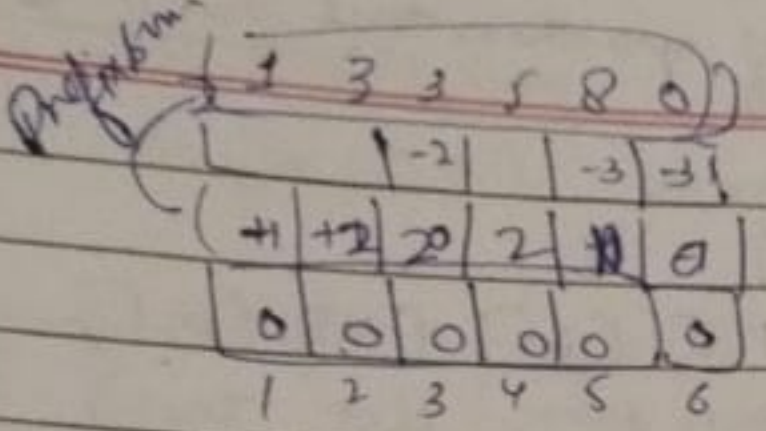
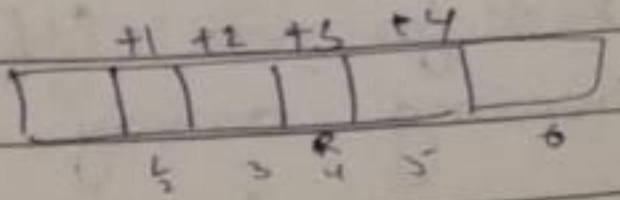
Page No.

Date





Q queries

L R

[2 5]

(i) (n+2,0) prefix 1 (+1, -1) → prefix 1(L) += 1

prefix 1(R+1) -= 1

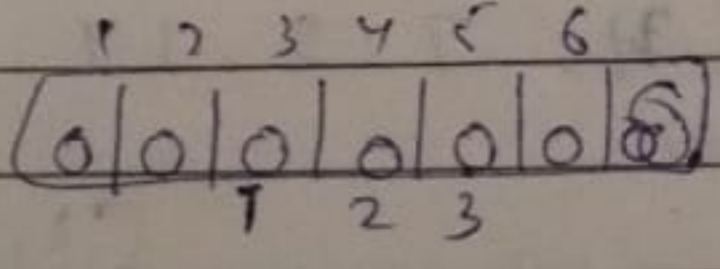
(ii) (n+2, 0) prefix 2 → prefix 2(R+1) -= (R-L+1)

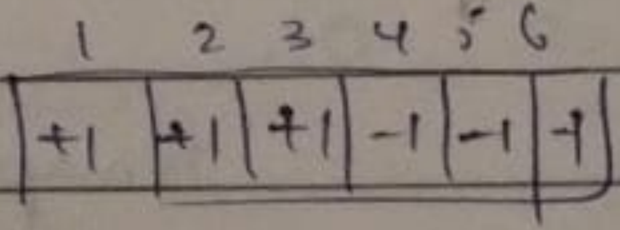
→ for open sequences ; need to calculate prefix sum of prefix1.

(iii) i=1 to n prf1[i] += prf1[i-1]

i 1 to n

ans [i] = ans[i-1] + prf1[i] + prf2 [i]





1 2 3

1 2 3

ans 1365830

Page No.
Date 10. 12. 25

(1) Prefix Sums

→ Create a prefix array . O(n).
→ Calculate subqueries O(q) } O(1 q)
→ for each query O(1)
i.e.

prefix [i] = prefix [r] - prefix (l-1)
Total complexity :- O(n+q)

(2) Maximum Subarray Sum

again using prefix sum instead of Kandane's Algo.

arr =	x	-5	2	3	-1	5	-8
prefix =	0	-5	-3	0	-1	4	-4

Using 3 variables only.

* End-point fixing

		initial					
Step(1)	prefix sum =	0	-5	-3	0	-1	4
(3)	minprefix sum =	0	-5	-5	-5	-5	-5
(2)	ans =	-18	-5	2	5	5	5	9

ans = prefix sum - min prefix sum
ans = Math. max(ans, prefixsum - minprefixsum)

(3) No. of odd subarrays (having odd sums).

arr = [3 5 1 2 3]

sum [l, 5] = odd

prefix [5] - prefix (l-1] = odd

if (even) then ⇒ even - odd = odd
if (odd) then ⇒ odd - Even = odd

arr = [3 5 1 2] 8

curr-prefix sum =	0	3	8 ← check	9	11
odd-prefix sum =	0	1		2	3
even-prefix sum =	1		2		

$ans = 0 + 1 + 1 + 2 + 2 = 6$

Ques) Sum of subarray sum ending at index R & starting anywhere after or at index L.

$$\begin{array}{l} [L, R] \\ [L+1, R] \\ [L+2, R] \\ \vdots \\ [R-1, R] \\ [R, R] \end{array} = \text{sum} \left[\begin{array}{l} \text{prefix}[R] - \text{prefix}(L-1) \\ + \text{prefix}(R) - \text{prefix}[L] \\ \qquad\qquad\qquad\qquad [L+1] \\ \qquad\quad \vdots \qquad\qquad \vdots \\ \qquad\qquad\qquad \text{prefix}[R-1] \end{array} \right.$$

$$ans = (R-L+1) \times \text{Prefix}[R] - \left[P_0P[R-1] - P_0P(L-2) \right]$$

Q)

