



Date

Page

Sorts

o Selection Sort

= Select min

Sorteflow

down

bottom

[min] 46

46

bottom

13	46	24	52	20	9	
0	1	2	3	4	5 - m	0

bottom

13 46

bottom

13 46

bottom

9 46 24 52 20 13 - m step 1

9 13 24 52 20 46 - m step 2

9 13 20 52 24 46 step 3

9 13 20 24 52 46 step 4

9 13 20 24 46 52 step 5

(--j; i <= j; i - m = j) not

}

= for (i = 0; i < i = m - 2; i++) {

{ min = i;

method for (j = i; j < i = m - 1; j++) {

min = j;

swap

if (arr[i] < arr[min]) {

min = j;

}

swap (arr[min], arr[i]);

(i+1) 0

}

[bottom] + 28 ←

(i+1) 0

(min, bottom)

Time Complexity $O\left(\frac{n(n+1)}{2}\right)$

$= O(n^2)$

Best, avg, worst $\Rightarrow O(n^2)$

~~efficiency~~

#

Bubble Sort

~~fixed position~~

Compare two elements swap if needed

~~minm fix~~

At every round max will be set
end of array

	P	05	52	45	34	E1
0	n-1	4	E	S	L	O
0	n-2					
107+2	n-3	05	52	45	34	P
50	n-4	05	52	45	E1	P
E:	34	45	52	05	E1	P
P0	1	34	52	45	E1	P
2		52	34	45	05	E1

for (i=n-1 ; i >= 1 ; i--)

{

for (j=0 ; j <= i-1 ; j++) {

{

if (arr[j] > arr[j+1]) {

swap

j = minm }

} No swap happen
then break.

} (minm step > [i] steps) Ti

: i = minm

O(N²)

→ (Worst Complexity)

O(N)

→ Best (Sorted)

(modification)

$$\left(\frac{(1+m)m}{2} \right) O$$

it's balance sum

$$(^2 m) O =$$

(Smallest → + max, avg, best)



0 Insertion Sort

[+, 2, 5, 2, 1, 4, 8, 1, 2]

→ Take element Place in Correct Position [1, 2, 5, 2, 1, 4, 8]

[+] for [(i=0; i < n-1; i++) [2, 1, 8]

{

[0] [2] if [j = (i-1)] [1, 2, 5, 1, 8]

while [j > 0 && arr[j] > arr[j+1]] {

[1] [2] [5] swap [arr[i-1], arr[i]] [1, 5, 2, 1, 8]

[0, +] [2, 5] [1, 2, 1, 8]

{

[+, 1] [2, 1, 8]

[2, 1, 8]

→ $O(n^2)$ → worst [2, 1, 8]

$O(n^2)$ → average

$O(n)$ → Best (Sorted)

[2, 1, 8, 2, 1, 4, 5, 2, 1, 1]

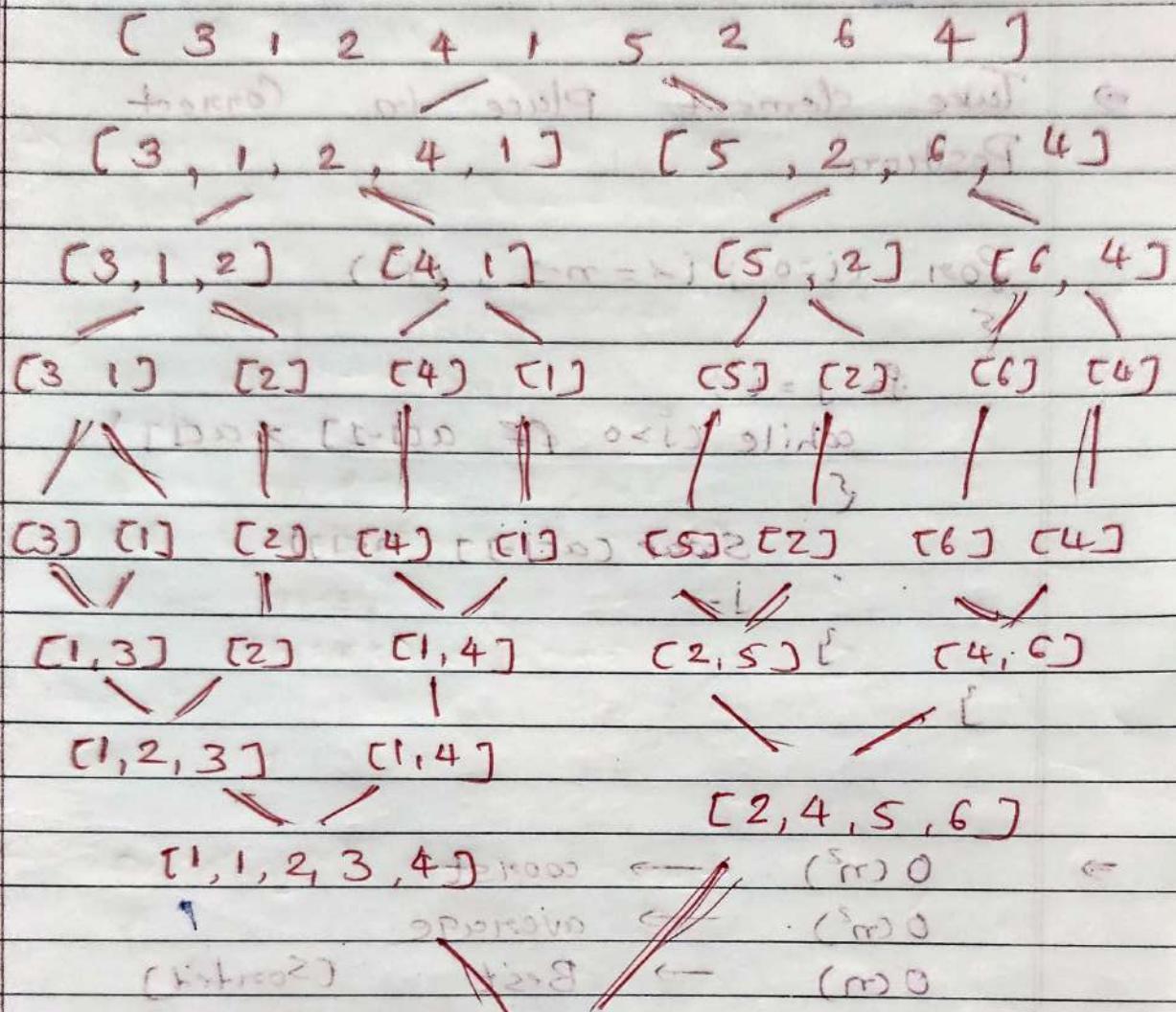
0 Merge Sort

→ Divide and merge



Date _____ / _____ / _____

Page _____

~~tree2~~~~preorder~~[1, 1, 2, 2, 3, 4, 4, 5, 6]~~tree2~~~~preorder~~

searching insert delete



#

merge sort (arr, low, high) { }

{ (arr) (low, high)

$$\text{mid} = \frac{\text{low} + \text{high}}{2}$$

(temp = arr[low : mid])

merge-sort (arr, low, mid);

merge-sort (arr, mid+1, high);

merge ~~sort~~ (arr, low, mid, high)

if (low >= high)

return; ← well = i mod

}

[well - i] arr. = (i) temp

merge (arr, low, mid, high)

