

Twitter Online Assessment (OA) 2021 | Weird Faculty

```
public static int exam(List<Integer> v) {  
    int totalSum = 0;  
    for(int score: v) {  
        if (score == 0) totalSum -= 1;  
        else totalSum += 1;  
    }  
  
    int currSum = 0;  
    for(int i = 0; i < v.size(); i++) {  
        if (currSum > totalSum) return i;  
        currSum += v.get(i) == 0 ? -1 : 1;  
        totalSum -= v.get(i) == 0 ? -1 : 1;  
    }  
    return v.size();  
}
```

Twitter Online Assessment (OA) - Unique Twitter User ID Set

```
public int minIncrementForUnique(int[] nums) {  
    Arrays.sort(nums);  
    int sum = nums[0];  
    int low = nums[0];  
    for(int i=1; i<nums.length; i++){  
        if(low<nums[i]){  
            low = nums[i];  
        }else{  
            low++;  
        }  
        sum += low;  
    }  
    return sum;  
}
```

945. Minimum Increment to Make Array Unique

```

public int minIncrementForUnique(int[] nums) {
    Arrays.sort(nums);
    int step = 0;
    int low = nums[0];
    for(int i=1;i<nums.length;i++){
        if(low<nums[i]){
            low = nums[i];
        }else{
            low++;
            step+=low-nums[i];
        }
    }
    return step;
}

```

Twitter Online Assessment (OA) - K-different Pairs in an Array

532. K-diff Pairs in an Array

Need to be unique!

Sequence does not matter!

```

public int findPairs(int[] nums, int k) {

    int result = 0;

    HashMap <Integer,Integer> counter = new HashMap<>();
    for (int n: nums) {
        counter.put(n, counter.getOrDefault(n, 0)+1);
    }

    for (Map.Entry <Integer, Integer> entry: counter.entrySet()) {
        int x = entry.getKey();
        int val = entry.getValue();
        if (k > 0 && counter.containsKey(x + k)) {
            result++;
        } else if (k == 0 && val > 1) {
            result++;
        }
    }
    return result;
}

```

Twitter Online Assessment (OA) 2021 | Efficient Job Processing Service

```

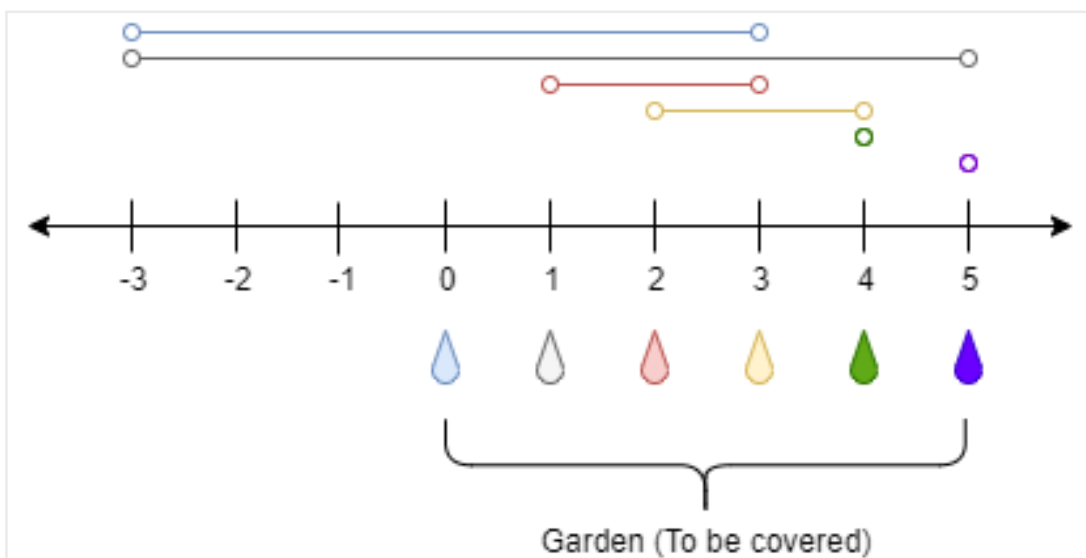
public class EfficientJobProcessingService_14 {
    int[][] dp = new int[tasks.length + 1][p / 2 + 1]; // task will be processed double of weight, hence halve p
    // for (int i = 0; i < tasks.length; i++) {
    //     tasks[i] *= 2;
    // }

    for (int i = 1; i < dp.length; i++) {
        for (int j = 1; j < dp[0].length; j++) {
            if (j < tasks[i - 1]) {
                dp[i][j] = dp[i - 1][j];
            } else {
                dp[i][j] = Math.max(dp[i - 1][j], dp[i - 1][j - tasks[i - 1]] + weights[i - 1]);
            }
        }
    }

    return dp[tasks.length][p / 2];
}

