile takes *1* second.

For each person, find the time when they will pass through the turnstile.

Function Description

Complete the function *getTimes* in the editor below.

getTimes has the following parameters:

int time[n]: an array of *n* integers where the value at index *i* is the time in seconds when the i^{th} person will come

答案：
https://leetcode.com/discuss/interview-question/699973/Goldman-Sachs-or-OA-or-Turnstile

题目：You are given an array with random numbers `[1,4,6,7,6]`. Return an array in the same order but with minimized values: `[1,2,3,4,3]`. So since 4 is greater than 1 but less than 6 it becomes 2, 6 is greater than 4 so it's value should be 3. `[3,5,3,8]` would become `[1,2,1,3]`. My guess is we need to somehow calculate the value that then would be subtracted from each number in the array.

https://leetcode.com/discuss/interview-question/1355972/Amazon-OA

答案：**Leetcode 1331**

```
public int[] arrayRankTransform(int[] arr) {
        int[] A = Arrays.copyOf(arr, arr.length);
        Arrays.sort(A);
        HashMap<Integer, Integer> rank = new HashMap<>();
        for (int x : A)
          rank.putIfAbsent(x, rank.size() + 1);
        for (int i = 0; i < arr.length; ++i)
          A[i] = rank.get(arr[i]);
        return A;
    }
```

题目：Secret Array

https://leetcode.com/discuss/interview-question/1332322/amazon-online-assessment-july-2021-secret-array

An array is said to be analogous to the secret array if all of the following conditions are true:

- The length of the array is equal to the length of the secret array.
- Each integer in the array lies in the interval `[lowerBound, upperBound]`.
- The difference between each pair of consecutive integers of the array must be equal to the difference between the respective pair of consecutive integers in the secret array. In other words, let the secret array be *[s[0], s[1],..., s[n-1]]* and let the analogous array be *[a[0], a[1],..., a[n-1]]*, then *(a[i-1] - a[i])* must be equal to *(s[i-1] - s[i])* for each i from 1 to n -1.

Given the value of integers *lowerBound* and *upperBound*, inclusive, and the array of differences between each pair of consecutive integers of the secret array, find the number of arrays that are analogous to the secret array. If there is no array analogous to the secret array, return 0.

For example:

consecutiveDifference = [-2, -1, -2, 5]
lowerBound = 3
upperBound = 10

Note that none of the values is out of the bound. All possible analogous arrays are:
[3, 5, 6, 8, 3]
[4, 6, 7, 9, 4]
[5, 7, 8, 10, 5]

Tha answer is *3*.

答案：
https://leetcode.com/discuss/interview-question/1332322/amazon-online-assessment-july-2021-secret-array

题目：**0**，**1**，**9** 组成的**2D grids** 找到**9**的路径
类似岛屿问题

Given a 2D grid containing 1's, 0's and 9, find the shortest distance between (0,0) and the position that has value 9. 0's are obstacles and we can move horizontally as well as vertically.

https://leetcode.com/discuss/interview-question/1248182/Amazon-SDE1-OA

类似答案：
https://leetcode.com/problems/shortest-path-to-get-food/
https://leetcode.com/problems/rotting-oranges/

题目：**Flight Media –**
- basically two sums, but you had to account for multiple pairs that could add up to the target
https://leetcode.com/discuss/interview-question/1273052/Amazon-OA-or-SDE2
https://leetcode.com/discuss/interview-question/313719/Amazon-or-Online-Assessment-2019-or-Movies-on-Flight

题目：**Subscriber Groups - similar to https://leetcode.com/problems/number-of-provinces/, find the number of unique friend groups**

题目：shopping options -类似LC 4 Sum II problem

A customer wants to buy a pair of jeans, a pair of shoes, a skirt, and a top but has a limited budget in dollars. Given different pricing options for each product, determine how many options our customer has to buy 1 of each product. You cannot spend more money than the budgeted amount.

Example
priceOfJeans = [2, 3]
priceOfShoes = [4]
priceOfSkirts = [2, 3]
priceOfTops = [1, 2]
budgeted = 10

The customer must buy shoes for 4 dollars since there is only one option. This leaves 6 dollars to spend on the other 3 items. Combinations of prices paid for jeans, skirts, and tops respectively that add up to 6 dollars or less are [2, 2, 2], [2, 2, 1], [3, 2, 1], [2, 3, 1]. There are 4 ways the customer can purchase all 4 items.

