Hard

31 December 2022 23:15

Square root of a number

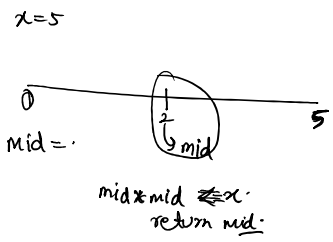
Medium Accuracy: **54.03%** Submissions: **129K+** Points: **4**

Stand out from the crowd. Prepare with Complete Interview Preparation

Given an integer **x**, find the square root of x. If **x** is not a perfect square, then return floor(√x).

Example 1:

Input:
x = 5
Output: 2
Explanation: Since, 5 is not a perfect square, floor of square_root of 5 is 2.



Example 2:

Input:
x = 4
Output: 2
Explanation: Since, 4 is a perfect square, so its square root is 2.

Your Task:
You don't need to read input or print anything. The task is to complete the function **floorSqrt()** which takes x as the input parameter and return its square root.
Note: Try Solving the question without using the sqrt function. The value of x>=0.

```cpp
// x: element to find square root
class Solution{
  public:
    long long int floorSqrt(long long int x)
    {
        // Your code goes here
        long long  start =0;
        long long  end = x;
        long long  res = 0;
        while(start <= end){
            long long  mid = (start + end)/2;
            long long square = mid * mid;
            if(square == x){
                res = mid;
            }
            if(square > x){
                end = mid -1;
            }
            else{
                res = mid;
                start = mid +1;
            }
        }
        return res;
    }
};
// } Driver Code Ends
```

Screen clipping taken: 31-12-2022 23:16

Screen clipping taken: 31-12-2022 23:21

que 2:-

Find Nth root of M

Easy Accuracy: **25.06%** Submissions: **50K+** Points: **2**

Stand out from the crowd. Prepare with Complete Interview Preparation

You are given 2 numbers **(n , m)**; the task is to find $\sqrt[n]{m}$ (n^{th} root of m).

Example 1:

Input: n = 2, m = 9
Output: 3
Explanation: $3^2 = 9$

Example 2:

Input: n = 3, m = 9
Output: -1
Explanation: 3rd root of 9 is not integer.

```cpp
// } Driver Code Ends
class Solution{
    public:
    int NthRoot(int n, int m)
    {
        // Code here.
        int low = 1;
        int high = m;
        while(low <= high){
            int mid = (low + high)/2;
            double val = pow(mid,n);
            if(val == m){
                return mid;
            }
            else if(val < m){
                low = mid+1;
            }
            else{
                high = mid-1;
            }
        }
        return -1;
    }
};
// } Driver Code Ends
```

Screen clipping taken: 31-12-2022 23:22

que 3:- Koko Eating bananas

Description Discussion (9) Solutions (1.4K) Submissions

875. Koko Eating Bananas

Medium 5.7K 266

Companies

Koko loves to eat bananas. There are `n` piles of bananas, the `i`th pile has `piles[i]` bananas. The guards have gone and will come back in `h` hours.

Koko can decide her bananas-per-hour eating speed of `k`. Each hour, she chooses some pile of bananas and eats `k` bananas from that pile. If the pile has less than `k` bananas, she eats all of them instead and will not eat any more bananas during this hour.

Koko likes to eat slowly but still wants to finish eating all the bananas before the guards return.

Return *the minimum integer `k` such that she can eat all the bananas within `h` hours.*

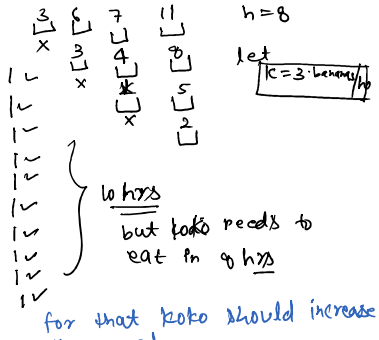
Example 1:

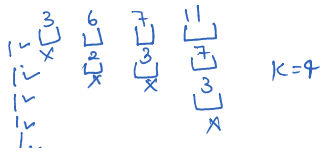
Input: piles = [3,6,7,11], h = 8
Output: 4

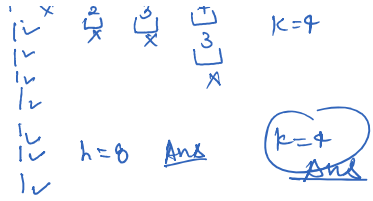
Example 2:

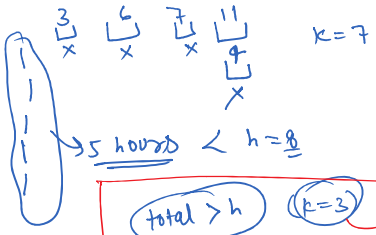
Input: piles = [30,11,23,4,20], h = 5
Output: 30

Screen clipping taken: 01-01-2023 16:19







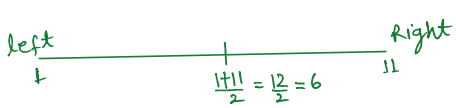


if k ↑ then time less
but we need to return minimum k

total > h → increase k value
total < h → decrease k value

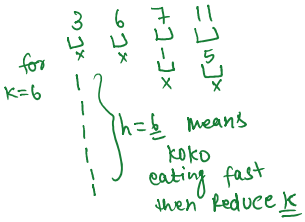
this concept is Binary Search on Answers → total time will koko taking to eat all bananas.