```

Twitter Online Assessment (OA) 2021 | Activate Fountain 1326. Minimum Number of Taps to Open to Water a Garden



Trans into Jumping Game!

First trans garden arr

How to treat 0?

```

int[] newRanges = new int[ranges.length];
for(int i=0;i<ranges.length;i++){
    if(ranges[i]==0) continue;
    int idx = Math.max(0,i-ranges[i]);
    int val = Math.min(n,i+ranges[i]);
    newRanges[idx] = Math.max(val,newRanges[idx]);
}

```

Jumping Game II

```

int farthest = 0;
int step = 0;
int end = 0;
for(int i=0;i<newRanges.length;i++){
    if(farthest<i) return -1;
    farthest = Math.max(farthest,newRanges[i]);
    if(end==i){
        end = farthest;
        step++;
    }
}
return step-1;

```

!! Pay attention to the final step!

Twitter Online Assessment (OA) 2021 | Partition Array

"make sure numbers.length is divisible by k and no element appears more than numbers.length/k times".

```

public static boolean partitionArrayUnique(int[] nums, int k){
    if(nums.length % k != 0){
        return false;
    }

    HashMap<Integer, Integer> map = new HashMap<>();
    int max = 0;
    for(int num: nums){
        map.put(num, map.getOrDefault(num, 0) + 1);
        if(map.get(num) > max){
            max = map.get(num);
        }
    }

    return max <= (nums.length / k);
}

```

Amazon | OA 2020 | Transaction logs

```

static List<String> processLogs(List<String> logs, int threshold) {
    Map<String, Integer> map = new HashMap<>();
    for (String logLine : logs) {
        String[] log = logLine.split(" ");
        map.put(log[0], map.getOrDefault(log[0], 0) + 1);
        if (log[0] != log[1]) {
            map.put(log[1], map.getOrDefault(log[1], 0) + 1);
        }
    }

    List<String> userIds = new ArrayList<>();
    for (Map.Entry<String, Integer> entry : map.entrySet()) {
        if (entry.getValue() >= threshold) {
            userIds.add(entry.getKey());
        }
    }

    Collections.sort(userIds, new Comparator<String>() {
        @Override
        public int compare(String s1, String s2) {
            return Integer.parseInt(s1) - Integer.parseInt(s2);
        }
    });

    return userIds;
}

