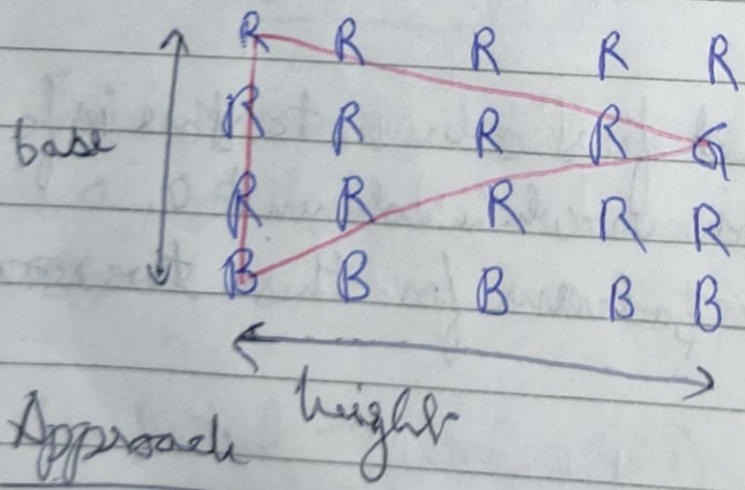
Maximum Area of Triangle - IB

Given a matrix of character of 'r', 'g', 'b'.
r → red
g → green
b → blue

find area of largest triangle whose base is parallel to y axis. that have all vertices of different colour.

Example :



output = 10

Approach

As area of triangle is $\frac{1}{2} \times$ base $\times$ height.
And base is given to be parallel to y axis.

1st Choice

Page No.
Date: 11/7/18

- For a fixed base if we move the third vertice we notice that the height of the triangle is described by its x-coordinate only.
(Area of Δ inscribed b/w same parallels is equal).

- Our goal is to find farthest 3rd coordinate to farthest point for each base.

Solution:-

- Keep leftmost and rightmost column values of r, g and b in variables.

  lr = inf, rr = -1, lb = inf, rb = -1, lg = inf, rg = -1

- Traverse matrix and update respective variables.
- int ans = 0
- Traverse matrix again and now calculate area for each possible base and Δ. (in column major order)

- return ans;

Find Permutation - IB

Given the int n and a string containing of D and I. Return permutation of no.s 1 to n (of length(str-1)) that satisfies the given scheme.

D → Decrement, I → Increment

Expected time and space complexity: O(n).

Example:- n=3
S = "ID"
output: [1,3,2].

Approach

Keep a set of all numbers and keep substituting the smallest from remaining for an increment and the largest remaining for a decrement. As it automatically makes the next element to be larger or smaller respectively.

Ex: n=5 S = "DIDD"

output: [5,1,4,3,2]

Maximum Consecutive Gap Ref LC

Given an unsorted array, find the maximum difference between successive elements in its sorted form.

Solution 1.

Use bucket sort to sort the array that will take O(n) time and O(n) space.
Then compute the desired.

Solution 2

Use Bucketing.

The idea is that bw and will be maximum if (mx and mn) are adjacent. And our answer will be minimum if all the 'gap's are equal.
So array will be

$mn + gap, mn + 2^{*} gap, mn + 3^{*} gap \ldots\ldots mn + (n-1)^{*} gap$

This gap can be calculated as

$max = min + (n-1) * gap$

$\therefore gap = \frac{max - min}{(n-1)}$

So if we create $(n-1)$ buckets like

$[min, min+gap), [min+gap, min+2*gap) \ldots$

There will only be $n-1$ such buckets
(we can calculate for max separately)

Algo:

Traverse the array and store values in these buckets

The difference between elements inside same bucket is redundant as there should be at least difference of 'gap' to contribute for our ans.

Traverse these buckets and compare diff between minimum element of current bucket and maximum element of previous bucket.

Prob 6] Triplets with sum between given range

Given an array of real numbers (in string format) greater than 0.
Find if there exists a triplet (a, b, c) such that
1 < a+b+c < 2

O(n) solution expected.
So we can convert string to double for calculations.

Approach (Quite non-intuitive)

You can convert string to double using stod(arr[i])

Now we will use bucketing.

1st Choice

Page No.

Find missing and Duplicate Number

Given an array of integers 1 to n. (Distinct)
Each integer appear exactly once except A which is repeated once and B which is missing.

Find A and B.

Time O(n) space O(1)

Approach 1.

We have two variables x (missing) and y (repeating). As we know numbers are fixed 1 to n and are distinct.

We can form two equations in terms of sum of n No. and sum of squares of n No.

Then solve linear eq^n in two variable.

Approach 2

Use the array itself as a hash by increasing the corresponding index by n and taking division and modulo afterwards.