Contribution Technique

$[3 \mid 2 \mid 5]$

3	3, 2
2	2, 5
5	3, 2, 5

$a_1\ a_2\ a_3 \ldots a_i \ldots a_n$

$$\underset{\text{element}}{\text{Total}\overset{\text{subarrays of}}{\text{contribution}}} = i \times (n-i+1)$$

Longest AP in array :-

$$\underset{\text{array}}{\text{long. AP in}} = \text{Max}\left(\underset{\text{at index 1}}{\text{longest AP}}, \underset{\text{at index 2}}{\text{longest AP}}, \ldots \right)$$

```
prevdiff = arr[i-1]-arr[i-2]
 if (curr = prevff && (currdiff = 1 or currdiff -1))
    fop[i] = fop[i-1] + 1)
 }
else if(currdif ==1 ||currdiff == -1) fop[i]=2;

else {
     fop[i] =1;
    }
int ans=0;
 for( i=1 to n){
     r=i;
     l= i-fop[i])+1;
   ans += (r-[illegible])prefix(r) -[pop[R-1] + pop[L-2]])
```

3	6	9	1	2	3	4
1	2	3	2	2	3	4

ans[i] = ans[i-1] + 1 if arr[i] == arr[i-1] ~~+~~ arr[i-2]
= 2

~~Sum~~(Sum of all AP's):-

O(N)

1	2	3	4	6

All AP's. [1] [1,2] [1,2,3] [1,2,3,4]
[2] [2,3] [2,3,4]
[3] [3,4]
[4] [4,6]
[6]

(i) end-point fixing

Sum(Sum of all AP) = sum of all AP's ending at index 1 + sum of all AP ending at index 2 ... index N

$a_1, a_2, \ldots a_x$

↓

r = 1, l = farthest point ↓ r+l λ_1

r = 2, l = fop(2)

1) prefix O(N)
2) pop O(N)
3) fop O(N)
4) endpoint O(N)

= O(4N)
= O(N)

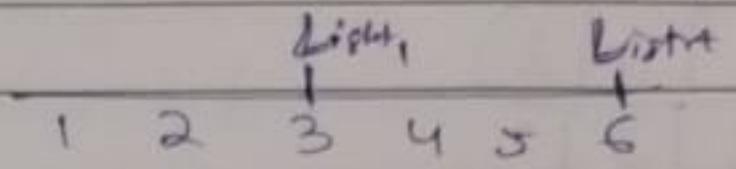
= pref(i-1) + arr(i)
= pop(i-1)

fop(1) = 1
fop[2] = (arr[2] - arr[1] == 1) || (arr[2] - arr[1] = 1)

i→3 to n
curr diff = arr[i] - arr(i-1)

+1, -1 technique

# N → array size
M → No. of lights
α → light source
β → req. points covered by lights



$\alpha = 2$ Light₁ = [1, 5]
Light₂ = [4, -]

$\beta = 2$
ans = 4,5

ex. N = 6
M = 2
pos = [3,6] / [1,5]. [4, 6] O(q+N)
$\beta = 2$

0	1	2	3	4	5	6	7
	+1		+1		-1	-1	-1

prefixsum ~~00 1 1 2 2~~ →

0	1	2	3	4	5	6	7
	+1			+1		-1	-1
0	1	1	1	2	2	1	0

#

1	2	3	4	5	6
0	0	0	0	0	0

queries
(3,5)
(2,4)
(1,3)

$(r-l+1)$

1	2	3	4	5	6
+1	+1	+1	-1	-1	-1

prefix[i] =
prevsum + open + pending
(i-1)

1	2	3	4	5	6
0	0	0	0	0	0

value 1 2 3 4
(that is prev index of that sequence) 1 2 3
1 2

Page No.
Date

Equal no. of zero's & one's in array. [Good subarray].

(i) prefix arr for 1
(ii) " arr for 0

0	1	1	0	1	0

pre 1	0	0	1	2	2	3	3
pre 0	0	[illegible]	1	1	2	2	3

[L, R]

onepref (R) − onepref (L−1) = =
~~[illegible]~~ zeropre(R) − ~~[illegible]~~ (L−1)
zeropref.

$\Rightarrow$ one(R) − zero(R) = one(L−1) = zero(L−1)

x [i] = one(i) − zero (i)
x(R) = x(L−1)

x	-1	0	1	0	1	0

hashmap (storing frequency of hashmap).

[
0 : 3
-1 : 1
1 : 2
)

Count one = 3
Count zero = 3

ans = 0 + 1 + 2 + 1 + 3
= 7

Max. length of subarray which is good subarray:

[0, 1, 0, 1]

[2, 5] = good & max. length.

Page No.:
Date : 11 / 02 / 25

Select subset & subtract 1 from any & to make zero array if yes then true else false.