{

temp
arr → ()

left ← low; ← (high) 0

right = mid + 1;

while (left <= mid for

right <= high)

{

+ temp → (mid) 0

if (arr[left] <= arr[right])

temp.add (arr[left])

left++;

else

temp.add (arr[right])

right++;

}



while ($\text{left} \leq \text{mid}$)

{ } (dp[0], val, res) here we have
temp.add (arr[left]);
left;

y $(\text{dp}[0] + \text{val}) = \text{bim}$

while ($\text{right} \leq \text{high}$)

{ } (bim, val, res) here we have
temp.add (arr[right]);

(dp[0], right++) ; res = bim

}

(dp[0] = < val) ? i ~

for ($i = \text{low} \rightarrow \text{high}$) { }

{ }

$\text{arr}[i] = \text{temp}[i - \text{low}]$

y (dp[0], bim, val, res) here we have
} () \leftarrow arr[0]

arr[0] \leftarrow arr[1]

O($N \log N$) \rightarrow cost + t[0]

; i + bim = t[0]

? bim \rightarrow t[0]) slides

(dp[0] \rightarrow t[0]

Quick Sort

((t[0]) res \rightarrow (t[0]) res) ? i

((t[0]) res) b[0]. res[1].

; ++t[0];

res

((t[0]) res) b[0]. res

; ++t[0];

{ }

# Second largest

largest = arr[0], second_largest = INT-MIN

for (int i=1; i<n; i++) {

if (arr[i] > largest) {

largest = arr[i];

second_largest = arr[i];

} else if (arr[i] < largest) {

(arr[i] > second_largest) {

second_largest = arr[i];

}

} return second_largest;

}

[2, 4, 8, 5, 1, 1] = 4 (Ans)

[2, 2, 4, 5, 5] = 4 (Ans)

Remove Duplicate

Same id same remove

i=0

for (j=i; j<n; j++)

{

if (arr[i] == arr[j]) {

{

i++;

arr[i] = arr[j];

}

}

return i+1;

Ans



Date

Page

Left rotate array

Normal example, reverse & implement
→ rotate arr with D

$i \leftarrow (i+D) \% n \quad i = 0 \rightarrow i+1 \rightarrow 1$
 $[1, 2, 3, 4, 5, 6, 7] \xrightarrow{\text{reverse}} [6, 7, 1, 2, 3, 4, 5]$

$[4, 5, 6, 7, 1, 2, 3] \xrightarrow{\text{+ 1}} [5, 6, 7, 1, 2, 3, 4]$

$\xrightarrow{\text{DPR = implement}}$

Reverse (start to D)

Reverse (D+1 to end)

Reverse (start to end)

$\xrightarrow{\text{DPR = implement}}$

Union of two Sorted array

arr1[] = [1, 1, 2, 3, 4, 5]

arr2[] = [2, 3, 4, 5, 6]

→ add all to set

remove one by one

Brute force

O(n^2)

$(i=0, j=1 : i=1, j=2)$...

→ Two Pointer

[1, 1, 2, 3] (arr1) [4, 5, 6] (arr2) ↑
↑

i

↑ + 1

[2, 3, 4, 5, 6]

↑

j

↑ + 1

Code

Find missing Number

$$\text{arr}[] = [1, 2, 4, 5] \quad N = 5$$

$\underbrace{1 \ N-1}_{\text{ans}} \ \underbrace{\dots}_{\text{size}} = 1 \ \text{gap}$

ans = 3
Not there b/w 1 to N

→ Brute force

for every 1 to $(N - \text{ans})$ check if it = exist

$O(N^2)$

: Time complexity = $O(N^2)$

→ keep arr[]

make 1 in for all ∞ first

Iteration

find 0 in second iteration

returning with prepared sum #

$O(N) + O(N)$

$O(2N)$

start goal sum ↗

→ Sum all arr[] elements

$(n \leftarrow 0 \text{ to } j) \text{ not}$

Total { Sum of $(n \leftarrow i \text{ to } N) \text{ not}$

$(i \leftarrow j \text{ to } N) \text{ not}$

$\frac{N(N-1)}{2} - \text{sum of arr[]}$

not start goal sum ↘

= ans.

$O(N)$

$(n \leftarrow 0 \text{ to } j) \text{ not}$

$(n \leftarrow j \text{ to } i) \text{ not}$

→ Total XOR → Array XOR

$(x = arr[0]) \text{ if}$

$O(N)$: Time complexity = $O(N)$



Maximum Consecutive ones

$i = 14 \quad [2, 1, 1, 1, 1, 0, 1, 1, 1] = O(N^2)$

$\text{ans}[1] = \underbrace{[1, 1, 0]}_2, \underbrace{[1, 1, 1]}_3, \underbrace{[0, 1, 1]}_2 \quad O(N^2)$

in $O(N^2)$ with growth to $O(N^3)$

$\text{maxi} = 0$

$\text{cnt} = 0$

for ($i=0 \rightarrow n$) do i for i \leftarrow max i

if ($\text{num[i]} == 1$) then

$\text{cnt}++;$ $O(N^2)$

$\text{maxi} = \max(\text{maxi}, \text{cnt});$

else

if ($\text{cnt} \neq 0$) then i \leftarrow max i

maxi \leftarrow i

maxi broad \leftarrow 0 begin

max Subarray with Sum K

Three loop Brute force

for ($i=0 \rightarrow n$)

for ($j=i \rightarrow n$) to min $\left\{ 1, O(n^3) \right.$

for ($k=j \rightarrow i$)

$O(N^3)$ to max $\left. \left(1 - N \right) N \right)$

Two loop Brute force

for ($i=0 \rightarrow n$)

for ($j=i \rightarrow n$)

sum $\leftarrow \text{num[i:j]}$

if ($\text{sum} == k$)

$\text{max} = \max(\text{max}, j-i+1) \left(O(N^2) \right)$



Hashing

Prefix Sum

$x = k$

$\left[\underline{0}, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \right]_{i=1, j=1}$

i
 j

if there exist a subarray with sum k as first element as last element

$$(i = 1, j = 1)$$

$$\text{arr}[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

$$(i = 1, j = 1, k = 3)$$

map

9, 5
8, 4
4, 3
6, 2
3, 1
1, 0

$$(11530) \leftarrow 2T$$

$$\text{len} = 2$$

$$6 - 3 = 3 \text{ but } \text{len} = 1$$

Don't update out #

at this we're getting in arr out

$$9 - 6 = 3 \quad (i \text{ find } + C(j) \text{ sum})$$

$$6 - 2 = 4$$

\therefore len for sum = 3 is, after

$$5 - 2 = 3 \quad \text{not out}$$

$$(1150)$$

∴ update $\boxed{\text{len} = 3}$

$$Tc \rightarrow O(n \log N) + O(n)$$

$$Sc \rightarrow O(n)$$

($s = 11, 8, 2, 3, 5, 7$)

$$(arr \leftarrow arr)$$

$$(\text{real } X \leftarrow \text{out})$$



Two Pointers

[1, 2, 3, 1, 1, 1, 1, 3, 3] → 9

↑
i j

Slow pointer → takes small i

→ $\text{sum} \leftarrow \text{nums}[i] \rightarrow \text{num}$
 $i++$; Initially fast

if ($\text{sum} \geq k$)