```

Twitter Online Assessment (OA) 2021 | Game Events

```
1 class Result {
2     static class Score{
3         int actualTime;
4         String timeString;
5         String teamName;
6         String playerName;
7         String substituteName;
8         char eventType;
9         boolean isFirstHalf;
10        public Score(int actualTime, String timeString, String teamName,String playerName,String substituteName,char eventType, boolean isFirstHalf){
11            this.actualTime = actualTime;
12            this.timeString = timeString;
13            this.teamName = teamName;
14            this.playerName = playerName;
15            this.substituteName = substituteName;
16            this.eventType = eventType;
17            this.isFirstHalf = isFirstHalf;
18        }
19        public String toString(){
20            return actualTime + " " + timeString + " " + teamName + " " + playerName + " " + substituteName + " " + eventType +
21                " " + isFirstHalf;
22        }
23        public String getOutputString(){
24            return this.teamName + " " + this.playerName + " " + this.timeString + " " + this.eventType + " " + this.substituteName;
25        }
26    }
27    static Map<Character, Integer> map = new HashMap<>();
28    public List<String> getEventsOrder(String team1, String team2, List<String> events1, List<String> events2) {
29        map.put('G', 1);
30        map.put('Y', 2);
31        map.put('R', 3);
32        map.put('S', 4);
33        List<Score> scores = new ArrayList<>();
34        for(String e1: events1){
35            Score score = parseString(e1, team1);
36            scores.add(score);
37            System.out.println(score);
38        }
39        for(String e2: events2){
40            Score score = parseString(e2, team2);
41            scores.add(score);
42            System.out.println(score);
43        }
44        Collections.sort(scores, new Comparator<Score>(){
45            public int compare(Score s1, Score s2){
46                if(s1.isFirstHalf== true && s2.isFirstHalf==false){
47                    return -1;
48                }
49                if(s1.isFirstHalf== false && s2.isFirstHalf==true){
50                    return 1;
51                }
52                if(s1.actualTime != s2.actualTime)
53                    return s1.actualTime - s2.actualTime;
54                if(map.get(s1.eventType) == map.get(s2.eventType)){
55                    return map.get(s1.eventType) - map.get(s2.eventType);
56                }
57            }
58        });
59        List<String> answer = new ArrayList<>();
60        for(Score score: scores){
61            answer.add(score.getOutputString().trim());
62        }
63        return answer;
64    }
65    public static Score parseString(String str, String team){
66        String[] words = str.split(" ");
67        int time = getTimeIndex(words);
68        char event = words[time+1].charAt(0);
69        String player = "";
70        for(int i=0;i<time;i++){
71            player = player + " " + words[i];
72        }
73        player = player.trim();
74        String sub = "";
75        if(event=='S'){
76            for(int i = time+2;i<words.length;i++){
77                sub += words[i] + " ";
78            }
79            sub = sub.trim();
80        }
81        int actualTime = 0;
82        boolean isFirstHalf = false;
83        if(words[time].contains("+")){
84            String timeSplit[] = words[time].split("\\+");
85            actualTime += Integer.parseInt(timeSplit[0]);
86            if(actualTime <= 45){
87                isFirstHalf = true;
88            }
89            actualTime += Integer.parseInt(timeSplit[1]);
90        }else{
91            actualTime += Integer.parseInt(words[time]);
92            if(actualTime <= 45){
93                isFirstHalf = true;
94            }
95        }
96        Score score = new Score(actualTime, words[time], team, player, sub, event, isFirstHalf);
97        return score;
98    }
99    public static int getTimeIndex(String[] words){
100        for(int i = 0;i<words.length;i++){
101            if(words[i].charAt(0)>='0' && words[i].charAt(0)<='9'){
102                return i;
103            }
104        }
105        return -1;
106    }
107 }
108
109
110
111
```

```
56 }
57 }
58 if(s1.teamName.equals(s2.teamName))
59     return s1.teamName.compareTo(s2.teamName);
60 return s1.playerName.compareTo(s2.playerName);
61 }
62 List<String> answer = new ArrayList<>();
63 for(Score score: scores){
64     answer.add(score.getOutputString().trim());
65 }
66 return answer;
67 }
68 public static Score parseString(String str, String team){
69     String[] words = str.split(" ");
70     int time = getTimeIndex(words);
71     char event = words[time+1].charAt(0);
72     String player = "";
73     for(int i=0;i<time;i++){
74         player = player + " " + words[i];
75     }
76     player = player.trim();
77     String sub = "";
78     if(event=='S'){
79         for(int i = time+2;i<words.length;i++){
80             sub += words[i] + " ";
81         }
82         sub = sub.trim();
83     }
84     int actualTime = 0;
85     boolean isFirstHalf = false;
86     if(words[time].contains("+")){
87         String timeSplit[] = words[time].split("\\+");
88         actualTime += Integer.parseInt(timeSplit[0]);
89         if(actualTime <= 45){
90             isFirstHalf = true;
91         }
92         actualTime += Integer.parseInt(timeSplit[1]);
93     }else{
94         actualTime += Integer.parseInt(words[time]);
95         if(actualTime <= 45){
96             isFirstHalf = true;
97         }
98     }
99     Score score = new Score(actualTime, words[time], team, player, sub, event, isFirstHalf);
100     return score;
101 }
102 public static int getTimeIndex(String[] words){
103     for(int i = 0;i<words.length;i++){
104         if(words[i].charAt(0)>='0' && words[i].charAt(0)<='9'){
105             return i;
106         }
107     }
108     return -1;
109 }
110 }
111
```