Array :- [1, 2, 3]

[0, 1] [0, 2, 3]

Queries :- [0, 1] [1, 1] [1, 1] [0, 2] [0, 1]

1) Check for -ve (if -ve then false else true).

Creating array with +1 & -1 technique. ⇒ $\overset{0}{+3}\ \overset{1}{+2}\ \overset{2}{-2}$

prefixsum / prev array ⇒ $\overset{0}{3}\,|\,\overset{1}{5}\,|\,\overset{2}{3}$

compare with given array
if greater then true else false.

Q) Given N no. : a[0] ... a[N-1] find i <= j such that $\boxed{a[i] = a[j]}$ & sum is maximized.

Max. good subarray sum = Max. (

[2, 2, 3, 2, 5]

prefix [] = 0, 2, 4, 7, 9, 14

hmap [
arrvalue
2 : 0
3 : 4.
5 : 9
]

In hashmap storing minprsum at previous index.

[2, 2, 3, 2, 5]
2 4 7 9 14

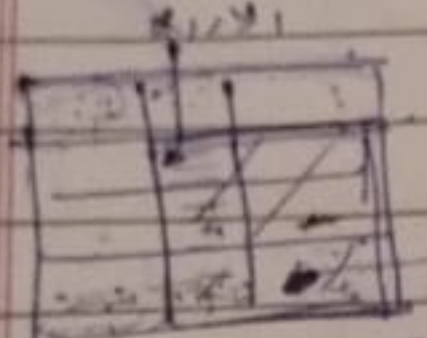
Output 7 2 4

2D Prefix sum

prefix [i] [j] = sum of submatrices $(1,1) \rightarrow (i,j)$

$\boxed{304}$

prefix [i] [j] = prefix [i-1] [j] + prefix [i] [j-1] − prefix [i-1] [j-1] + arr[i] [j]



ans = pref(x_2)(y_2) − pref(x_2)(y_1−1) − pref (x_1−1) (y_2) + pref(x_1−1) (y_1−1)

Converting 2-D to 1-D :-

Count submatrix with sum equals ~~to~~ 'k'.

$$\begin{array}{c|cccc} & y_1 & y_2 & y_3 & y_4 \\ \hline & 3 & 1 & 4 & 1 \\ x_2 & 4 & -3 & 3 & 4 \\ x_3 & 5 & -2 & 4 & 1 \\ x_4 & 6 & 1 & 3 & 2 \end{array}$$

if k = 11

fixing the rows (by merging them)

• (x_2, x_3, x_4) col^n can be

sum = $\underbrace{15, -4}_{k=11}$, 10, 7

Col :- y_1, y_2

$$\begin{bmatrix} 3 & 1 & 4 & 1 \\ 4 & -3 & 3 & 4 \\ 5 & -2 & 4 & 1 \\ 6 & 1 & 3 & 2 \end{bmatrix}$$

Page No.
Date

Q.1) Count no. of substring with equal no of a, b, c in string of length N.

2) 1124 leetcode
124 "] Sorting + heap

3) Count no. of submatrix with equal no. of zeroes & ones / with all one (Imp)

4) Count the largest area of submatrix with k.

5) Count largest area of submatrix with sum greater than k given all elements are positive in the matrix.

6) 3026 leetcode.

(Sol^n 1)

$$S = \overset{1}{d}\,\overset{2}{a}\,\overset{3}{b}\,\overset{4}{c}\,\overset{5}{d}\,\overset{6}{a}\,\overset{7}{b}$$

		1	2	3	4	5	6	7
$prefix_a$ =	0	0	1	1	1	1	2	2
P_b =	0	0	1	1	1	1	1	2
P_c =	0	0	0	1	1	1	1	1