[ε, sum] = $\text{nums}[i:j, s, t] = []$ []
 $\epsilon = i$ $j++;$

TC $\rightarrow O(N^2)$

$\Omega = \text{const}$

$\Sigma = \text{const}$ $\text{and } \epsilon = \epsilon - 2$

Two Sum first

2, P
4, S
E, P
S, D
I, E
O, I

two num in nums such that the
 $\text{nums}[i] + \text{nums}[j] = k - P$

$s \rightarrow 0$

Brute Force $\epsilon = \text{const}$ not const \therefore

two for loop $\epsilon = s - 2$
 $O(N^2)$

$\epsilon = \text{const}$

stuck

Hashing

(W) + (N) $\rightarrow 2T$

[2, 6, 5, 8, 11] (N)target = 142

at 2 search for 12 (14 - 2)

TC $\rightarrow O(N \times \log N)$

seq $\rightarrow O(N)$



Two Pointers

Full Functionality lib()

1 pointer

(multimap)

Sort the array

[q = start, s = 0, i, j, b] = O(N^2)

j = end

Sum $\geq K$ =

j-- =

Sum $\leq K$ =

i++

(i - cool)

O

(i - bim ... cool)

(i - m ... i + cool)

$$O(N \log N) + O(N) \text{ bim} = T_{CC} \text{ val}$$

$$O(q) \uparrow \uparrow \uparrow \uparrow \uparrow = S_{CC} \uparrow \uparrow \uparrow \uparrow \uparrow$$

{ { } } { } { } { } { } { } { } { } { } { }

Sort Booleans, 0's

cool

arr[] = [0, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1]

[0, 0, 0, 1, 0, 1, 0, 1, 1, 0] = O(N^2)

\Rightarrow Brute force

Merge sort

b.m

$T_{CC} \rightarrow O(N \log N)$

S_{CC} : $i \rightarrow j$ O(N) val \rightarrow 23ms

$i + bim$ bim \rightarrow 23ms

\Rightarrow Taking i-Count

val \rightarrow 23ms

Cnt0 = 5

Cnt1 = 4

Cnt2 = 3

Set According

$T_{CC} \rightarrow O(N) + O(N) = O(2N)$

$S_{CC} \rightarrow O(1)$

Optimal

(Dutch National Flag Algorithm)

$arr[] = [0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0]$

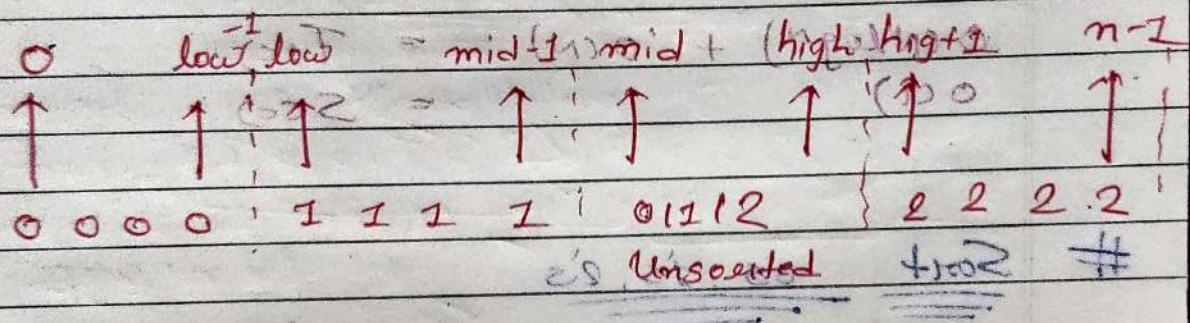
$low = 0$

$[0 \dots low-1] \Rightarrow 0$

$[low \dots mid-1] = -1$

$[high+1 \dots n-1] \Rightarrow 2$

$++j$



$[1, 0, 0, 0] \rightarrow [1, 0, 0, 0] = [1, 0, 0, 0]$

$arr[] = [0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0]$

mid $+ \text{mid} \quad \geq \text{high}$

(1) 0 \leftarrow (0) T

0 Comes at low (1) 0 \leftarrow (0) T

1 Comes at mid (1) 0 \leftarrow (0) T

2 Comes at high (1) 0 \leftarrow (0) T

2 = 0 T

1 = 1 T

0 = 0 T

1 = 1 T

0 = 0 T

1 = 1 T

0 = 0 T

(1) 0 \leftarrow (0) 0 \leftarrow (0) T
(0) 0 \leftarrow (0) 2

$a[\text{mid}] == 0$

$\text{Score}(\text{a}[low], \text{a}[mid])$

$\text{low}++ ; [d^{\text{left}}] \rightarrow \text{mid}$

$\text{mid}++ ;$

$a[\text{mid}] == 1$

$\text{mid}++ ;$

$a[\text{mid}] == 2$

$[2, 2, 2, 2, \text{Score}(a[\text{mid}], a[\text{high}])] = (\text{mid},$

$i, \text{high} - j, \text{f})$

$T(C) \rightarrow O(N)$

$O = T(S) \rightarrow O(1) \approx f + f$

$\Rightarrow O(N) \approx f + f$

Majority element

$\Rightarrow \text{Brute force}$

have cnt for all ($\text{cnt} > n/2$)

$O(N^2)$

$f \approx$

$\Rightarrow \text{Better}$

$\Rightarrow \text{Hashing}$

first iteration

Save

Second iteration $> n/2$

$O(N) + O(N) + O(N \log N)$

$O(N) + O(N \log N)$



① Better - 2

Sort nums[bimm] ; $\text{bimm} \leftarrow \text{bimm} / 2$
 return $\text{nums}[m_2] ; \text{bimm} / 2$
 bimm

$O(N \log N)$ Sorting Only

② optimal

Moore's Voting Algorithm

arr[] = [7, 7, 5, 7, 5, 1, 5, 7, 5, 5, 7, 2, 5, 5, 5, 5]

el = 7

Cnt = 0 1 2 1 2 1 0

(0) \leftarrow Cnt
 7 7 5 7 1 \leftarrow Cnt = 0

j = 7

other

3 times trans | 3 times sort #

el = 5

Cnt = 0 1 0

group found = 0
 group count = 1
 group sum = 0
 Cnt = 0

5 7

just 1 8 <

el = 5

Cnt = 0 1 2 1 0 1 2

5 position 7
 min max + 1

el = 5

Cnt = 0 1 2 3 4

min < pro interest
 bimm + 5(1)5 + 5(1)5
 (n * 10)0 + 5(1)5
 (n * 10)0 + 1(8)0



Rearrange array by sign twist

$$\text{arr} [] = [3, 1, -2, -5, 2, -4] \quad | \quad \downarrow$$

$$\text{ans} [] = [3, -2, 1, -5, 2, -4] \quad | \quad \begin{matrix} S & E \\ S & E \\ S & E \end{matrix}$$

o Brute force

make two array $\begin{matrix} S & 1 & S \\ E & 1 & E \end{matrix}$

pos \rightarrow save Positive $\begin{matrix} S & 1 & S \\ E & 1 & E \end{matrix}$

neg \rightarrow save Negative $\begin{matrix} S & 1 & S \\ E & 1 & E \end{matrix}$