Birthday Card collection

Disk Space Analysis

```

int segment(int x, vector<space>){
    int n = space.size();
    if(n==1)
        return space[0];

    int count = 0;
    int currMin = INT_MAX;
    int globMax = -1;
    for(int i=0; i<n; i++){
        if(count<x){
            currMin = min(currMin, space[i]);
        }else{
            globMax = max(globMax, currMin);
            if(space[i-count]==currMin){
                currMin = space[i-count+1];
                int j = i-count+1;
                while(j<=i){
                    currMin = min(currMin, space[j]);
                    j++;
                }
            }else{
                currMin = min(currMin, space[i]);
            }
        }
    }
    globMax = globMax == -1?currMin:globMax;
    return globMax;
}

```

Balancing Parantheses

```

public int minAddToMakeValid(String s) {
    int left = 0;
    int ans = 0;
    for(char each:s.toCharArray()){
        if(each=='('){
            left++;
        }else{
            if(left==0){
                ans++;
                continue;
            }else{
                left--;
            }
        }
    }
    return ans+left;
}

```

String Reduction

Eg : s = "abab" substr = { a , b , ab , ba , aba , bab , abab }.

output : Delete one 'a' and one 'b' -> return 2

– return len(string) - len(set(string)) -> one liner

- How many sentences?
- Largest lexicographical string with at most K consecutive elements


```

static String getLargestString(String s,int k){

int []frequency_array = new int[26];
for (int i = 0;i < s.length(); i++) frequency_array[s.charAt(i) - 'a']++;

String ans = "";

for (int i = 25; i >= 0;i--){
    if (frequency_array[i] > k)
    {
        int temp = k;
        String st = String.valueOf((char)(i + 'a'));
        while (temp > 0)
        {
            ans += st;
            temp--;
        }

        frequency_array[i] -= k;
        int j = i - 1;

        while (frequency_array[j] <= 0 &&
                j >= 0)
        {
            j--;
        }
        if (frequency_array[j] > 0 &&
            j >= 0)
        {
            String str = String.valueOf((char)(j + 'a'));
            ans += str;
            frequency_array[j] -= 1;
        }
        else
        {
            break;
        }
    }
    else if (frequency_array[i] > 0)
    {
        int temp = frequency_array[i];
        frequency_array[i] -= temp;
        String st = String.valueOf((char)(i + 'a'));
        while (temp > 0)
        {
            ans += st;
            temp--;
        }
    }
}
}

```

```
while (temp > 0)
{
    ans += st;
    temp--;
}
}
else
{
    i--;
}
}
return ans;
}
```

Twitter | OA 2019 | Get Set Go

```

private static boolean isPossibleDfs(int[] nums, int target) {
    return dfs(nums, target, 0);
}

private static boolean dfs(int[] nums, int target, int i) {
    if(target < 0 || i >= nums.length)
        return false;
    if(target == 0)
        return true;
    if(dfs(nums, target - nums[i], i+1) || dfs(nums, target, i+1))
        return true;
    return false;
}

private static boolean isPossible(int[] nums, int target) {
    Arrays.sort(nums);
    boolean[][] dp = new boolean[nums.length+1][target + 1];
    for(int i=0;i<dp.length;i++) {
        dp[i][0] = true;
    }
    for(int i=1;i<dp.length;i++) {
        for(int j=1;j<dp[0].length;j++) {
            if(j >= nums[i-1])
                dp[i][j] |= dp[i-1][j] | dp[i-1][j-nums[i-1]];
            else
                dp[i][j] |= dp[i-1][j];
        }
    }
    return dp[dp.length-1][dp[0].length-1];
}