1st Choice

Page No.
Date

Approach-3 (Aditya Verma)

☆ Use swap sort:

Exploits the fact that our range of numbers is fixed. So we can sort by swaping the number with the place where it should be in the sorted array (As no. are fixed to be between 1 and n).

N/3 repeated number in a array - O(1) space.

Given a read only array of n integers. Find and element that appears more than n/3 times in the array. If not return -1.

1st Choice

Page No.

Approach:- Moore's Voting Algorithm

↓

Standard to find majority element

Majority element :- An element that occurs more than n/2 times.

Algo:-

First step gives the element that may be the majority element.

In second step we check if it is or not.

Code:- maj_index ← 0, count ← 1.

```
loop(i to n){
  if(a[maj_index] == a[i])
    count++;
  else
    count--;
  if(count == 0){
    maj_index = i
    count = 1
  }
}
```

Afterwards just check if it is or not.

Now for n/3 repeated number

Code:

```
first ← INT_MAX, second ← INT_MAX
count1 ← 0 , count2 ← 0.
loop( i, [illegible] = 0, & i < n; i++) {
    if (first == arr[i])
        count1++;
    else if (second == arr[i])
        count2++;
    else if ( count1 == 0){
        count1 = 1
        first = arr[i]
    }else if (count2 == 0){
        count2 = 1
        second = arr[i]
    }else{
        count1--;
        count2--;
    }
end-loop. // Afterwards check for majority
```

First Missing Integer - IB

Given an unsorted integer array, find the first missing +ve integer.

Ex:- [3, 4, -1, 1] output: 2.

O(n) time constant space.

Sol:- Use swap sort.

Find K closest element

Given an array and two integers k and x. find k closest elements to x.

Approach 1 :- Use vector< pair<int, int> > — O(nlogn)

(abs(arr[i] - x), arr[i])

Sort and print elements from first k pairs.

Approach 2 :- Use heap instead of sorting in above approach. O(nlogk)

Page No.
Date 10/12/21

1st Choice

Variation - The given array is in sorted form.

★ Approach-3 ~~O(log n)~~ O(log(n-k)+k)

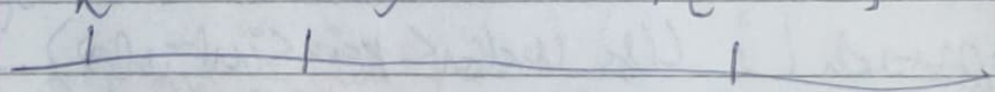
Using Binary search

Take window of mid to mid+k, slide on the array by condition of binary ~~search~~ search.

We compare the distance between

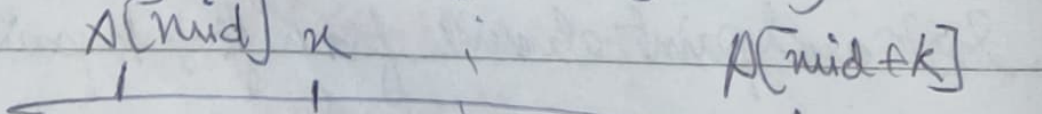
x - A[mid] and A[mid+k] - x

Case 1: x - A[mid] < A[mid+k] - x



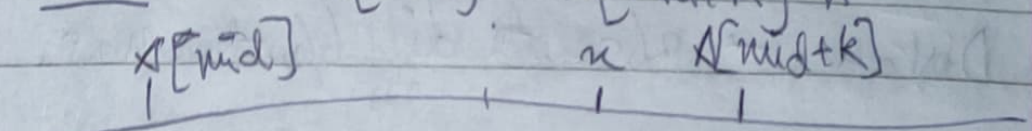
Window needs to move left.

Case 2: x - A[mid] > A[mid+k] - x



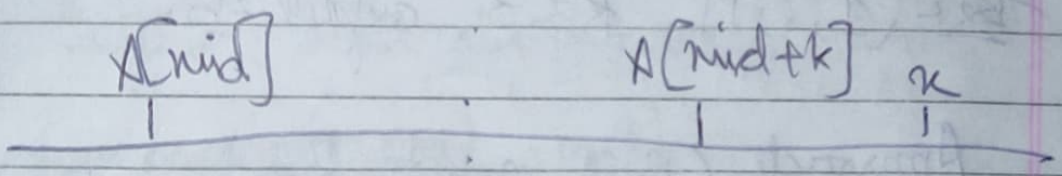
Window needs to move ~~right~~ left.

Case-3:- $x - A[mid] > A[mid+k] - x$



~~Add~~ Window needs to move right.

Case-4:- $x - A[mid] > A[mid+k] - x$



Window needs to move right.