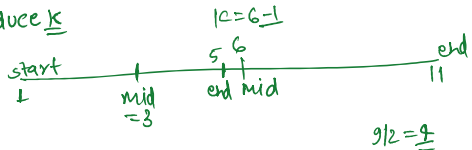
3 6 7 11 h=8

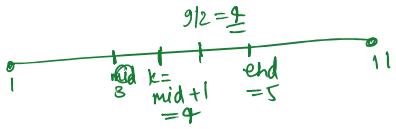


left = 1 hr
right = 11

k=6







for k=3
total hour = 10 very greater then increase the value of (k)

for k = (4) it takes 8 hours to it all the bananas

```java
class Solution {
    private int getMaxPile(int[] piles) {
        int maxPile = Integer.MIN_VALUE;
        for (int pile : piles) {
            maxPile = Math.max(pile, maxPile);
        }
        return maxPile;
    }
    public boolean caneatIntime(int[] piles, int k, int h){
        int hours =0;
        for(int pile : piles){
            int div = pile /  k;
            hours += div;
            if(pile % k != 0){
                hours++;
            }
        }
        return hours <= h;
    }
    public int minEatingSpeed(int[] piles, int h) {
        int left =1;
        int right = getMaxPile(piles);
        while(left <= right){
            int mid = left + (right - left)/2;
            if(caneatIntime(piles, mid, h)){
                right = mid -1;
            }
```

Or

```java
class Solution {

    public boolean caneatIntime(int[] piles, int k, int h){
        int hours =0;
        for(int pile : piles){
            int div = pile /  k;
            hours += div;
            if(pile % k != 0){
                hours++;
            }
        }
        return hours <= h;
    }
    public int minEatingSpeed(int[] piles, int h) {
        int left =1;
        int right = 1000000000;
        while(left <= right){
            int mid = left + (right - left)/2;
            if(caneatIntime(piles, mid, h)){
                right = mid -1;
            }
            else{
                left = mid +1;
            }
```

```java
public int minEatingSpeed(int[] piles, int h) {
    int left =1;
    int right = getMaxPile(piles);
    while(left <= right){
        int mid = left + (right - left)/2;
        if(caneatIntime(piles, mid, h)){
            right = mid -1;
        }
        else{
            left = mid +1;
        }
    }
    return left;
}
}
```

Screen clipping taken: 01-01-2023 19:43

Screen clipping taken: 01-01-2023 16:46

```java
    while(left <= right){
        int mid = left + (right - left)/2;
        if(caneatIntime(piles, mid, h)){
            right = mid -1;
        }
        else{
            left = mid +1;
        }
    }
    return left;
}
}
```

Screen clipping taken: 01-01-2023 16:47

Ques:- Minimum Number of Days to make m Bouquets :-

bloomDay = [1,10,2,9,3,8,4,7,5,6] , m=4, k=2

Description | Discussion (4) | Solutions (623) | Submissions

1482. Minimum Number of Days to Make m Bouquets

Hint

Medium 2.3K 63

Companies

You are given an integer array `bloomDay`, an integer `m` and an integer `k`.

You want to make `m` bouquets. To make a bouquet, you need to use `k` **adjacent flowers** from the garden.

The garden consists of `n` flowers, the `i`th flower will bloom in the `bloomDay[i]` and then can be used in **exactly one** bouquet.

Return *the minimum number of days you need to wait to be able to make `m` bouquets from the garden*. If it is impossible to make m bouquets return `-1`.

Example 1:

```
Input: bloomDay = [1,10,3,10,2], m = 3, k = 1
Output: 3
Explanation: Let us see what happened in the first three days. x means flower bloomed and _ means flower did not bloom in the garden.
We need 3 bouquets each should contain 1 flower.
After day 1: [x, _, _, _, _]   // we can only make one bouquet.
After day 2: [x, _, _, _, x]   // we can only make two bouquets.
After day 3: [x, _, x, _, x]   // we can make 3 bouquets. The answer is 3.
```

bloomday: 1 | 10 | 2 | 9 | 3 | 8 | 4 | 7 | 5

m=4, k=2.

m= Bouquets

k=flower.

conditions

① Adjacent use

② Use one flower exactly ones

Example 2:

```
Input: bloomDay = [1,10,3,10,2], m = 3, k = 2
Output: -1
Explanation: We need 3 bouquets each has 2 flowers, that means we need 6 flowers. We only have 5 flowers so it is impossible to get the needed bouquets and we return -1.
```