not required to do \rightarrow unnecessary

for $i=0 \rightarrow n_1$) \rightarrow inner loop \rightarrow inner loop

\rightarrow note $\text{arr}[2 \times i] = \text{pos}[i]$

$\text{arr}[2 \times i + 1] = \text{neg}[i] \times ! \text{HCO}$

$$T(C) = O(N) + O(N) + \boxed{2} = O(N)$$

$$S(C) = O(N)$$

o Better twist \rightarrow inner loop \rightarrow inner loop

$$[3, 1, -2, -5, 2, -4] \rightarrow$$

\uparrow

\rightarrow also \rightarrow in primitive E.S

$$\begin{bmatrix} i & [j \geq 1] & 0 & 0 & \in S \end{bmatrix}$$

\leftarrow

\Rightarrow if pos $\text{ans}[i] = \text{num}[i]$
 $i += 2$

if Neg $\text{ans}[j] = \text{num}[i]$
 $j += 2$



Next Permutation

Given $C = [3, 1, 2]$, $S = [1, 2]$ = (1) MP

1 2 3, 2 1, 1 2, 2 = (1) MP

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

3 1 2 → first sort

3 2 1 → sort. ← 209

3 1 2 → sort. ← 109

⇒ make all Permutation

Linear Search ($S^m \leftarrow i$) = Brute

$[1]^{209} = [1 \times S]^{\text{MP}}$ force

$O(N! \times N)$ time = $[1 \times S]^{\text{MP}}$

ans $C = [2, 1, 5, 4, 3, 0, 0, 1, 0, 2] = (0) T$
 $(N!) = (0) 2$

Take just greater than 1

figure out break point right 0

0 ←
Smaller → greater, 1, 2, 3

2 3 remaining in sorted order

E1

$[2, 3, 0, 0, 1, 4, 5] \quad j$

>1

$(1)_{\text{209}} = (1)_{\text{209}}$ 209 ←
 $S = +j$

$(1)_{\text{209}} = (1)_{\text{209}}$ 209 ←
 $S = +i$



for ($i = n-2 \rightarrow 0$) {

if ($c[i] < c[i+1]$) { swap $c[i]$ and $c[i+1]$

+ type no $ind = i$ and $c[i] \leftarrow$ $c[0 \dots i-1]$

break

}

if ($c[i] = -1$)

Reverse;

[2 3 | 5 4 1 0 0]

[2 1 | 5 4 1 0 0]

reverse.

for ($i = n-1 \rightarrow ind$) {

if ($c[i] \geq c[ind]$) { swap $c[i]$ and $c[ind]$

swap ($c[i]$, $c[ind]$) $\leftarrow ind$

Break; reverse :-

y

position \rightarrow

reverse ($c[i : ind+1, n-1]$);

THE POSITION = $i \rightarrow m$

$\rightarrow Tcc$

$\exists (0 \leftarrow \rightarrow m) Scc \rightarrow oof$

$O(N) + O(N) + O(N)$

$\approx O(3N)$

$(C(i), index) \rightarrow m = index$

Leaders in An array

\Rightarrow Everything in right should be
Smaller

:- Leader



$\exists (0 < s-m-1) \text{ not}$

→ Brute force $\exists (c[i:m] > c[m:j])$

Take one check in all on right
linear search word

$c[0:i+2] \leq c[i]$

$\approx O(N^2)$

~~$c[i:m-1] \leq c[m:j]$~~

Brute force

optimal

$\exists (bri < l-m-1) \text{ not}$

at right char find maximum \exists

if $a[i] > \max(l:m)$ now

- leader inserted

Back iteration

$\therefore [l-m, l+bri]$ segment

maxi = INT-MIN;

for ($i \geq m-1 \rightarrow 0$) {

(if $a[i] > \max$) $\exists + (m:0 + (n:0$

ans.push_back($a[i]$)); $(n:0) \approx$

y

maxi = max (maxi, $a[i]$);

y

l m bri maxi ans #

for blonde taking mi printing \exists

2nd last

just \exists

# longest Consecutive Sequence

arr[C] = [102, 4, 100, 1, 101, 3, 2, 1, 1]

[1, 2, 3, 4] 3 (4 - 0 = 4 ans)

[100, 101, 102, 1] = = [3] + 1

L = 4 (4)

→ for every element i from 0 to n-1

Search x+1

Cnt = 0 (initial value + 1) = 1 (1)
if Count increase L = L + 1 } Brute force
find x+2 (initial value + 2)

= O(N²) (initial value + 2) * n = O(n²)

for (i = 0 ; i < N ; i++) {

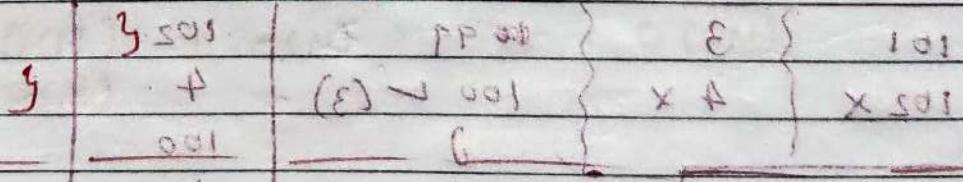
x = arr[i]

Cnt = 1

while (lscare1, x+1) == true) {

x = x+1 ; mi print(" ") ; L = L + 1 }

Cnt = Cnt + 1 ;



Sout it

[1, 1, 1, 2, 2, 2, 3, 3, 4, 100, 100, 101, 101, 102]



Soul Q; ~~max~~ ~~min~~

longest = 1

last - smallest = INT-MIN

(Integers = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) = []

$O(N \log N)$

for $i=0 \rightarrow n$

if $\text{current}[i] == (\text{last} - \text{smallest})$

Cnt = 1

last - smallest = current[i]

else if $\text{current}[i] != \text{last} - \text{smallest}$

then Cnt = 1

last - smallest = current[i]

longest = max(longest, Cnt)

(${}^n \log n$)

Optimal

$\Rightarrow \text{output} = (I + X, MEX) \text{ video}$

Put everything in set = ∞

$(I + X, MEX) = f(n)$

101	{ 3 }	{ 99 }	102.
102 X	{ 4 X }	{ 100 } $\leftarrow (3)$	4.
0		2	100
1 $\leftarrow (4)$		101	1
2		102	101
3			3) i 1002
4			2
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			
72			
73			
74			
75			
76			
77			
78			
79			
80			
81			
82			
83			
84			
85			
86			
87			
88			
89			
90			
91			
92			
93			
94			
95			
96			
97			
98			
99			
100			



$n = 0$

return 0;

longest = 1

unordered_set<int> st;

for (i=0 → n)

st.insert(c[i]);

for (auto it : st) {

if (st.find(i-1) == st.end()) {

int cont = i;

int x = it;

while (st.find(x+1) != st.end()) {

x = x + 1;

cont = cont + 1;

}

longest = max (cont, longest)

}

}

return longest;

$T(c) \approx O(N) + O(N) = O(2N)$

$S(c) \approx O(N)$



Rotate matrix by 90 degrees

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{array}
 \rightarrow
 \begin{array}{cccc} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{array}$$

$$\begin{array}{lll}
 i \ j & j \ n-j-i & i \ j \\
 [0][0] \rightarrow [0][3] & [1][0] \rightarrow [0][2] \\
 [0][1] \rightarrow [0][3] & [1][1] \rightarrow [1][1] \\
 [0][2] \rightarrow [2][3] & [1][2] \rightarrow [2][2] \\
 [0][3] \rightarrow [3][3] & [1][3] \rightarrow [3][2]
 \end{array}$$

$$\underline{\underline{[i][j]}} \rightarrow \underline{\underline{[j][n-j-i]}}$$