Code:

```
l ← 0, r ← n-k-1
while (l <= r) {
    m = l + (r-l)/2
    if (x - arr[m] <= arr[m] - x) {
        r = m-1 // move left.
    }
    else {
        l = m+1 // move right
    }
}
return last location of window.
```

Page No.
Date

Find Kth Smallest Pair Distance

Distance → absolute diff b/w two nos.

Given an integer array, find k^{th} smallest dist among all pairs.

Ex:- [1, 6, 1] k = 3 output = 5.

Approach (Painter Partition Prob.)

Use Binary search on answer.

~~Sort~~ Sort the array.
Our answer lies b/w 0 and (max - min).

i) guess the answer ~~can~~ and count possible no. of pairs with diff <= guess.
To do that,

Use sliding window approach.
We maintain the loop invariant: left is the smallest value such that arr[right] - arr[left] <= guess. Then add the number of possible lefts for the right.

Burst Balloons - Leetcode

n balloons, each ~~pa~~ painted with a number.
Burst all the balloons.
when burst ith balloon we get nr[i-1]*nr[i]*nr[i+1] coins. If i-1 and i+1 goes out of bounds multiply 1 with 1.
Output: max coin that we can earn.

Approach (MCM) $O(n^3)$

Reverse thinking

~~If the ith balloon is burst last and~~
first of all, to compensate for out of bounds, Append 1 to both the ends of the array.
Now we will try combinations for left+1 to right-1.

the i^{th} balloon here will be burst last and the rest of the balloons from left to i-1 and i+1 to right-1 are already processed and burst by recursive call.

1st Choice

Page No.
Date: 13/12/21

Code:-

```
int mcm(arr, l, r)
    if(l==r-1) // as we have already eliminated
                out for bounds, this
                will result in 0.
    return 0;

    int ans=0;
    for(i=l+1; i<r; i++)
    ans = max(ans, arr[l]*arr[i]*arr[r]
                + solve(arr, l, i)
                + solve(arr(i, r));

    return ans;
```

⇒ dp

Initialize and append (0 both sides of the array ^ in main function.

Minimum cost to merge stones - LC Hard

There are n piles of stones arranged in a row. The i^{th} pile has stones[i].

A move consists of merging exactly k consecutive piles into one pile. And the cost of the move is equal to the total number of stones in these k piles.

Return minimum cost to merge all piles of stone to one pile. If impossible, return -1.

Prefer Leetcode solution OR Video

Too Hard !

1st Choice

Page No.
Date

Binary Indexed Tree / Fenwick Tree

Binary No.

$x = 110110100$

Can be represented as

$x = a1b \rightarrow a$ all 0's $(000\ldots0)$

↳ Rightmost set bit.

$-x$ = 2's compliment of x

$= (x)' + 1$

$= (a1b)' + 1$

$= a'0(111\ldots1) + 1$

$= a'1(000\ldots0)$

$= a'1b$

$x = a1b$ and $-x = a'1b$

$\therefore x \& (-x) = (00\ldots0)1(000\ldots0)$

↳ Rightmost set bit of x

To remove rightmost set bit of x

do

$x - (x \& (-x))$

- Binary indexed tree is an array. BIT[N]
- Every index stores a partial sum.

i → index

j = remove last set bit of (i)

bit[i] will store sum from j+1 to i.

Ex:

sum(1, 13) = bit[13] + bit[12] + bit[8]

```
int sum(int i){
    int ans = 0
    while(i > 0){
        ans += bit[i]
        i = i - (i & (-i))
    }
    return ans
}
```

(l, r) Query = sum(r) - sum(l-1)

How to construct bit array:-

Suppose we want to add x to the ith element.

Initially put all zeros in the bit array.

Use 1 based indexing!

To update, update indices by adding the last set bit to the number in every iteration.

```
update(int i, int x){
    while(i<=N){ → always adds not updates.
        bit[i] += x // update
        i = i + (i&(-1)) // next.
    }
}
```

Stock span Problem

Given a series of n daily prices of a stock.

Span S_i of the day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day.

Print span values at every day.

Example:- [100, 80, 60, 70, 60, 75, 85]