Example 3:

```
Input: bloomDay = [7,7,7,7,12,7,7], m = 2, k = 3
Output: 12
Explanation: We need 2 bouquets each should have 3 flowers.
Here is the garden after the 7 and 12 days:
After day 7: [x, x, x, x, _, x, x]
We can make one bouquet of the first three flowers that bloomed. We cannot make another bouquet from the last three flowers that bloomed because they are not adjacent.
After day 12: [x, x, x, x, x, x, x]
It is obvious that we can make two bouquets in different ways.
```

Screen clipping taken: 01-01-2023 19:44

exp:- ①

Input bloomDay [6,6,6,7,7,7,12] m=2 (k=3) → We have to make 2 bouquets

each bouquets have three Adjacent flower

wait six days for ③

1 bouquets

wait seven days for 7

so the minimum no. of days = ⑦

exp2:- bloomDay = [6,6,6,7,12,7,7] m=2, k=3

1st bouquets

so the minimum no. of days we have to wait is 12

[6,6,6,7,12,7,7]

Output is 6 | 12 | T/f

canMake(mid)
{ cnt = 1 → 3 → 3 adjacent flower (k)
now bouquets++
b++ = 1
again made cnt=0 b/c we found 3 adjacent

Time

(8)

Time Complexity O(nlog m)

6 8

8 10 12

10 mid

b++ = 1

again made cnt=0 b/c we found 3 adjacent

cnt = 1

for bloom[i] = 12 < 8

again

q

```java
class Solution
{
    public int minDays(int[] bloomDay, int m, int k)
    {
        //  m*k= Number of flowers required that is should be <= n (bloomDay.length)
        //  k: is also the length of the subarray (flowers for bouquet)

        if (m*k> bloomDay.length)
            return -1;
        int ans=-1;

        /*  Main Idea: m subarrays (bouquets) of size k, and we need to minimize the maximum value in these sub-arrays
         -> if above condition (m*k> len) is false, we are guaranteed an answer.
            Also, we know the answer lies between the maximum and minimum value in our bloom-array, and since the answer is guaranteed we can check the feasibility (which means searching will be involved) and depending on that change our search space.
         -> So we might be able to use Binary Search instead of checking for each element of the array.

         Let's think*/
         int low=0;
         int high= (int)Math.pow(10,9); // Also can find the max and min values from the array

        while (low<=high)
        {
            int mid= low+ (high-low)/2; // Possible result

            // Check if it is possible?
            if (isPossible(bloomDay, mid, m, k)){
                ans= mid;
                high= mid-1; // Lets check for a lesser possible answer
            }
            else
                low= mid+1; // Looks like we can not have m bouquets for this mid

        }
        return ans;

    }
    // Check for a possible answer (mid): Going through the array and checking if for an assumed answer, can we make more than or equal to required bouquets
    // [Essentially finding the count of subarrays of size k with all elements less than or equal to mid]
```

Screen clipping taken: 01-01-2023 20:47

c

```java
    // [Essentially finding the count of subarrays of size k with all elements less than or equal to mid]
    public boolean isPossible(int arr[], int mid, int bouquets, int k)
    {
        int flowers=0;
        int boqs=0;

        for (int i=0; i<arr.length; i++)
        {
            if (arr[i]>mid) // Implies flower hasn't bloomed so can not be a part of subarray, so count set to 0
                flowers=0; // Since adjacent

            else
            // We need to check if the number of flowers is k(indicates size of curr subarray)
            {
                flowers++; // Count of elements/ flowers in subarray/ bouquets (adjacent)
                if (flowers==k)
                    {
                    boqs++; // Count of bouquets
                    flowers=0;
                }
                else
                    continue;
            }

        }
        if (boqs>= bouquets) // We can make m bouquets
            return true;
        else
            return false;
    }
}
```

Screen clipping taken: 01-01-2023 20:47

Que 5:-

1283. Find the Smallest Divisor Given a Threshold

Hint

Medium 1.8K 163

Companies

Given an array of integers `nums` and an integer `threshold`, we will choose a positive integer `divisor`, divide all the array by it, and sum the division's result. Find the **smallest** `divisor` such that the result mentioned above is less than or equal to `threshold`.

Each result of the division is rounded to the nearest integer greater than or equal to that element. (For example: `7/3 = 3` and `10/2 = 5`).

The test cases are generated so that there will be an answer.