```

Twitter | OA 2019 | Final Discounted Price

```
private static void getTotalCost(int[] prices) {  
    int[] tmp = new int[prices.length];  
    for(int i=0;i<tmp.length;i++) {  
        tmp[i] = prices[i];  
    }  
    Stack<Integer> s = new Stack<>();  
    for(int i=0;i<prices.length;i++) {  
        while(!s.isEmpty() && prices[s.peek()] >= prices[i]) {  
            int pre = s.pop();  
            tmp[pre] = prices[pre] - prices[i];  
        }  
        s.push(i);  
    }  
    int res = 0;  
    for(int t : tmp)  
        res += t;  
    System.out.println(res);  
    System.out.println(Arrays.toString(tmp));  
}
```

Twitter | OA 2019 | Authentication Tokens

```

public static int numberOfTokens(int expiryLimit, List<List<Integer>> commands) {
    if(commands == null || commands.isEmpty()) {
        //invalid input
        return 0;
    }
    //maintain tokenId with expiry in a map
    Map<Integer, Integer> tokenIdToTokenExpiry = new HashMap<>();

    for (List<Integer> token : commands) {
        if(token.size() != 3){
            //invalid input
            return 0;
        }
        //tokenCommand can be either 0 (get) or 1 (reset)
        Integer tokenCommand = token.get(0);
        Integer tokenId = token.get(1);
        Integer tokenTime = token.get(2);

        if(tokenCommand == 0){
            // Get command
            tokenIdToTokenExpiry.put(tokenId, tokenTime + expiryLimit);
        } else {
            //Reset command
            if(tokenIdToTokenExpiry.containsKey(tokenId)){
                if(tokenTime <= tokenIdToTokenExpiry.get(tokenId)){
                    //If not expired, update token time with new value
                    tokenIdToTokenExpiry.put(tokenId, tokenTime + expiryLimit);
                } else {
                    //if expired, remove token from map
                    tokenIdToTokenExpiry.remove(tokenId);
                }
            }
        }
    }

    //find the last inputed tokentime and filter data based on expiry
    Integer lastTime = commands.get(commands.size() - 1).get(2);
    return (int) tokenIdToTokenExpiry.values().stream().filter(tokenTime -> tokenTime >= lastTime).count();
}

```

Twitter | OA 2019 | Parking Dilemma

```

def ParkingDilemma(self, cars, k):
    # write your code here
    cars.sort()
    n = len(cars)
    res = float('inf')
    for i in range(n-k+1):
        res = min(res, cars[i+k-1] - cars[i])
    return res+1

```

Twitter | OA 2019 | Social Network

```

static Map<Integer, Node> graph = new HashMap<>();
static class Node {
    int rank;
    Node parent;
    Node() {
        rank = 0;
        parent = this;
    }
}

private static Node findParent(Node node) {
    if (node == node.parent) return node;
    node.parent = findParent(node.parent);
    return node.parent;
}

private static void union(Node node1, Node node2) {
    Node p1 = findParent(node1);
    Node p2 = findParent(node2);

    if (p1 == p2) return;
    if (p1.rank >= p2.rank) {
        p2.parent = p1;
        p1.rank += 1;
    } else {
        p1.parent = p2;
        p2.rank += 1;
    }
}

public static int countGroups(List<String> related) {
    for(int i = 0; i < related.size(); i++) {
        graph.put(i, new Node());
    }

    for(int i = 0; i < related.size(); i++) {
        for(int j = i+1; j < related.size(); j++) {
            if (related.get(i).charAt(j) == '1') {
                union(graph.get(i), graph.get(j));
            }
        }
    }

    Set<Node> set = new HashSet<>();
    for(int i = 0; i < related.size(); i++) {
        set.add(findParent(graph.get(i)));
    }
    return set.size();
}

```

```
int tallestHashtag(int[] positions, int[] heights){
    int max = 0;
    for(int i = 1; i < positions.length; i++){
        if(Math.abs(positions[i-1] - positions[i]) > 1){
            max = Math.max(max, getMaxHeight(positions[i-1], positions[i], heights[i-1], heights[i]));
        }
    }
    return max;
}

int getMaxHeight(int t1, int t2, int h1, int h2){
    int shorter = Math.min(h1, h2);
    int taller = Math.max(h1, h2);
    int gap = Math.abs(t2 - t1) - 1;
    if(taller >= shorter + gap){
        return shorter + gap;
    } else {
        int top = shorter + gap;
        int down = taller + 1;
        return (top + down) / 2;
    }
}
```