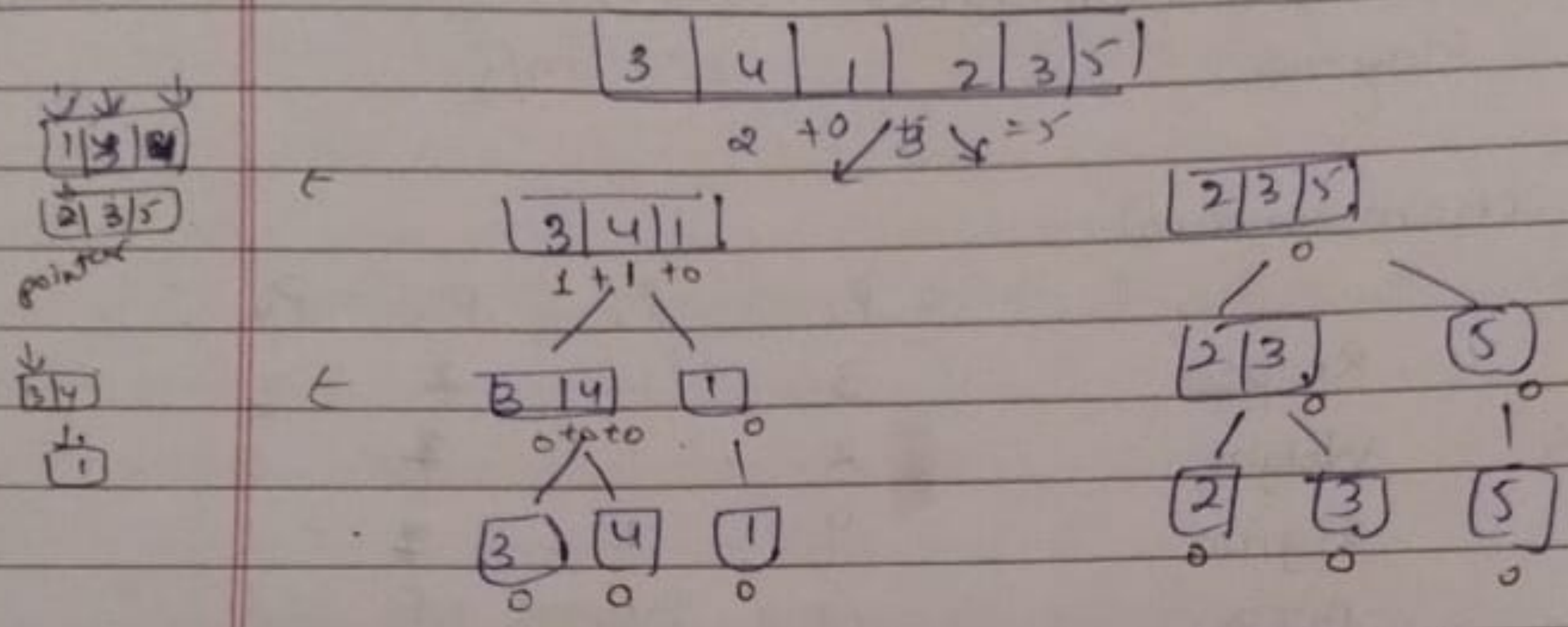
	0	1	2	3	4	5	6	7
V_{ab} =	0	0	0	0	0	0	1	0
V_{ac} =	0	0	1	0	0	0	0	1

pair = [(0,1), (1,0) (0,0)]

Inversion

3	4	1	2	3	5

such that $i<j$ & $arr[i] > arr[j]$
ex(4,1)



3 cases

	Tukda 1	Tukda 2
(i)	i, j	
(ii)		i, j
(iii)	i	j [in different sub-arrays]

Related i.e. to 1124 (Leetcode) [equal no. of zero's & one's + inversion].

	1	2	3	4	5	6	7
ex:-	1	1	0	0	0	0	1

1 → tiring day
0 → Non-tiring day

Good subarray
= no. of 1's > no. of 0's

	0	1	2	3	4	5	6	7
P_0	0	0	0	1	2	3	4	4
P_1	0	1	2	2	2	2	2	3
$X = (P_1 - P_0)$	0	1	2	1	0	-1	-2	-1

$$P_1 r - P_0 r > P_1(l-1) - P_0(l-1)$$

$$\boxed{x(r) > x(l-1)}$$

(Inversion) "this is for less than not equal to" $x(l-1) < x(r)$ $l<r$

pairs 3 4 5 4 3 1 0 0 = 20 $l-1<r$

$^{8}C_2 = 28$ (size of (x); pairs)

ans = 28 − 20 = 8

(Qoe2 Q)

Comparators

[lambda expression] Collections sort (players, $(P_1, P_2) \rightarrow P_2.score - P_1.score$);

N Players   K :- m/2

M (Characteristics)

	P_1	P_2	P_3	P_4
Respect	3	1	7	6
Weight	2	1	7	6
Sigma	4	1	7	1
Aura	1	1	5	1

n, m

arr [n] [m]

0 to n

0 to m

arr [ ].

sort (arr.

if -ve then $o_1 > o_2$

0 then $o_1 = o_2$

+ve then $o_2 > o_1$

int compare (Integer o1, Int o_2)

$$R + 2R + 3R - - -$$

$$x_{paratha} = R\,(1+2+3 - - -)$$

$$= R \times \frac{(x)\times(x+1)}{2} \le mid$$

(i) if ~~[illegible]~~ time = 10 & R = 1

$$1 \times \frac{(x)(x+1)}{2} \le 10$$

mid $\boxed{x(x+1) \le 20}$

only true values.

Largest submatrix have sum k.

(l, r)   (length - l+1

$O(\log_2 \min(N,m) \times NM)$

No. of matrix with values less than k. ~~or equal to~~

Upperbound (R)
Lowerbound (L)

ex: K = 4

2	3	4	5
3	4	5	6
7	8	9	10
11	12	13	14

ans = index of (4) i.e. 3

ans = 3 + 2 (index of (4))

ans = 3+2 = 5

Complexity - $O(n+m)$

Page No.
Date 13.02.25

Shortest substring with atleast one of each character present in alphabet.

⇒ Space is length
⇒ func^n is monotoneous. bcs if mid is shortest substring contains all characters then mid+1 will also contain no need to check in mid+1 area.

ex1- aaabb~~c~~c

low = 3 high = 7

mid = 5

✓

Yes

ans = 5
hi = mid-1

Sol^n (1) Making prefix array. end = mid.
occurances start = ~~[illegible]~~ +1
of characters

(i) a = pr_a(end) − pr_a(start −1)
⋮
26. ← No of characters in alphabet.