Example 1:

```
Input: nums = [1,2,5,9], threshold = 6
Output: 5
Explanation: We can get a sum to 17 (1+2+5+9) if the divisor is 1.
If the divisor is 4 we can get a sum of 7 (1+1+2+3) and if the divisor is 5 the sum will be 5 (1+1+1+2).
```

Example 2:

```
Input: nums = [44,22,33,11,1], threshold = 5
Output: 44
```

Screen clipping taken: 01-01-2023 20:56

nums = 1,2,5,9 , threshold = 6

l	r	mid	sum	Condition
1	9	5	1+1+1+2 = 5	T
L	5	3	1+1+2+3 = 7	F
4	5	4	1+1+2+3 = 7	F
4+1 = 5	—	—		

sum = $\frac{num + div - 1}{div} \Rightarrow \frac{1+5-1}{5} \Rightarrow 1$

$\Rightarrow \frac{2+5-1}{5} \Rightarrow \frac{6}{5} = 1$

$\frac{5+5-1}{5} \Rightarrow 2$

$\frac{9+5-1}{5} \Rightarrow$

e

```java
class Solution {
    public int smallestDivisor(int[] nums, int threshold) {
        int left = 1;
        int right = (int) 1e6;

        while (left < right) {
            int middle = left + (right - left) / 2;
            if (sumIsLessOrEqual(nums, middle, threshold))
                right = middle;
            else
                left = middle + 1;
```

```java
class Solution {
    public int smallestDivisor(int[] nums, int threshold) {
        int left = 1;
        int right = (int) 1e6;

        while (left < right) {
            int middle = left + (right - left) / 2;
            if (sumIsLessOrEqual(nums, middle, threshold))
                right = middle;
            else
                left = middle + 1;
        }

        return left;
    }

    private boolean sumIsLessOrEqual(int[] nums, int divisor, int threshold) {
        int sum = 0;
        for (int num : nums) {
            sum += Math.ceil((float) num / divisor); //(num + divisor - 1) / divisor;
        }

        return sum <= threshold;
    }
}
```

$\frac{5+5-1}{5} \Rightarrow 9$

$\frac{9+5-1}{5} \Rightarrow$

Screen clipping taken: 01-01-2023 21:36

que 6°

1011. Capacity To Ship Packages Within D Days

Hint

Medium 5.7K 118

Companies

A conveyor belt has packages that must be shipped from one port to another within `days` days.

The i^{th} package on the conveyor belt has a weight of `weights[i]`. Each day, we load the ship with packages on the conveyor belt (in the order given by `weights`). We may not load more weight than the maximum weight capacity of the ship.

Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within `days` days.

Example 1:

Input: weights = [1,2,3,4,5,6,7,8,9,10], days = 5
Output: 15
Explanation: A ship capacity of 15 is the minimum to ship all the packages in 5 days like this:
1st day: 1, 2, 3, 4, 5
2nd day: 6, 7
3rd day: 8
4th day: 9
5th day: 10

Note that the cargo must be shipped in the order given, so using a ship of capacity 14 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) is not allowed.

Screen clipping taken: 02-01-2023 08:02

Example 2:

Input: weights = [3,2,2,4,1,4], days = 3
Output: 6
Explanation: A ship capacity of 6 is the minimum to ship all the packages in 3 days like this:
1st day: 3, 2
2nd day: 2, 4
3rd day: 1, 4

Example 3:

Input: weights = [1,2,3,1,1], days = 4
Output: 3
Explanation:
1st day: 1
2nd day: 2
3rd day: 3
4th day: 1, 1

Screen clipping taken: 02-01-2023 08:02

Days = 1

0	1	2	3	4
3	5	4	6	2

Capacity = 20

Lets say No of Days = 5

this one is Brute force Approach D=5

3	5	4	6	2

minimum = max = 6

highest = 20 days

boat break. 6kg boat carry all the Weight

for Days = 2

3	5 | 4 | 6 | 2

day1 day2 = 17 kg.

→ not will be the Optimal option.

for large Days value partition will not Worked

Lets calculate using low and high°-

low = 6
high = 20
days = 3

3	5	4	6	2

day 1 | day 2 | day 3 | day 4 | day 5

assume Ans = 6

will take 5 days

then pt will not the ans

3	5	4	6	2

Ans=7

1 2 3 4 5

again (5)

ans = 8

3	5	4	6	2

1 | 2day | 3day

total pt will take 3days so this one is answer

instead of linear search use the concept of Binary search O(log(high - low))

Dry Run

3	5	4	6	2

day 1 | day 2

low = 6
high = 20
mid = 13

Day -> 3

ans = 2 days

better answer than 13 → mini

low = 6
high = mid - 1 = 12
mid = 18 = 9

better answer than (12) → mini

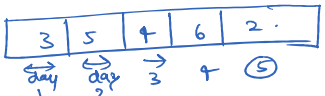
low = 6
high = mid − 1 = 12
mid = 18/2 = (9)

mid = 9

3days

9 is also will be a ans

low = 6
high = 8
mid = (7)



5 days

So this is not the ans.