Function Description

Complete the getNumberOfOptions function in the editor below. The function must return an integer which represents the number of options present to buy the four items.

getNumberOfOptions has 5 parameters:
int[] priceOfJeans: An integer array, which contains the prices of the pairs of jeans available.
int[] priceOfShoes: An integer array, which contains the prices of the pairs of shoes available.
int[] priceOfSkirts: An integer array, which contains the prices of the skirts available.
int[] priceOfTops: An integer array, which contains the prices of the tops available.
int dollars: the total number of dollars available to shop with.

Constraints

1 ≤ a, b, c, d ≤ 103
1 ≤ dollars ≤ 109
1 ≤ price of each item ≤ 109
Note: a, b, c and d are the sizes of the four price arrays

答案:

https://leetcode.com/discuss/interview-question/1031663/Amazon-OA

题目：robot-bounded-in-circle

答案：https://leetcode.com/problems/robot-bounded-in-circle/

题目：**Mean of K largest Top Elements - find median from data stream**
答案：https://leetcode.com/problems/find-median-from-data-stream/solution/

题目：**Word break - add spaces to form valid words**
答案：https://leetcode.com/problems/word-break-ii/

题目：**Number of unique integers after remove k elements**
答案：https://leetcode.com/problems/least-number-of-unique-integers-after-k-removals/

题目：**Reorder Data in Log files**
答案：https://leetcode.com/problems/reorder-data-in-log-files/

题目：**given a list of forward routes and reverse routes and the maxTravelAllowed: return the most optimal pairs**

Problem:
This problem is a variant of closest pair sum. You'll be given two arrays
arr1 = `[ [1, 2000], [2, 3000], [3, 4000] ]`
arr2 = `[ [ 1, 5000 ], [2, 3000] ]`
the first element of every pair represents *id* and the second value represents the *value*.
and a target `x = 5000`
Find the pairs from both the arrays whose vaue add upto a sum which is less than given target and should be close to the target.

Output for the above example:
`[ [1, 2] ]` // Note that the output should be in id's

答案：https://leetcode.com/discuss/interview-question/1459915/Amazon-or-OA-or-Prime-air-time
Binary search
https://leetcode.com/discuss/interview-question/1025705/Amazon-or-OA-or-Prime-Air-time
https://leetcode.com/playground/TZCQh49C
https://leetcode.com/discuss/interview-question/373202

题目：**Exam marks**
https://leetcode.com/discuss/interview-question/1363152/Amazon-OA-Question-or-2021
答案：类似https://leetcode.com/problems/partition-array-for-maximum-sum/

题目： **Two-merge sort**
答案：https://leetcode.com/discuss/interview-question/1236946/2D-Mergesort
https://sapphireengine.com/

题目：**employment tree [NO ANS]**

Question 1 :

Imagine that an employment tree represents the formal employee hierarchy at Amazon. Manager nodes have chid nodes for each employee that reports to them; each of these employees can, in turn, have child nodes representing their respective reportees. Each node in the tree contains an integer representing the number of months the employee has spent at the company. Team tenure is computed as the average tenure of the manager and all the company employees working below the manager. The oldest team has the highest team tenure.

Write an algorithm to find the manager of the team with the highest tenure. An employee must have child nodes to be a manager.

Input
The input to the function/method consists of an argument -
president, a node representing the root node of the employee hierarchy.

Output
Return the node which has the oldest team.

Note
There will be at least one child node in the tree and there will be no ties.

Example

```
Input
   President =
                  20
           12              18

       11    2   3       15    8

Output = 18
```

Explanation :
There are three managers in this tree with the following team tenures :
12 => (11+2+3+12)/4 = 7
18 => (18+15+8)/3 = 13.67
20 => (12+11+2+3+18+15+8+20)/8 = 11.125

答案：
https://leetcode.com/discuss/interview-question/797541/amazon-online-assessment-2-sde-1-new-graduate-2021-coding-2-questions-with-solutions

题目： **Longest increasing subsequence**
答案：**Leetcode 300**