Transpose matrix

$$\begin{array}{cccc} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{array}
 \rightarrow \text{Rev all Rows}$$



$$\begin{array}{lll}
 [0][1] \rightarrow [1][0] & 0 \rightarrow (1 \text{ to } 3) \\
 [0][2] \rightarrow [2][0] & 1 \rightarrow (2 \text{ to } 3) \\
 [0][3] \rightarrow [3][0] & i \rightarrow (i+1 \text{ to } n-1) \\
 [1][2] \rightarrow [2][1]
 \end{array}$$



~~for (i = 0, \rightarrow n-1) {
 for (j = i+1, \rightarrow n) {
 Scoop. \langle a[i][j], a[i][i] \rangle ;
 }~~

~~[8 - 5, 19 - 1, 11, 2 - 8, 1] = (2)~~
~~i = 4 }~~

Point Spiral matrice $(1, 2, 3, 4)$
 $= (1, 1, 1, 2, 3)$

	0	1	2	3	
left	1	2	3	4	right
0 top	1	2	3	4	
1	5	6	7	8	$(2, 2, 1, 1)$
2	9	10	11	12	
3	13	14	15	16	
bottom					

$O(N \times N) \leftrightarrow SCo$
 $O(N \times N) \leftrightarrow TCo$

- $[1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5,$
 $6, 7, 11, 10]$ $(++\text{time})$

$\} (m \leftarrow 0 = i) \text{ next}$
right $\} (m \leftarrow j = i) \text{ next}$
 $top = 0$ $left = 0 \text{ init}$
 $bottom = 3$ $\} (i < right) = 3$

$\} (i < right) = 3$
 $\} (i < bottom) = 3$
 $\} (i < left) = 3$

$top++;$

$for (i = top \rightarrow bottom)$
 $a[i][right]$

$right--;$

$for (i = right \rightarrow left)$
 $a[bottom][i]$

$bottom++;$

$for (i = bottom \rightarrow top)$

$a[i][left]$

$left++;$



Subarray & Sum equals K

$$\text{Sum} \leftarrow \text{Sum} + \text{arr}[i]$$

$$\text{Sum} \leftarrow \text{Sum} + \text{arr}[i] = 1 + 1 = 2$$

$$\therefore (\text{Sum}, \text{Index}) = (2, 1)$$

$$\text{arr}[C] = [1, 2, 3, -3, 1, 1, 1, 4, 2, -3] \quad | \quad K=3$$

[1, 2]

[1, 2, 3, -3]

[3]

[2, 3, -3, 1] returns 1 index

[3, -3, 1, 1, 1]

[1, 1, 1]

$\Rightarrow [4, (4x+3)]$

$\Rightarrow [-3, 1, 1, 1, 4, 2, -3]$

8 Subarray

Sum = 0
+ 1 2 3 4 5 6 7 8
S1 S2 S3 S4 S5 S6 S7 S8

Brute force

S1 S2 S3 S4 S5 S6 S7 S8

\Rightarrow generate all subarray

? . P Sym. = K, S1, S2, S3, S4, S5, S6, S7, S8 =

Count++ ; [01, 11, F, 2]

for (i=0 → n) {

for (j=i → n) {

Sum = 0, j

Sum = 0, i

for (k=j → i) {

Sum = mottad

Sum = Sum + arr[k]

0 = 70t

if (Sum = K) {

i ← 110t = j, mottad

Count++

(i)(j)(n)

}

j

j

(mottad ← 70t = j, mottad)

(i)(j)(n)

$\approx O(n^3)$ = 12 not

(+70t) ~ 12 not

(12 not)

(12 not)

(12 not)

(-12 not)

(-12 not)

(-12 not)



Q Optimal

Prefix sum

Prefix Sum

$$\text{arr} [] = [1, 2, 3, -3, 1, 1, 1, 4, 2, -3] \quad k=3$$

Prefix = 0 1 8 6 3 4 8 8 1 8 1 2 9

Cnt = 0

Index	Value	cnt	pair
1	1	0	(1, 1)
2	2	1	(1, 2)
3	3	2	(1, 3)
4	-3	0	(3, -3)
5	1	1	(4, 1)
6	1	2	(5, 1)
7	1	3	(6, 2)
8	4	4	(3, 2)
9	2	5	(7, 1)
10	3	6	(0, 1)
11	-3	7	
12	1	8	
13	2	9	
14	3	10	

$$\frac{s-k}{2} \times \frac{k}{2} \times p \times o! = e^{O(1)}$$

Prefix (sum; s > i ; o = j) not

Q prefixSum (int, int) > mpp; (j-m) ≥ 0 $\Rightarrow 2^m$

cnt = 0, sum = 0; $2^m = 2^m$

for (i = 0 $\rightarrow n$) $\{$ $i+j$

sum += mpp[i]; $\}$

remove = sum - k;

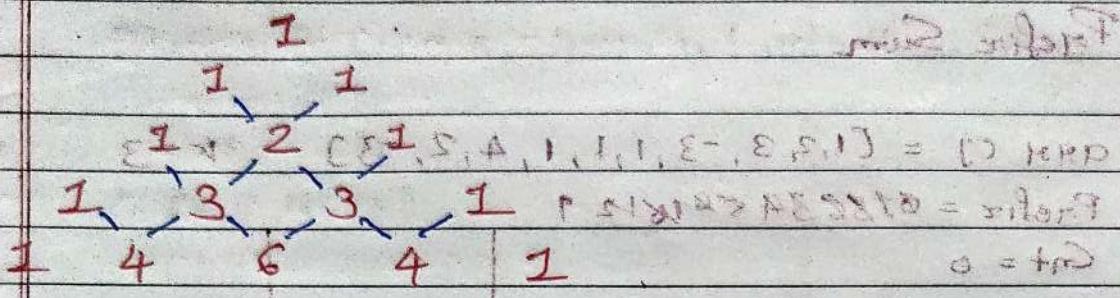
Cnt += mpp[remove];

mpp[sum]++;

return cnt;

# Pascal Triangle

1000470



\Rightarrow Given $r = \text{row no.}$ $\rightarrow S = E - 1$
 $c = \text{col no.}$ $\rightarrow F = r - m$ $\rightarrow O = E - c$

give element at $\frac{r!}{c!(r-c)!} \rightarrow E(r-c) = \frac{m!}{(r-c)(m-c)!}$

ans = $\frac{r-1}{c-1} \frac{(r-1)!}{(c-1)!} \rightarrow S = E - 2$
 $\frac{(r-1)!}{(c-1)!} \rightarrow F = E - 1$
 $\frac{(r-1)!}{(c-1)!} \rightarrow O = E - 3$
 $\times S = E - 2$
 $\times F = E - 1$

${}^r C_2 = \frac{7!}{2! \times 5!} = \frac{7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1}{(2 \times 1) \times 5 \times 4 \times 3 \times 2 \times 1}$