low = mid + 1 = 8
high = 8

So answer for 8 kg = 3days

and the minimum weight that you will have is 8 kg

low = 6 (max all (day = 4)
high = 20 sum (day == 1)

e

```java
class Solution {
    boolean ispossible(int mid, int[] weight, int D, int n){
        int sum =0;
       int days =1;
        for(int i=0;i<n;i++){
            if(sum + weight[i] <= mid) sum += weight[i];
            else{
                sum = weight[i];  //reset
                days++;           //increase no of days
            }
        }
        if(days <= D) return true;
        else return false;
    }
    public int shipWithinDays(int[] weights, int days) {
        int sum =0;
        int maxi = 0;
        int ans =0;
        int n = weights.length;
        for(int i=0;i<n;i++){
                if(maxi < weights[i]){
                    maxi = weights[i];

                }
                sum += weights[i];
        }
        int low = maxi;
        int high = sum;
        while(low <= high){
            int mid = (low + high)/2;
            if(ispossible(mid , weights, days, n)){
                ans = mid;
                high = mid -1;
            }
            else{
                low  = mid +1;
            }
        }
        return ans;
    }
}
```

Screen clipping taken: 02-01-2023 08:32

que 7:-

4. Median of Two Sorted Arrays

Hard 21.3K 2.4K

Companies

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The overall run time complexity should be `O(log (m+n))`.

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

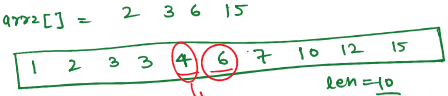
Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```

Screen clipping taken: 02-01-2023 08:33

arr1[] = 1 3 4 7 10 12

arr2[] = 2 3 6 15



len = 10

even then (2) mid

n/2

(4+6)/2 = (5) Ans

exp2

[1 3]

[4 5 6]

→ 1 3 4 5 6

len = 5

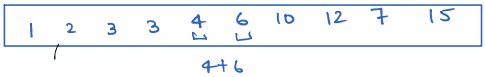
Ans = 5

Brute force Approach:-

arr1[] = 1 3 4 7 10 12

arr2[] = 2 3 6 15

Merge both the array



(4+6)/2

e

```java
float median(int nums1[],int nums2[],int m,int n) {
    int finalArray[n+m];
    int i=0,j=0,k=0;
    while(i<m && j<n) {
        if(nums1[i]<nums2[j]) {
            finalArray[k++] = nums1[i++];
        }
        else {
            finalArray[k++] = nums2[j++];
        }
    }
    if(i<m) {
        while(i<m)
            finalArray[k++] = nums1[i++];
    }
    if(j<n) {
        while(j<n)
            finalArray[k++] = nums2[j++];
    }
    n = n+m;
```

both the array

1 2 3 3 4 6 10 12 7 15

$\frac{4+6}{2}$

TC:- O(N1+N2)

SC:- O(N1+N2)

```
        while(i<n)
            finalArray[k++] = nums1[i++];
    }
    if(j<n) {
        while(j<n)
            finalArray[k++] = nums2[j++];
    }
    n = n+m;
    if(n%2==1)
        return finalArray[((n+1)/2)-1];
    else return ((float)finalArray[(n/2)-1]+(float)finalArray[(n/2)])/2;
}
```

Screen clipping taken: 02-01-2023 09:20

Optimize Approach:-

ds:- 1 2 3 3 4 6 7 10 12 15

1 2 3 4 ⑤ ⑥

cnt

get only the desired element

Better version of naive SC:- O(1)

We can optimize in space complexity.

Approach :

Similar to the naive approach, instead of storing the final merged sorted array, we can keep a counter to keep track of the required position where the median will exist. First, using the median formula, get a position where the median will exist. Then, follow the above approach and instead of storing elements in another array, we will increase the counter value. When the counter value is equal to the median positions, return element.

Time Complexity : O(m+n)

Reason – We are still traversing both the arrays linearly.

Space Complexity: O(1)

Reason – We do not use any extra array.

Screen clipping taken: 02-01-2023 09:21

Better Approach:-

arr1[] → 1 3 4 7 10 12

arr2[] → 2 3 6 15

⑤ ⑤

1 2 3 3 4 6 7 10 12 15

7<=3 ✗

1 3 4 7 | 10 12

2 | 3 6 15

pick 3 elem from arr1

arr1[] 1 3 4 7 10 12

arr2[] 2 3 6 15

2 ele. from arr2

4<=6 ✓

3<=7 ✓

1 3 4 | 7 10 12

2 | 6 15

max of left half

max(l1, l2)

min of right half

min(r1, r2)