$O(\log_2 N \times N \times K)$

Sol^n (2) Sliding Window (for fixed array)

Auxillary Array
mid = 3
a a a b b c c

a	b	c
~~0~~	~~0~~	0
~~3~~ ~~2~~ 1	~~1~~ 2	

# of count of zero
zero = ~~0~~
zero = ~~3~~ ~~2~~ 1

Sol^n 3) 2-Pointer App. (Variable length)

Time $O(N+N) = O(2N) \Rightarrow O(N)$

bcs start & end pointers each travels N times.

arr [illegible]
[illegible] 2 1 3 3 4 5

sort
[5 | 1 | 2 | 2 | 4 | 5] val
1 4 3 2 5 0 index

Binary Search

Queries (sort)
[illegible] [illegible] n = 6 q = [9]
[1, 2]
[1, 3]
3 4 5 6

Monotonic Appⁿ of Binary Search.

Longest subarray sum less than k.

Does exist a subarray of length 'X' having sum ≤ k

Claims (i) Starting [illegible] hogya hai
(ii) [illegible] ho gya hai

SPOJ

Ques) In a factory of [illegible] [illegible] [illegible] [illegible] R_i, A_i
[illegible] [illegible] [illegible] k [illegible] [illegible] [illegible]
[illegible] [illegible] [illegible]

Time - 1 mid [illegible]

[illegible]

lo = mid

hi = mid - 1

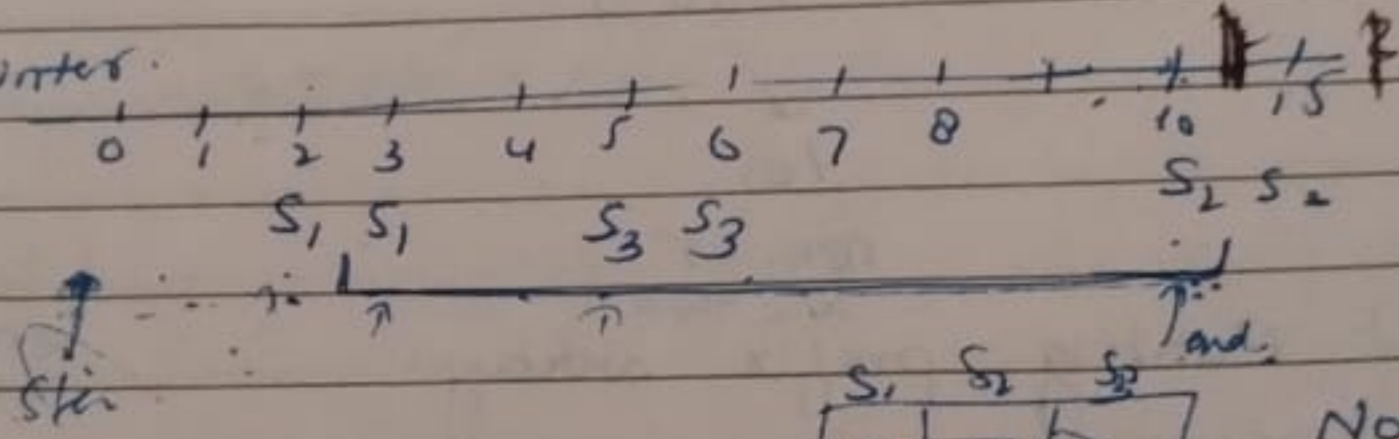
Space Complexity : $O(k)$ ← No. of characters.
→ for array

[0|b|0]
& variable
for zero count

Given array of pairs:-

$$arr = \begin{bmatrix} \overset{s_1}{\begin{pmatrix} 2 \\ 3 \end{pmatrix}} & \overset{s_2}{\begin{pmatrix} 10 \\ 15 \end{pmatrix}} & \overset{s_3}{\begin{pmatrix} 5 \\ 6 \end{pmatrix}} \end{bmatrix}$$

Soln(1) 2-pointer.



O (max (time))

s_1	s_2	s_3
~~0~~	~~0~~	~~0~~
2	1	1 2

No. of zeros = ~~3~~ 2

Sol 2) Coordinate Compression

Soln 3) ~~Use Hash Map~~ Pair bnao & then sort.

$$\begin{pmatrix} 2 \\ s_1 \end{pmatrix}\begin{pmatrix} 3 \\ s_1 \end{pmatrix}\begin{pmatrix} 5 \\ s_3 \end{pmatrix}\begin{pmatrix} 6 \\ s_3 \end{pmatrix}\begin{pmatrix} 10 \\ s_2 \end{pmatrix}\begin{pmatrix} 15 \\ s_2 \end{pmatrix}$$

then apply 2-pointer.

As now, we will only traverse time where student exist.

find cnt pairs ~~that~~ whose diff $\le k$.

-3	5	10	12	13	14

$a_l \underbrace{\cdots\cdots}_{a_{l+1} \quad a_{R-1}} a_R$

$(R-1)-(l+1)-1$
$= r-l-1$

k^{th} ~~largest~~ smallest difference - (719)

~~end~~

st=0
en=1

```
count no. pairs ~~whose~~ diff less than k {
while( st < arr.size() ){
        while(en< arr.size() & arr[en] - arr[st]
            <= mid) { en++;
            ans += (en-st-1)
        }
    }
    st++;
}
return ans;
}
```

int small

(1) Sort given array.