${}^{10} C_3 = \frac{10 \times 9 \times 8}{3 \times 2 \times 1} \rightarrow$

for ($i=0$; $i < n$; $i++$) \rightarrow $i = j$

$res = res * (n-i)$ \rightarrow $i = m-j$ \rightarrow $f(m, j)$

$res = res / i + 1$ \rightarrow $i = m-j$ \rightarrow $O = +m$

j \rightarrow $(j) \text{ sum} = + m^2$

return res \rightarrow $i = m^2 = \text{sum}$

$(\text{sum}) \text{ sum} = + m^2$

$(+ + (\text{sum})) \text{ sum} = + m^2$

time complexity



→ Print given row

110 true ↪

N^{th} row $\rightarrow N$ elements

for ($c=1 \rightarrow n$) {

Print ($f(n-1, c-1)$) to std::

}

"990. Segmented sorting | Recursion | Competitive Coding"

$O(N \times R)$

6th

II. functions ptiroj[5] #

1 5 10 10 5 1

$1, \frac{5}{3}, \frac{5 \times 4}{1 \times 2}, \frac{5 \times 4 \times 3}{1 \times 2 \times 3}, \frac{5 \times 4 \times 3 \times 2}{1 \times 2 \times 3 \times 4}, \frac{5 \times 4 \times 3 \times 2 \times 1}{1 \times 2 \times 3 \times 4 \times 5}$

So

$\frac{5!}{3!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{2 \times 1}$

ans = ans * (n-i)

cout << ans; // [i, n]

ans = ans / i

for (i = 1 → n)

register ans = ans * (n-i)

ans = ans / i

Print (ans);

y

110 seg sort

(N × R)

$O(N)$



Date _____

Page _____

→ Point all were mind trick

generate states using Second type
for all

$\rightarrow (m \leftarrow l-1) \text{ not}$

Code at $(l-1, l-m)$ true

L

"github/Vinayak501/ePP-DSA/Pascal-triangle-CPP"
(ex100)

htj

Majority element II

at least $\frac{n}{2}$ times

$\text{ans} = [1, 1, 3, 3, 2, 2, 2]$ $n=8$

$$\left\lfloor \frac{n}{3} \right\rfloor = \left\lfloor \frac{8}{3} \right\rfloor = 2$$

$[1, 2]$ appearing more than 2 times

more than 2 times

$(m \leftarrow l=j) \text{ not}$

ans only can (two * one - less Integer)
 $512 \text{ mod } 2 = 1$

Brute force : $O(n^2)$ time

check for all

$O(N \times N)$

L

(n²)



o Better
→ Hashing

(O) ~~Time~~ == ~~O(n)~~ ~~O(1)~~ ~~O(n)~~
~~i * O(n)~~

Number → (Counting == ~~O(n)~~ ~~O(1)~~ ~~O(n)~~
~~O(n * n)~~)

Is.add cohesive

$$\text{Count} \geq \frac{n}{3} - \text{Error}$$

O(n) + O(n)

↑
check while adding, ~~O(n)~~
we can remove this

O(n)

∴ O(n)

Ans.

Space → O(n)

[F, T, S, I, O, H] = O(NH)

o optimal

~~O = C1rep + C2rep + C3rep~~
 $n = i - j = l$

[2, 1, 1, 3, 1, 4, 5, 6] ~~grouped~~ ~~11A~~

Cnt1 = 0 # O1

Cnt2 = 0 # O2

el1 = 2 ~~from start~~ 11B el2 = 2 ~~last~~

Cnt1 = 0, el1 =

Cnt2 = 0, el2 =

for ci = 0 → n-1 { (n ← n+1), j = 1 } ~~last~~

if Cnt1 == 0 (if numsc(i) != el1) {

Cnt1 = 1, el1 = numsc(i)

} else-if Cnt2 == 0 (if numsc(i) != el2) {

Cnt2 = 1, el2 = numsc(i)



else if (el1 == nums[i])

cnt1++;

else if (el2 == nums[i])

cnt2++;

else

cnt1 - $\frac{1}{2}$ = < time

cnt2 - $\frac{1}{2}$

(n0 + n0)

manual check on

el1, el2 both satisfy condition

2nd remove and see

(n0)

(n0)

3Sum

(n0) \leftarrow source

arr[] = [-1, 0, 1, 2, -1, 4]

arr[i] + arr[j] + arr[k] = 0

j! = i! = k

All unique Triplets {2, 1, 1, 2, 1, 1, 2}

Brute Force

$O(N^3) = O(n^3)$

Try to output all elements $i = l, j = m, k = n$

for (i = 0 \rightarrow m)

$i = l, 0 = l + m$

for (j = i+1 \rightarrow n) { $l + m \leftarrow 0 = j$ not

3 (3) for (k = j+1 \rightarrow m) $0 = l + m + j$ if

$l + m + j = 0$, $i = l + m$

$O(N^3)$: (i) 2^{m+n} \Rightarrow $O = 2^{m+n} C(4) = 2^{m+n} C(4)$

(ii) $2^{m+n} = 2^{m+n}$, $i = l + m$

?



⇒

for eliminating $O(N^3)$
(third loop) $\rightarrow (m \leftarrow 0 \text{ to } \text{mid})$

for $i=0 \rightarrow n$ $\rightarrow (m \leftarrow i+j = i+m) \text{ just}$
($\text{if } \text{for } j=i+1 \rightarrow m$) $\rightarrow \text{just}$

find $\rightarrow \text{ans} = (\text{ans} + \text{arr}[i] + \text{arr}[i])$;

ans $\leftarrow \text{ans} + \text{arr}[i]$

{ bkt Hashing $\rightarrow \text{ans} = \text{ans} + 3 =$

$(0 \text{ to } 9) + (1 \text{ to } 9) + (2 \text{ to } 9) + \dots + (n-1 \text{ to } 9)$

$[-1, 0, 1, 2, \dots, n-4] \text{ in } \rightarrow$

$\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow$

$\rightarrow [(0 \text{ to } 9) + (1 \text{ to } 9) + \dots + (n-1 \text{ to } 9)]$

$- (-1+0)$

$= +1 \times$

$j + t \rightarrow 2 \text{ to } 9 \rightarrow \text{ans} = \text{ans} + 2 \text{ to } 9$

store j

2
0
 \uparrow

(-1, 1, 0)

(-1, 2, 1)

$= - (-1+1)$

$= 0 \checkmark$

(up to n) 0

$= - (-1+2)$

$= - (1)$

$= -1 \times$

$= - (-1-1)$

$= - (-2)$

$= 2 \checkmark$

⇒

After j make set empty
move i

move j from $i+1$

repeat this till i reaches end



Set <vector<int>> S; // priliminary test
for (int i = 0 → m) { (good birth)

first ←

hashset<int> hashset;

for (int j = i + 1 → m) { (m - o = j) test

thins = - (chums[i] + mums[i])

(if (hashset.find(thins)) {

vector<int> temp

= {temp[i], temp[j], third}

Sort (temp.begin(), temp.end())

st.insert({temp[0], 1});

g

i ← i ← i ↑

hashset.insert(chums[i]);

(0 + 1) -

x ← x =

vector<vector<int>> ans(st);

return ans; i wrote

(0, 1, -)

(1, 2, -)

(1 + 1) - =

→ TLE in Leetcode

✓ 0 =

O(N² * log N)