1 3 | 4 7 10 12

2 3 | 15

3<=15 ✓

6<=4 ✗ this is also not the left half

$\frac{\max(l_1, l_2) + \min(r_1, r_2)}{2}$ =

$l_1 <= r_2$

$l_2 <= r_1$ has to be Valid

how to apply binary search

arr1[] = 1 3 | 4 7 10 12

arr2[] = 2 3 6 15

move left

you need to move left

7<=3 ✗

3 4 7 | 10 12

2 | 3 6 15

4<=3

move Right

shifted the partition left side

1 3 4 | 7 10 12

2 3 | 6 15

6<=4 ✗

1 3 | 4 7 10 12

2 3 6 | 15

arr1 → 7 10 14 15

partition

arr2 → 1 2 3 4 9 11

0 1 2 3 4

0 1 2 3 4 5 6

5 5

minele → low 0 high 4 max

0 1

cut 1 = $\frac{0+4}{2}$ = 2

cut 2 = (5-2) = 3

l1 12 r1 14

l2 3 r2 4

12<=4 ✗

move left

high = mid-1

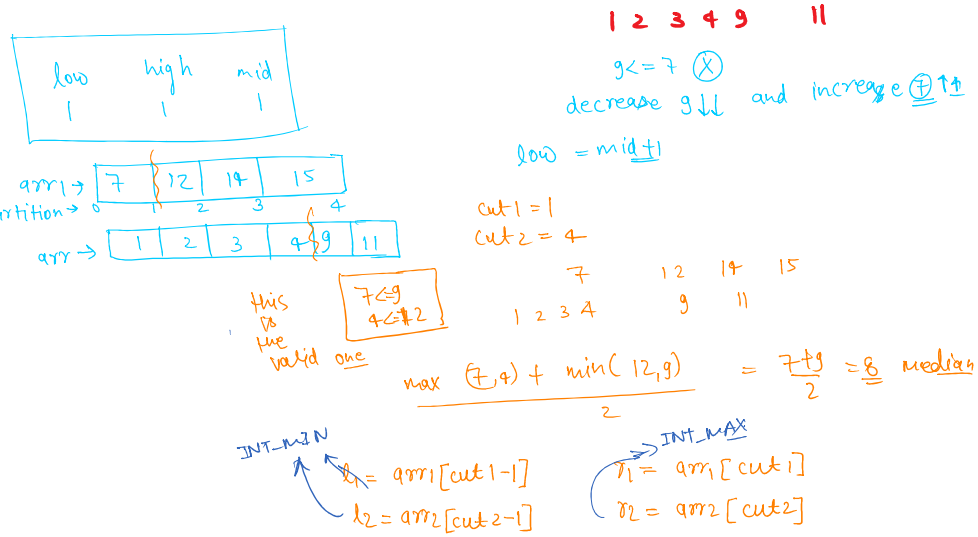
not a valid partition

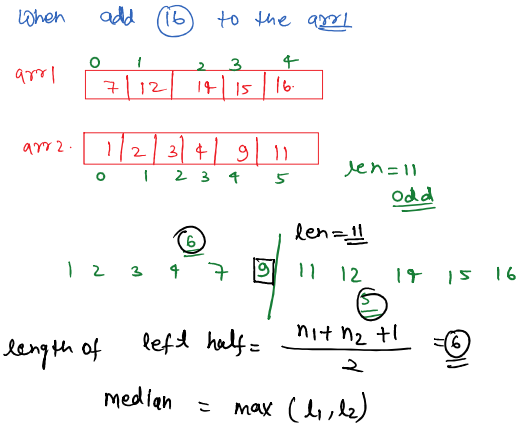
cut 1 = $\frac{0+1}{2}$ = 0

cut 2 = 5-0 = 5

l1 = INT_MIN

7 12 14 15






```java
class Solution {
    public static double findMedianSortedArrays(int[] arr1, int[] arr2) {
        int m = arr1.length;
        int n = arr2.length;
        if(m>n)
        return findMedianSortedArrays(arr2,arr1);//ensuring that binary search happens on minimum size array

    int low=0,high=m,medianPos=((m+n)+1)/2;
    while(low<=high) {
        int cut1 = (low+high)>>1;
        int cut2 = medianPos - cut1;

        int l1 = (cut1 == 0)? Integer.MIN_VALUE:arr1[cut1-1];
        int l2 = (cut2 == 0)? Integer.MIN_VALUE:arr2[cut2-1];
        int r1 = (cut1 == m)? Integer.MAX_VALUE:arr1[cut1];
        int r2 = (cut2 == n)? Integer.MAX_VALUE:arr2[cut2];

        if(l1<=r2 && l2<=r1) {
            if((m+n)%2 != 0)
                return Math.max(l1,l2);
            else
                return (Math.max(l1,l2)+Math.min(r1,r2))/2.0;
        }
        else if(l1>r2) high = cut1-1;
        else low = cut1+1;
    }
    return 0.0;
    }
}
```

Screen clipping taken: 02-01-2023 09:20

que:- Aggresive cow solution:-

Aggressive Cows

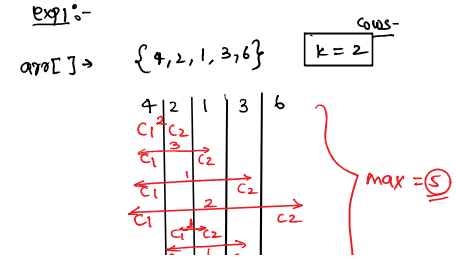
Medium Accuracy: 59.57% Submissions: 14K+ Points: 4

Stand out from the crowd. Prepare with Complete Interview Preparation

You are given an **array** consisting of **n integers** which denote the position of a **stall**.
You are also given an **integer k** which denotes the number of aggressive cows. You are given the task of **assigning stalls to k cows** such that the **minimum distance between any two of them is the maximum possible**.
The first line of input contains two space-separated integers **n** and **k**.
The second line contains **n** space-separated integers denoting the position of the stalls.