+ve values array.

Find Kth smallest subarray

Course Schedule III (Leetcode)

(630) sorting hereby comparator.

Array.sort(courses, (a,b) → Integer.compare(a[1], b[1]))

Courses = [[$\overset{dur}{100}$, $\overset{en}{200}$], [200, 1300], [1000, 1250], [2000, 3200]]

Duration	End
100	200
200	1300
1000	1250
2000	3200

100, 200 , (1000, 1250), (200, 1300)

Sort endpoints:-

$$\begin{array}{c} en \\ dur \end{array} \left[\begin{array}{cccc} 200, & 1250, & 1300, & 3200 \\ 100 & 1000 & 200 & 2000 \end{array}\right]$$

→ Using max heap to store duration of selected courses.

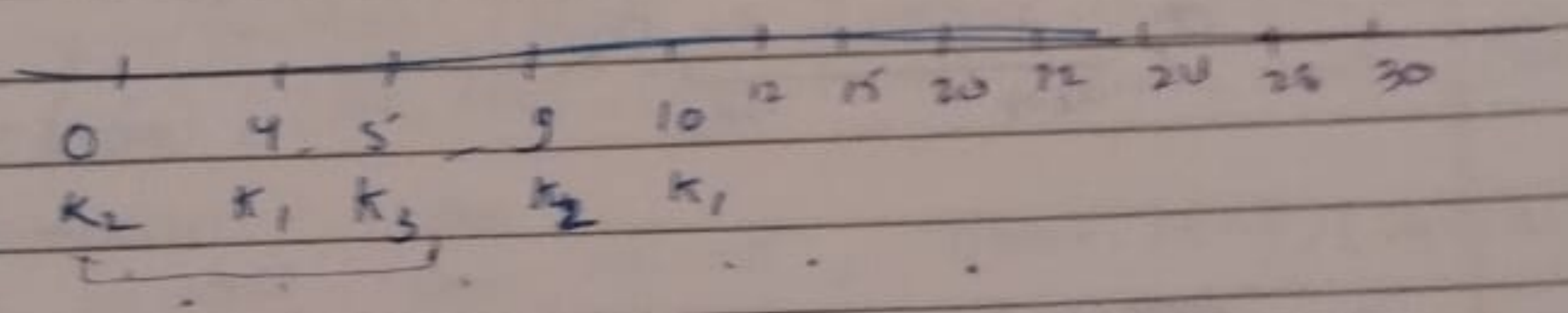
int total = 0;

Leetcode (632)

Smallest Range Covering Elements from K Lists.

k_1 k_2 k_3

nums = [[4, 10, 15, 24, 26], [0, 9, 12, 20], [5, 18, 22, 30]]



Date : 14 | 02 | 25

Longest increasing Subsequence.

Given array :-

10	2	-1	5	-2	9	0

(10, -1) (2, -1) [2, 5] -1, -2

Bruteforce [[illegible], [illegible]]

1	1	1	2	2	3	1
10	15	11	15	-2	19	0

length	0	1	2	3	4	5	6	7
value	-∞	~~∞~~	~~∞~~	~~∞~~	∞	∞	∞	∞
	-∞	10	~~15~~ 11	15				

$O(N \log N)$

(1) Array (n+1)
value ka upperbound calculate.

354 Leetcode (Russia doll Envelope)

(1) first element sort.
(2) then apply LIS on second element.

$$\begin{pmatrix}1\\2\end{pmatrix} \begin{pmatrix}2\\5\end{pmatrix} \begin{pmatrix}3\\6\end{pmatrix} \begin{pmatrix}7\\7\end{pmatrix} \begin{pmatrix}7\\9\end{pmatrix} \begin{pmatrix}8\\20\end{pmatrix}$$

1st loop is for computation
2nd loop for updation.

Page No.:
Date: 14/02/25

Longest increasing subsequence.

Given array:-

10	2	-1	5	-2	9	0

Bruteforce (10, [illegible]) (10,-1)(2,-1) [illegible] (2,5) -1,-2

1	1	1	2	2	3	1
10	15	11	15	-2	19	0

length	0	1	2	3	4	5	6	7
value	$-\infty$	∞	∞	∞	∞	∞	∞	∞
	$-\infty$	10	~~15~~ 11	15				

① Array (n+1) $O(N \log N)$
value ka upperbound calculate.

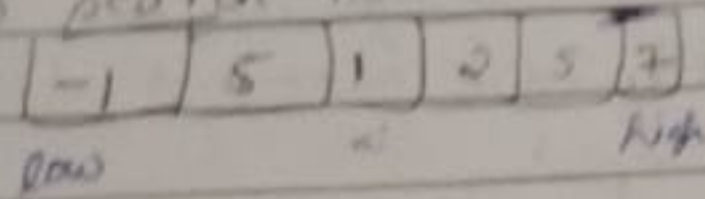
354 Leetcode (Russia doll envelope)

1. first element sort.
2. then apply LIS on second element.