(1 + 1) - =

(s + 1) - =

(s -) - =

(1) - =

✓ s -

x 1 - =

ans.push_back({j, sum});

i + j most i sum

long coordinate i hit with positive



(C) ~~Optimal~~ min \rightarrow max slides
~~min~~

(C) ~~min = 0~~ min \rightarrow max slides

~~ans[i] = max(Sort());~~

[-2 -2 -2 -1 -1 -1 0 0 0 2 2 2 2]

$\uparrow \uparrow \uparrow$
i j j k j k k

(num) \leq (num)

-2 + 2 - 2 = -2 + + j

-2 + 2 - 1 = -1 + + j

-2 + 2 + 0 = 0 ✓ j++ k++

curr = #

move i, j, k such it's not
Same ele again

0 = 0 \rightarrow sum [s, s-, 0, 1-, 0, 1] = [] RRD

E -2 -2 -2 -1 -1 -1 0 0 0 2 2 2 2]

$\uparrow \uparrow$ [s, 1, 1-, 1] ↑
i j [s, 0, 0, 1] k

[1, 0, 0, 1]

for (int i = 0 \rightarrow n) {

if (i > 0 $\&$ numsc[i] == minst(i, i))

continue;

CST

(* j = 0 + 1, k = n - 1

+ temp = 0 while (k > i) { temp = temp + numsc[j]; }

: (C) sum =

if (sum < 0) *

j++; minimum first word

else if (sum > 0)

(* H0) k--; and second seth

else

temp

ans. add (temp)

temp;



while ($i < k$) if $\text{nums}[i] == \text{nums}[i+1]$
 $j++;$

while ($i < k$) if $\text{nums}[i] == \text{nums}[k-1]$
 $i++;$ $s = s - \text{nums}[i] = \text{O}(N)$

g

$[s] [s] [s]$
 $i \quad i \quad i$

$O(N \log N) \times \approx O(N^2)$

$$i++ \quad s- = s - s + s -$$

$$i++ \quad i- = i - s + s -$$

$$i++ \quad i+i \quad \checkmark \quad 0 = 0 + s + s -$$

4 Sum

~~one sum~~ two sum $\rightarrow i, i, i$ three sum

$\text{arr}[] = [1, 0, -1, 0, -2, 2]$ target = 0

$[s] [s] [s]$

$\rightarrow [2, -1, 1, 2] \quad \begin{matrix} \uparrow & \uparrow \\ i & j \end{matrix}$

$-2, 0, 0, 2$

$-1, 0, 0, 1$

$3 (m \leftarrow 0 = i + i) \text{ not}$

$(i, j, k) \in L \Rightarrow (i \leq m \leq j) \quad 0 \leq k \leq i$

four loop

$L = \pi = \pi_1, 1 + O(N^4) \quad T(c)$

$\text{nums}[i] = \text{target} - (\text{nums}[i] + \text{nums}[j] + \text{nums}[k])$

$\Rightarrow (0 \leq m \leq 2) + i$

look for $\text{nums}[i] \quad i++$

$0 \leq m \leq 2 \quad i \quad \text{seen}$

use hashset like $\{\dots\} \quad O(N^3)$

Before $\{\dots\}$

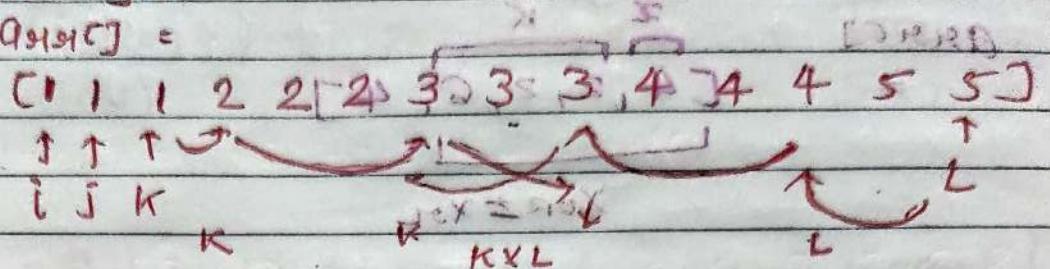
After

$\{\text{target} - \text{nums}[i]\}$



~~#target = 8~~

~~ans[] =~~



$$\Delta X = \Delta^1 X$$

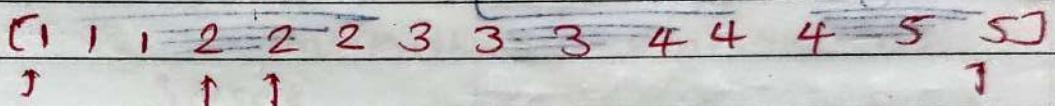
$$1+1+1+5 = 8 \quad \checkmark \quad \Delta^n X = K + \Delta^{n-1} (L - X)$$

$$1+1+2+4 = 8 \quad \checkmark \quad \Delta^{n-1} X = L - X$$

$$1+1+3+3 = 8 \quad \checkmark \quad K++ \quad L--$$

move j when K cross l

~~2nd approach~~ ~~prerequisite~~ ~~spouse~~



~~(81, 21), (21, 21), (4, 8), (K, 8), (11, 8), (P, 8) (2, 1) (E, 1)~~

Same Process

~~code~~

~~81 F "github.com/vinayak801/4sum.cpp" : S |~~

~~(81, 21) (11, 8) : (2, 1)~~

~~# Subarray with XOR K~~

~~⇒ Brute force~~

~~all Subarray
XOR == K~~

~~(1) Time O(n^2)~~

~~(81, 21) (81, 21) (81, 8) (11, 8) (P, 8) (2, 5) (P, 5) (E, 1)~~

~~O(n^2)~~

~~(11, 8)~~

~~(2, 5)~~

~~⇒ Same as Sum K~~

~~(11, 8)~~

~~(2, 5)~~



Date _____ / _____ / _____

Page _____

 $\Rightarrow \text{Ans}$

$$\text{ans} = + \underbrace{[4, 2, 2, 6, 4]}_X \underbrace{[2, 5, 1, 1, 1]}_K : \text{Ans}$$

$X \oplus X = 0$

$X^n K = X R$ $(X^n K)^n K = X R^n K$ $-X = X R^n K$ $- - - + + +$	$\Rightarrow \text{Ans} = 2 + (+ + +)$ $\Rightarrow \text{Ans} = A + S + H + I$ $\Rightarrow \text{Ans} = E + C + L + T$
---	--

220101 K modes in sorted

Merge overlapping Subproblems

~~220101~~ ~~H P S E S S S 1, 1, 1~~

$(1, 3), (2, 6), (8, 9), (9, 11), (8, 10), (2, 4), (15, 18), (16, 17)$

~~220101~~ ~~Ans~~
~~1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18~~

$(1, 6) \quad (8, 11) \quad (15, 18)$

~~Ans~~ ~~Ans~~ ~~Ans~~ #

Bonitc force

~~Sort()~~

~~Ans~~ ~~Ans~~ ~~Ans~~

$(1, 3), (2, 4), (2, 8), (8, 9), (8, 10), (9, 11), (15, 18), (16, 17)$

\checkmark \checkmark \checkmark
 $(1, 4) \quad (8, 10) \quad (15, 18)$
 $\checkmark \checkmark \checkmark$ $\checkmark \checkmark \checkmark$ $\checkmark \checkmark \checkmark$



C.P. 8 $O(N \log N) + \approx O(2N) \approx 2, \approx T(c)_{\text{worst}} = S(c)$