Twitter | OA 2019 | Anagram Difference

```

public static List<int> getMinimumDifference(List<string> a, List<string> b)
{
    string s1 = "", s2 = "";
    List<int> result = new List<int>();
    int length = a.Count, i = 0;
    while (i < length)
    {
        s1 = a[i];
        s2 = b[i];
        result.Add(countManipulations(s1, s2));
        i++;
    }
    return result;
}

static int countManipulations(string s1,
                               string s2)
{
    if (s1.Length != s2.Length) return -1;
    int count = 0;

    // store the count of character
    int[] char_count = new int[26];

    // iterate though the first String
    // and update count
    for (int i = 0; i < s1.Length; i++)
        char_count[s1[i] - 'a']++;

    // iterate through the second string
    // update char_count.
    // if character is not found in
    // char_count then increase count
    for (int i = 0; i < s2.Length; i++)
        char_count[s2[i] - 'a']--;

    for (int i = 0; i < 26; ++i)
    {
        if (char_count[i] != 0)
        {
            count += Math.Abs(char_count[i]);
        }
    }
    return count / 2;
}

```

Twitter | OA 2019 | Balanced Sales Array


```

public static int minIndex(int arr[]) {
    int totalSum=0;
    for(int i=0;i<arr.length;i++){
        totalSum+=arr[i];
    }
    int leftSum =0;
    for(int i=0;i<arr.length;i++){
        if(totalSum-arr[i]-leftSum == leftSum){
            return i;
        }
        leftSum+=arr[i];
    }
    return -1;
}

```

1. Twitter new office design

貌似是道数学题... <https://leetcode.com/discuss/int ... r-New-Office-Design> 169

2. Efficient Job Processing

经典的 0/1 背包问题，用 DP 解: <https://leetcode.com/discuss/int ... -Processing-Service> 107

3. Game Event

<https://leetcode.com/discuss/int ... 2019-or-Game-Events> 76

4. Unique Twitter User Id Set

<https://leetcode.com/discuss/int ... Twitter-User-Id-Set> 64

5. Partitioning Array

主要是判断 1. $\text{len}(\text{numbers}) \% k \neq 0$; 2. 是否有元素的个数 $> \text{len}(\text{numbers}) // k$: <https://leetcode.com/discuss/int ... -Partitioning-array> 58

6. Autoscale Policy

<https://leetcode.com/discuss/int ... or-Autoscale-Policy> 919

7. Authentication Token

<https://leetcode.com/discuss/int ... thentication-Tokens> 117

8. K difference

LC 伍叁贰

9. Buying show tickets

(这是唯一一个我们找到原题的...不好意思...)

10. Weird Faculty

<https://leetcode.com/discuss/int ... 19-or-Weird-Faculty> 39

11. Final discounted price

LC 齐三久 变种

12. Reaching Points

LC 岐拔灵 原

13. twitter social network

LC 吴思琪 原

14. Activate fountain

算是 greedy 里比较经典的区间覆盖问题，可以转化成 interval 以后 sort: <https://leetcode.com/discuss/int ... r-Activate-Fountain> 167

15. Coloring the blocks

LC 而物流 原

16. Parking Dilemma

这个题 LC 讨论里的图非常不清楚，不过好像不难，用 sliding window 就好: <https://leetcode.com/discuss/int ... -or-Parking-Dilemma> 69

17. Get set on

LC 似时 变种 不要求全部可能，所以可以用 set

18. Sub Palindrome

LC 刘思琪 变种 需要找到 unique substring，额外加一个 set() 检查一下就好

19. Restocking the Warehouse

这个不难...遍历一下就好

20. Balanced Sales Array

这个也不难，也是遍历就好

21. university career fair

<https://leetcode.com/discuss/int ... versity-Career-Fair> 74

22. Anagram Difference

Hacker 上的题: <https://www.hackerrank.com/challenges/anagram/problem> 73