output:- 1 1 1 2 1 4 6

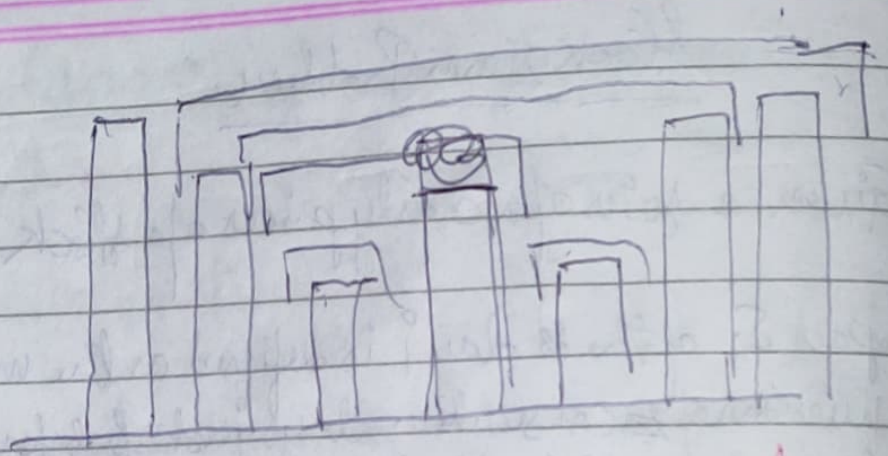
Approach - Standard problem of stack.

Time:- O(N) Space O(N).

Span[i] = distance from nearest greater element to left

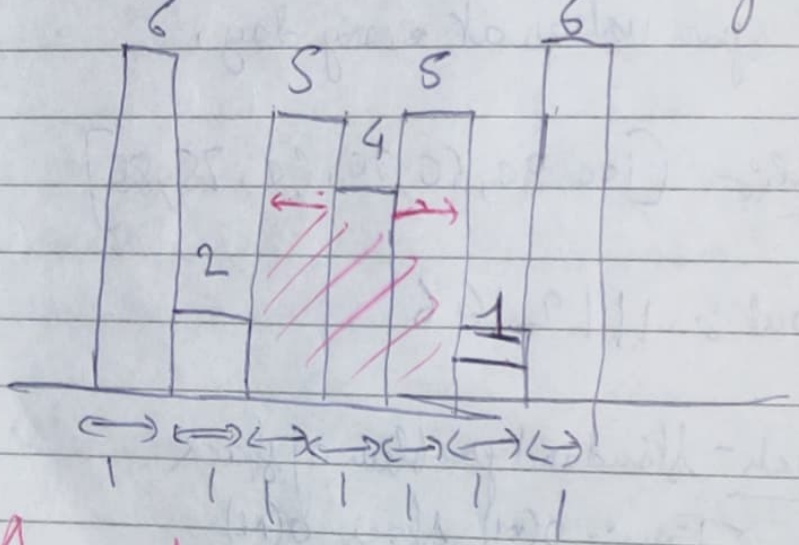
Find nearest greater element to left using stack and keeping its index with it.

Then simply calculate the distance.



Maximum Area Histogram

arr[]: {2 5 4 5 1 6} → Height



Approach

A histogram can be extended to its left and right histograms to subtend a rectangular area until and until

height[left] ≥ height[i] and height[right] ≥ height[i]

We will extend a histogram from both sides until we get an element smaller than its height.

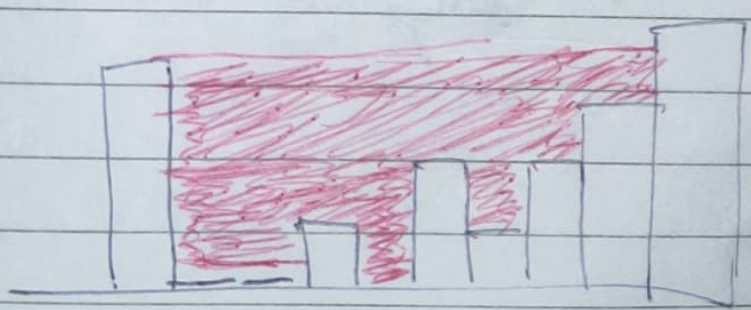
So basically, to extend every histogram we will find nearest smaller element to left and nearest smaller element to right for it. And then check the area occupied by the extended histogram.

ans = max(ans, (nsr[i] - nsl[i] - 1) * height[i]).

Maximum Area Rectangle - Binary Matrix

Approach → Prefer Video of Aditya Verma.

Rainwater Trapping



1st Choice

Page No.
Date

Approach

Area of Water = $\sum$ (height of water above each building.

Calculated by

min(max to left, max to right) - height(i).

Precompute max to Left and max to Right arrays.

Then calculate ans.

O(N)

★ Rainwater trapping and minimum area histogram are really different, just look similar. Hey.

Pythagorean Triplets

$a^2 + b^2 = c^2$

a, b and c are integers.

They can also be written as

$a = m^2 - n^2$

$b = 2 * m * n$

$c = m^2 + n^2$

as,

$a^2 = m^4 + n^4 - 2 * m^2 * n^2$

$b^2 = 4 * m^2 * n^2$

$c^2 = m^4 + n^4 + 2 * m^2 * n^2$

$\Rightarrow a^2 + b^2 = c^2$

This will generate these triplets in $O(K)$ time where K is no. of pythagorean triplets required.