Example 1:

Input:
n=5
k=3
stalls = [1 2 4 8 9]
Output:



Input:
n=5
k=3
stalls = [1 2 4 8 9]
Output:
3

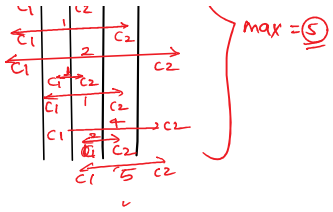
Screen clipping taken: 02-01-2023 09:22

3
Explanation:
The first cow can be placed at stalls[0],
the second cow can be placed at stalls[2] and
the third cow can be placed at stalls[3].
The minimum distance between cows, in this case, is 3,
which also is the largest among all possible ways.

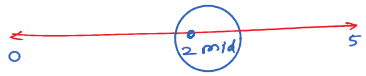
Example 2:

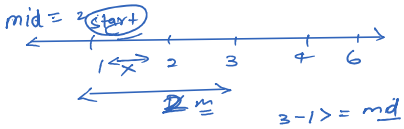
Input:
n=5
k=3
stalls = [10 1 2 7 5]
Output:
4
Explanation:
The first cow can be placed at stalls[0],
the second cow can be placed at stalls[1] and
the third cow can be placed at stalls[4].
The minimum distance between cows, in this case, is 4,
which also is the largest among all possible ways.

Screen clipping taken: 02-01-2023 09:22



Search space → min = 0
→ max = maxi − mini
= 6 − 1 = 5

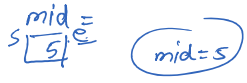




3−1 >= mid
2 >= 2 true

because want greater value − mid = 4

5 >= 4 possible soln :−
ans = 4
Right part



possible soln :−
ans = 5
s = mid + 1

s < e
↳ out of bound

i

```java
// User function Template for Java
class Solution {
    private static boolean ispossible(int[] stalls, int k, int n, int mid){
        int cowcnt = 1;
        int lastpos  = stalls[0];
        for(int i= 1;i<n;i++){
            if(stalls[i]  - lastpos >= mid){
                cowcnt++;
                 if(cowcnt == k){
                 return true;
                }
                lastpos = stalls[i];
            }
        }
        return false;
    }
    public static int solve(int n, int k, int[] stalls) {
        Arrays.sort(stalls);
        int low =0;
        // int maxi = -1;
        // for(int i=0;i<n;i++){
        //     maxi = Math.max(stalls[i] , maxi);
        // }
        int high = stalls[n-1];
        int ans = 1;
        while(low <= high){
            int mid= (low + high) /2;
            if(ispossible(stalls, k, n, mid)){
                ans = mid;
                low = mid +1;
            }
            else{
                high = mid -1;
            }
        }
        return ans;
    }
}
```

Screen clipping taken: 02-01-2023 10:11

que :− Book Allocation problem

Allocate minimum number of pages

Hard Accuracy: 35.51% Submissions: 108K+ Points: 8

Stand out from the crowd. Prepare with Complete Interview Preparation

You are given **N** number of books. Every **ith** book has **Ai** number of pages.
You have to allocate contiguous books to **M** number of students. There can be many ways or permutations to do so. In each permutation, one of the M students will be allocated the maximum number of pages. Out of all these permutations, the task is to find that particular permutation in which the maximum number of pages allocated to a student is the minimum of those in all the other permutations and print this minimum value.

Allocate minimum number of pages

Hard Accuracy: **35.51%** Submissions: **108K+** Points: **8**

Stand out from the crowd. Prepare with Complete Interview Preparation

You are given **N** number of books. Every **i**th book has **Ai** number of pages.
You have to allocate contiguous books to **M** number of students. There can be many ways or permutations to do so. In each permutation, one of the M students will be allocated the maximum number of pages. Out of all these permutations, the task is to find that particular permutation in which the maximum number of pages allocated to a student is the minimum of those in all the other permutations and print this minimum value.

Each book will be allocated to exactly one student. Each student has to be allocated at least one book.

Note: Return **-1** if a valid assignment is not possible, and allotment should be in contiguous order (see the explanation for better understanding).

Example 1:

Screen clipping taken: 02-01-2023 10:12

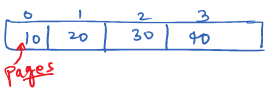
N = 4
A[] = {12,34,67,90}
M = 2
Output:113
Explanation: Allocation can be done in
following ways:{12} and {34, 67, 90}
Maximum Pages = 191{12, 34} and {67, 90}
Maximum Pages = 157{12, 34, 67} and {90}
Maximum Pages =113. Therefore, the minimum
of these cases is 113, which is selected
as the output.