$\binom{1}{2}$ $\binom{2}{5}$ $\binom{3}{6}$ $\binom{7}{7}$ $\binom{7}{9}$ $\binom{8}{20}$

1st loop is for computation
2nd loop for updation.

Ques) To search no. in binary searchable or not

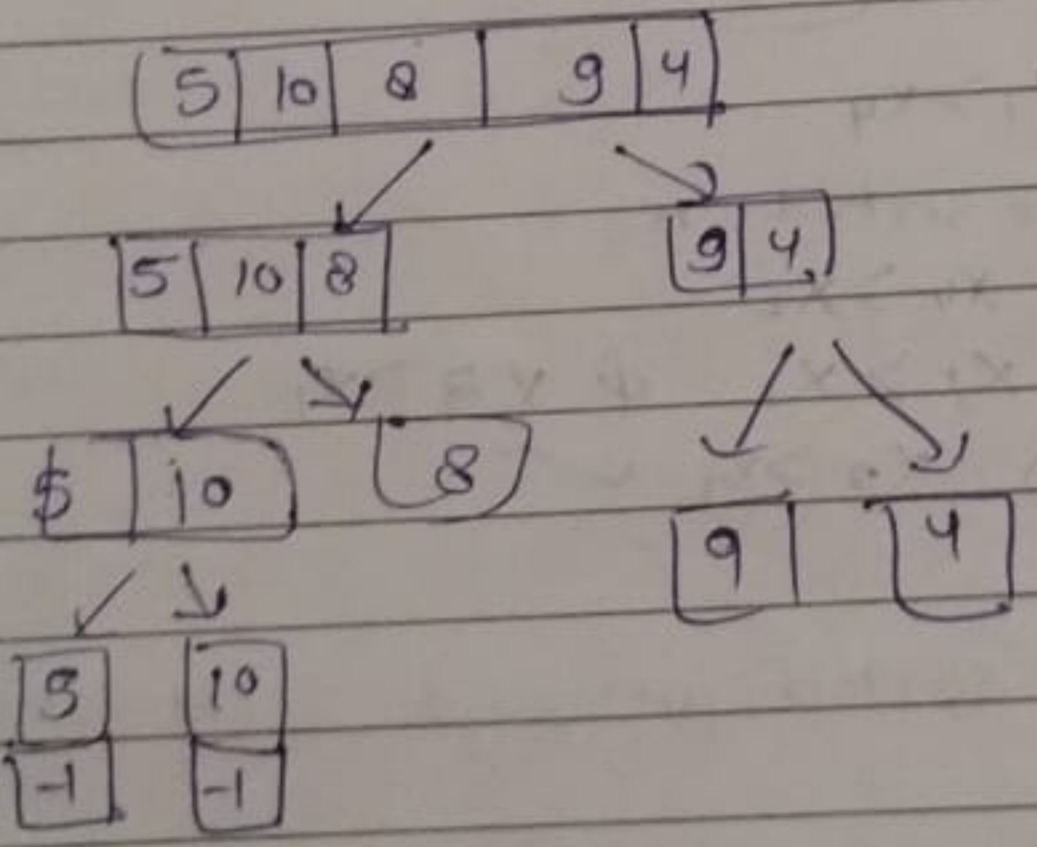


left me saare chote from pivot
right me saare bade " "

If no. are +ve in array then after iterating which are not bin searchable then make them -ve. (so that we can store them in O(1)).

For -ve array add an offset & make them +ve. Then apply same approach.

Next Smaller Element



val	5	10	8
index	1	2	3

left part

$T(n) = 2T(n/2) + O(n)$

$= O(N \log N)$

value	9	4
index	4	5

Right part

Page No.
Date

1793 Max score of a good subarray.

input :- $nums = [1, 4, 3, \overset{k}{7}, 4, 5]$ $k = 3$
(i, j)
Score = $\min(nums[i], nums[i+1] \dots nums[j]) *$
$(j - i + 1)$
good subarray $i <= k <= j$

Date :- 15 / Feb / 2025

Ques) given anst ~~no~~ of array of integers find no of pairs such that $(x * y) \% k = 0$

