

Classwork

Allocate Books [Coding Ninjas]

Question Link:-

https://www.codingninjas.com/studio/problems/allocate-books_1090540?leftPanelTab=0

The screenshot shows the Coding Ninjas Studio interface. On the left, the 'Current Submission' panel indicates the solution is 'Accepted' with 11/11 test cases passed. It shows an experience of 80/80, a 0% penalty, and a runtime of 2000 ms. A runtime graph is also visible. On the right, the code editor shows the Java solution for the 'Allocate Books' problem. The code defines a class 'Solution' with two methods: 'countStudents' and 'findPages'.

```
1 import java.util.*;
2 import java.util.ArrayList;
3 public class Solution {
4     public static int countStudents(ArrayList<Integer> arr, int pages){
5         int students=1;
6         long pagesStudent=0;
7         for(int i=0; i<arr.size(); i++){
8             if(pagesStudent + arr.get(i) <= pages){
9                 pagesStudent+=arr.get(i);
10            }
11            else{
12                students+=1;
13                pagesStudent=arr.get(i);
14            }
15        }
16        return students;
17    }
18    public static int findPages(ArrayList<Integer> arr, int n, int m) {
19        if(m>n) return -1;
20        int low = Collections.max(arr);
21        int high= arr.stream().mapToInt(Integer::intValue).sum();
```

Java Code

```
import java.util.*;
import java.util.ArrayList;
public class Solution {
    public static int countStudents(ArrayList<Integer> arr, int pages){
        int students=1;
        long pagesStudent=0;
        for(int i=0; i<arr.size(); i++){
            if(pagesStudent + arr.get(i) <= pages){
                pagesStudent+=arr.get(i);
            }
            else{
                students+=1;
                pagesStudent=arr.get(i);
            }
        }
        return students;
    }
    public static int findPages(ArrayList<Integer> arr, int n, int m) {
        if(m>n) return -1;
        int low = Collections.max(arr);
        int high= arr.stream().mapToInt(Integer::intValue).sum();
```

```

        while(low<=high){
            int mid=(low+high)/2;
            int students=countStudents(arr, mid);
            if(students > m) low=mid+1;
            else high = mid-1;
        }
        return low;
    }
}

```

Aggressive Cows [Coding Ninjas]

Question Link:-

https://www.codingninjas.com/studio/problems/aggressive-cows_1082559?source=youtube&campaign=love_babbar_codestudio2&utm_source=youtube&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2&leftPanelTab=0

The screenshot shows the Coding Ninjas IDE interface. On the left, the 'Current Submission' panel indicates the solution is 'Accepted' with 10/10 test cases passed. It shows a runtime of 749 ms and a memory usage of 38.72%. The right panel displays the Java code for the 'aggressiveCows' method, which uses a binary search approach to find the maximum minimum distance between stalls.

Java Code

```

import java.util.*;

public class Solution {
    public static int aggressiveCows(int [][]stalls, int k) {
        // Write your code here.
        Arrays.sort(stalls);
        int lo = 1;

```

```

        int hi = stalls[stalls.length-1]+1;
        int mid = 0;
        int pos =0;
        while(lo<=hi){
            mid = lo + (hi-lo)/2;
            boolean check = isPossible(stalls, k, mid);
            if(check){
                lo = mid+1;
                pos = mid;
            }else hi = mid-1;
        }
        return pos;
    }

    public static boolean isPossible(int[] stalls, int k, int dist){
        k--;
        int curr = 0;
        int i=1;
        while(k>0&&i<stalls.length){
            if(stalls[i]-stalls[curr] >=dist){
                k--;
                curr = i;
            }
            i++;
        }
        if(k>0) return false;
        return true;
    }
}

```

Homework

Painter's Partition Problem [Coding Ninjas]

Question Link:-

https://www.codingninjas.com/studio/problems/painter-s-partition-problem_1089557?source=youtu&utm_source=youtu&utm_medium=affiliate&utm_campaign=love_babbar_codestudio2&leftPanelTab=0

Current Submission

Accepted

Test Cases: 50/50

Language: Java

EXP: 80/80

Penalty: 0%

Runtime graph

You performed better than 10.42%

Runtime: 4046 ms

Best

Runtime (ms)

140 1,125 2,110 3,095 4,080

Java (SE 1.8)

Autocomplete

```

18 }
19 0 references
20 public static int findLargestMinDistance(ArrayList<Integer> boards, int k)
21 {
22     long low = 0;
23     long high = 0;
24     for( int i = 0; i < boards.size(); i++ ){
25         low = Math.max( low, boards.get(i) );
26         high += boards.get(i);
27     }
28
29     while( low <= high ){
30         long mid = ( low + high ) / 2;
31         long painters = countPainters(boards, mid);
32         if( painters > k ){
33             low = mid + 1;
34         }
35         else{
36             high = mid - 1;
37         }
38     }
39     return (int)low;
40 }

```

Run Submit View hints

Java Code

```

import java.util.ArrayList;
import java.lang.Math;
public class Solution
{
    static long countPainters( ArrayList<Integer> arr, long max ){
        long currBoards = 0;
        long painterCount = 1;
        for( int i = 0 ; i < arr.size() ; i++ ){
            if( currBoards + arr.get(i) <= max ){
                currBoards += arr.get(i);
            }
            else{
                painterCount++;
                currBoards = arr.get(i);
            }
        }
        return painterCount;
    }
    public static int findLargestMinDistance(ArrayList<Integer> boards,
int k)
    {
        long low = 0;
        long high = 0;
        for( int i = 0; i < boards.size(); i++ ){
            low = Math.max( low, boards.get(i) );

```

```
        high += boards.get(i);
    }

    while( low <= high ){
        long mid = ( low + high ) / 2;
        long painters = countPainters(boards, mid);
        if( painters > k ){
            low = mid + 1;
        }
        else{
            high = mid - 1;
        }
    }
    return (int)low;
}
}
```