Example 2:

Input:
N = 3
A[] = {15,17,20}
M = 2
Output:32
Explanation: Allocation is done as
{15,17} and {20}

Screen clipping taken: 02-01-2023 10:12

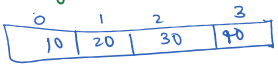
Expo:-



m=2 → student

Book allocation should be in a contiguous manner:-

You have to allocate the book to m students such that the maximum no. of pages assigned to a student is minimum-



n=4
m=2

1st way:- 10 | 20 30 40 → max = 90
2nd way :- 10 20 | 30 40 → max = 70
3rd way 10 20 30 | 40 → max = 60

minimum = 60

10 20 30 40 Jo entire ka minimum hoga

ans → search space

10 20 30 40
sum = 100
MAX = 100

min = 0

is possible solution (mid)
m=2 → for 2 student

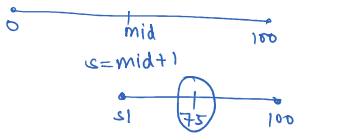
mid=50

for 1st student
10+20 +30X
II 2nd student 30 +40X

10+20 +30 ✗

II 2nd student 30 +40 ✗

III 3rd student → Do hi student hai & 3rd pe pahuch ja raha then so not a answer



10 20 30 | 40

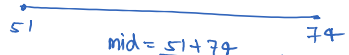
1st student 10+20+30 | 40

→ 60 < 75 ✗

2nd student 40 → Do student me sare Allot ho gaya then end ko picche le jayenge-

jab bhi possible soln aaye to Ans ko store kro &. left part me jayenge-

#



$mid = \frac{51+74}{2} = \frac{125}{2} = 62$

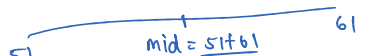
10 20 30 40

st 1 10+20+30 = 60 < 62 ✓ } possible-

st 2 40 < 62

ans = 62

high = mid-1 = 61



$\Rightarrow \frac{112}{2} = 56$

10 20 | 30 40

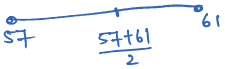
S1 → 10+20 30 < 56 ✓

S2 → 30 ✓

S3 ✗

Scroll Anyong taken: 02-01-2023 14:38

start = mid+1 = 57



S1 → 10+20 ✗ ✓

S2 → 30 ✓

S3 → ✗

S1 → 10+20+30 <= 60 ✓

S2 → 40

e = mid-1 ✗

return mid

e

```cpp
class Solution
{
    public:
    //Function to find minimum number of pages.
    bool isPossible(int arr[],int mid,int n ,int m){
        int studentcnt =1;
        int pagesum =0;
        for(int i=0;i<n;i++){
            if(pagesum + arr[i]<=mid){
                pagesum += arr[i];
            }
            else {
                studentcnt++;
                if(studentcnt > m || arr[i] > mid){
                    return false;
                }
                pagesum =arr[i];
            }
        }
        return true;
    }
    int findPages(int A[], int N, int M)
    {
        //code here
        int start =0;
        int sum =0;
        int ans = -1;
        for(int i=0;i<N;i++){
            sum += A[i];
        }
        int end = sum;
        while(start<=end){
            int mid = (start+end)/2;
            if(isPossible(A,mid,N,M)){
                ans = mid;
                end = mid-1;
            }
            else{
                start = mid +1;
            }
        }
        return ans;
    }
};
// } Driver Code Ends
```

que°

410. Split Array Largest Sum

Hard 7.4K 167

Companies

Given an integer array `nums` and an integer `k`, split `nums` into `k` non-empty subarrays such that the largest sum of any subarray is **minimized**.

Return *the minimized largest sum of the split.*

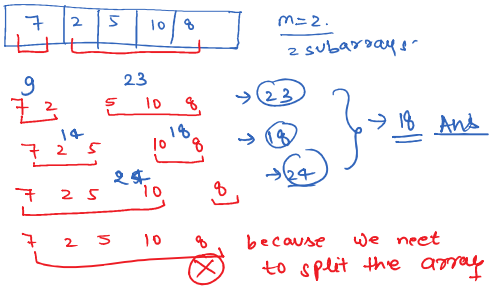
A **subarray** is a contiguous part of the array.

Example 1:

```
Input: nums = [7,2,5,10,8], k = 2
Output: 18
Explanation: There are four ways to split nums into two subarrays.
The best way is to split it into [7,2,5] and [10,8], where the largest
sum among the two subarrays is only 18.
```

Example 2:

```
Input: nums = [1,2,3,4,5], k = 2
Output: 9
Explanation: There are four ways to split nums into two subarrays.
The best way is to split it into [1,2,3] and [4,5], where the largest
sum among the two subarrays is only 9.
```



Screen clipping taken: 02-01-2023 15:07