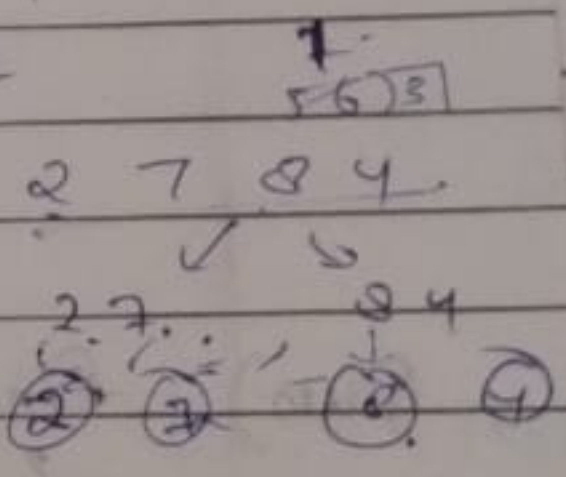
ex $k = 3$ in :-

2	7	8	4

$O(N + k \log k)$

low high.
2, 4, ⑦, 8
    mid

lo = 0   h = 4
mid = 2



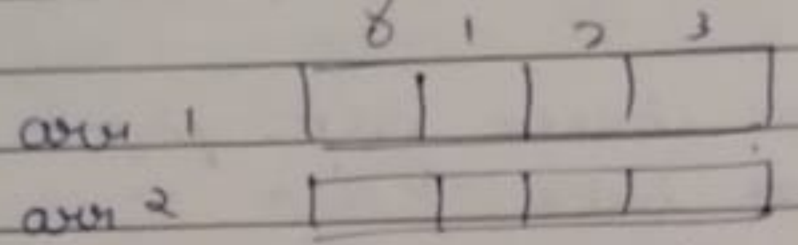
(i) if values are less &
size is large
Make frequency
array $O(k^2)$
k = max. value.

(ii) if size is less & values are large
then $O(n^2)$.
For same values in ans add ${}^{i}C_2$ ← index of that values.

$(a+b)\% mod = (a\% mod + b\% mod)\% mod$
$(a-b)\% mod = (a\% mod - b\% mod + mod)\% mod$
$(a \times b)\% mod = (a\% mod \times b\% mod)\% mod$

Page No.
Date

Ques) given 2 sorted arrays & merge them to take k elements.



k=5
x=2.

Apply binary search.

low = 0 high = n i.e. 4 here

mid = 2

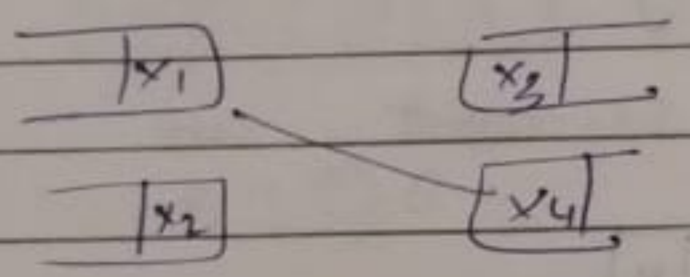
↓

NO

↓

low=0 high = 2-1 = 1

mid = 1



if $x_1 > x_4$

for sorted arr

$x_4 > x_2$

$x_1 > x_2$ & $x_3 > x_1$

$\Rightarrow x_3 > x_2$ ✓

Median of 2 sorted arrays (4 leetcode)

493 Reverse Pairs

315 Count of Smaller No.s After Self.

Apply Inversions

$$O(N \log N) \quad \begin{array}{l} \text{value} \rightarrow \\ \text{count} \rightarrow \\ \text{index} \rightarrow \end{array} \begin{pmatrix} 5 \\ 0 \end{pmatrix}_{0} \begin{pmatrix} 2 \\ 0 \end{pmatrix}_{1} \begin{pmatrix} 6 \\ 0 \end{pmatrix}_{2}$$

↙ ↘ break

then merge.

Length

Array of zero's & one's. ~~Equal~~ no. of subarrays having equal no. of zero's & ones whose length is divisible by 'k'.

$x(l-1) = 0$

$(r-(l-1)) \% k = 0$

Index%k:	0	1	2	0	1	2	0	1
arr	0	1	0	0	1	1	0	1

$x(i)$ 0 1 0 -1 0 1 0 1

= prf(one) prf(0)

map($x[i]$, $i\%k$)

$O(N \log N)$

(2949) Count Beautiful Substrings III.

i. vow = cons.

(ii) $(vow * cons) \% k = 0$

$\Rightarrow (vow^2) \% k = 0$

$$k = p_1^{e_1} \, p_2^{e_2} \ldots p_k^{k}$$

So, vow should be :- atleast $p_1^{\lceil t_1/2 \rceil} \, p_2^{\lceil t_2/2 \rceil} \cdot p_3^{\lceil t_3/2 \rceil}$

and for $P_1 = 2$

$\lceil t/2 \rceil + 1$

(363) Max Sum of Rectangle No Larger than k

$prf(r) - prf(l-1) < k$

$pref(l-1) > prf(o) - k$

(iii) $\boxed{218^3}$

$\left(\alpha, \frac{k}{\alpha} x\right)$ → Check $\alpha \% \left(\frac{k}{\alpha}\right)$

then total pair = $\text{freq}(\alpha) \times \left(\text{Multiples}\left(\frac{k}{\alpha}\right) - 1\right)$

else

total pairs = $\text{freq}(\alpha) \times \text{Multiples}\left(\frac{k}{\alpha}\right)$

Compl. :- $O\left(k \log k + k^{1/3}\right)$

$= O(k \log k)$

Steps:-

# Calculate freq

# Multiples freq k

# k ke divisors pe iterate in $k^{1/3}$

↳ α, $\frac{k}{\alpha}$

go & check condition.

Ques) Find shortest substring when duplicated gives you original string

$S = [\overset{0}{a}ab\,aab\,aab\,aa\overset{12}{b}]$

len = 12

factors
↳ 1 check for 1 len
↳ 2 — 2
↳ 3 — 3 ✓ we can duplicate.
↳ 4
↳ 6
↳ 12

$O(n \times n^{1/2})$