"github.com/Vinayak501/merge-interval.cpp"

(+1e9) (1e9) ~~long long~~ ~~long long~~ ~~long~~ =

`vector<vector<int>> ans;`

`P. sort(cinterval);`

[4, 2, 8, 15]

`for (int i = 0 → n) {`

`If (ans.empty() ||`

`interval[i][0] > ans.back()[1]) {`

`ans.push_back(cinterval[i]);`

`} else {`

`ans.back()[1] = max[ans.back()[1], interval[i][1]]`

`merge(ans.back(), interval[i]);`

P 8 2 5 0 F 2 E 1
T 1 4 7 1 T T F T

return ans;

$$\Sigma = [2] = 2 = \text{nop}$$

merge two sorted array without

Extra Space

P 8 2 5 0 F 2 E 1
T 1 4 7 1 T T F T

$$\text{ans1} = [1, 3, 5, 7] + \text{ans2} = [0, 2, 4, 8, 9]$$

$$\text{ans3} = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

↓

$$\text{P ans1} = [0, 1, 2, 3] + \text{ans2} = [5, 6, 7, 8, 9]$$

$$\Sigma = 1 \text{nop slide}$$

$$\Sigma = \frac{8}{2} = 4 \text{PC}$$



$$\text{gap} = \lceil \frac{2}{2} \rceil = 1 \quad \text{left} = [1, 3, 5, 7] \quad \text{right} = [0, 2, 6, 8, 9]$$

"गठन लम्बाई स्पेस में एक अल्पिक्य (रोड) दृष्टि"

gap method (shell sort)

$$[1, 3, 5, 7] \rightarrow [1, 3, 5, 7, 9]$$

$$3 (m \leftarrow 0 = i + \text{tri}) \rightarrow 9$$

$$4 \quad 11 \quad 0, 1, 2, 3, 2, 1, 0, 9 \rightarrow 5$$

$$3 (5) \rightarrow 4, 5 < [4+5] \rightarrow 4, 5$$

$$((i) \rightarrow 4, 5) \rightarrow 2 \rightarrow 4, 5, 2, 1, 0$$

$$= [4, 5, 1, 3, 2, 0] \quad 3, 2, 1, 0$$

$$\text{Initial gap} = 5 \quad [5] \rightarrow 4, 5, 2, 1, 0$$

$$[5] \rightarrow 4, 5, 2, 1, 0, [3] \rightarrow 4, 5, 2, 1, 0, 3$$

$$1 \quad 3 \quad 5 \quad 7 \quad 0 \quad 2 \quad 6 \quad 8 \quad 9$$

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

left

right

$$\text{gap} = 5 = \lceil \frac{5}{2} \rceil = 3$$

#

front two pair by two out program

$$1 \quad 3 \quad 5 \quad 7 \quad 0 \quad 2 \quad 6 \quad 8 \quad 9$$

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

left swap right $[4, 2, 1, 0] = [4, 2, 1, 0]$

$$(P \Rightarrow \text{swap } \Rightarrow 1, 0) = [1, 0]$$

$$1 \quad 0 \quad 2 \quad 7 \quad 5, 3, 1, 6, 8, 9$$

$$[5, 3, 1, 6, 8, 9] = [3, 1, 6, 8, 9]$$

$$\text{gap} = \lceil \frac{3}{2} \rceil = 2 \quad \text{while } (\text{gap}! = 1)$$

# Find repeating and missing Number

$$\text{QH4CJ} = [4 \ 3 \ 6 \ 2 \ 1 \ 1] \quad n=6$$

Repeating = 1

missing = 5

$$d = m \quad [1 \ 1 \ 2 \ 3 \ 4 \ 5] = m$$

For every Number

(\rightarrow Iteration + 1) - $\left(\begin{array}{l} (4+5+2+1) \\ 1+2+3+4+5+6 \end{array} \right)$

$t = d + \text{Brute force}$

0 time - missing

force =

1 time - x

+ - =

2 time - repeating

(i) $\left(\begin{array}{l} d = r - x \\ d = r - x \end{array} \right)$

$$TCC = O(N^2)$$

$$(1+2+3+4+5+N) - (1+2+3+4+5+6)$$

\rightarrow Hashing

First iteration

Save into hasharray

Second array

Find repeating

Find missing

$$TCC = O(N) + O(N)$$

$$= O(2N)$$

$$SoC = O(N)$$

(ii)

$$PS = r - x$$

$$PS = r + x$$

P-

$$S = r + x$$



② Optimal Scheduling Problem

$x \rightarrow$ Repeating 1 1 2 2 2 2 = 6 times

$y \rightarrow$ missing

Ex position

\geq position

$$\text{array} = [4, 3, 6, 2, 1, 1] \quad n=6$$

$$(4+3+6+2+1+1) - (1+2+3+4+5+6)$$

redundant pairs removed

$$= 3n - 5$$

position - sum 0

$$= -4$$

x - sum 5

$$[x - y = -4] \quad \text{--- (i)}$$

positions - sum 5

(sum) = cost

$$(4^2 + 3^2 + 6^2 + 2^2 + 1^2 + 1^2) - (1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2)$$

principle

$$= 1^2 - 5^2$$

redundant terms

$$= x^2 - y^2$$

position of other sum

$$= -24$$

position sum

$$x^2 - y^2 = -24$$

position sum

$$(y) + (w) = cost$$

$$(x^2 - y) (x+y) = -24$$

$$x+y = -24$$

$$-4$$

$$(w) = cost$$

$$[x+y = c] \quad \text{--- (ii)}$$



Sum both equ

$$2x = 2$$

$$\boxed{x = 1}$$

$$\boxed{y = 5}$$

Xor Method

$$\text{arr} = [4 \ 3 \ 6 \ 2 \ 1 \ 1]$$

$x \rightarrow$ repeating

$y \rightarrow$ missing

$$= 4^{\wedge} 3^{\wedge} 6^{\wedge} 2^{\wedge} 1^{\wedge} 1^{\wedge} 1^{\wedge} 2^{\wedge} 3^{\wedge} 4^{\wedge} 5^{\wedge} 6$$

$$= 1^{\wedge} 5 = 4$$

$$x^{\wedge} y = 4$$

First 1 from right

$$\begin{array}{r} 2 \ 1 \ 0 \\ 0 \ 0 \ 1 \quad 1 \\ 1 \ 0 \ 1 \quad 5 \\ \hline 1 \ 0 \ 0 \quad 4 \end{array}$$

bitNo = 2

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

Total

xor of
first = 1
Second = 5

# Count Inversion

$a[5, 3, 2, 4, 1] = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ 5 & 3 & 2 & 4 & 1 \end{matrix}$

$i < j$ if $a[i] > a[j]$

$(5, 3)$ $(3, 2)$ $(2, 1)$ $(4, 1)$ }
 $(5, 2)$ $(3, 1)$
 $(5, 4)$
 $(5, 1)$ } 8

Brute force

for $i = 0 \rightarrow n$ {

 for $j = i+1 \rightarrow n$ {

 if $a[i] > a[j]$ {

 cnt++;

}

}

}

$\Theta(N^2)$

O optimal

$[2, 3, 5, 6] = n$

i

l

-

,

$[2, 2, 4, 4, 8] = m$

j

l

←

left > right

$2 > 2$ X

i++

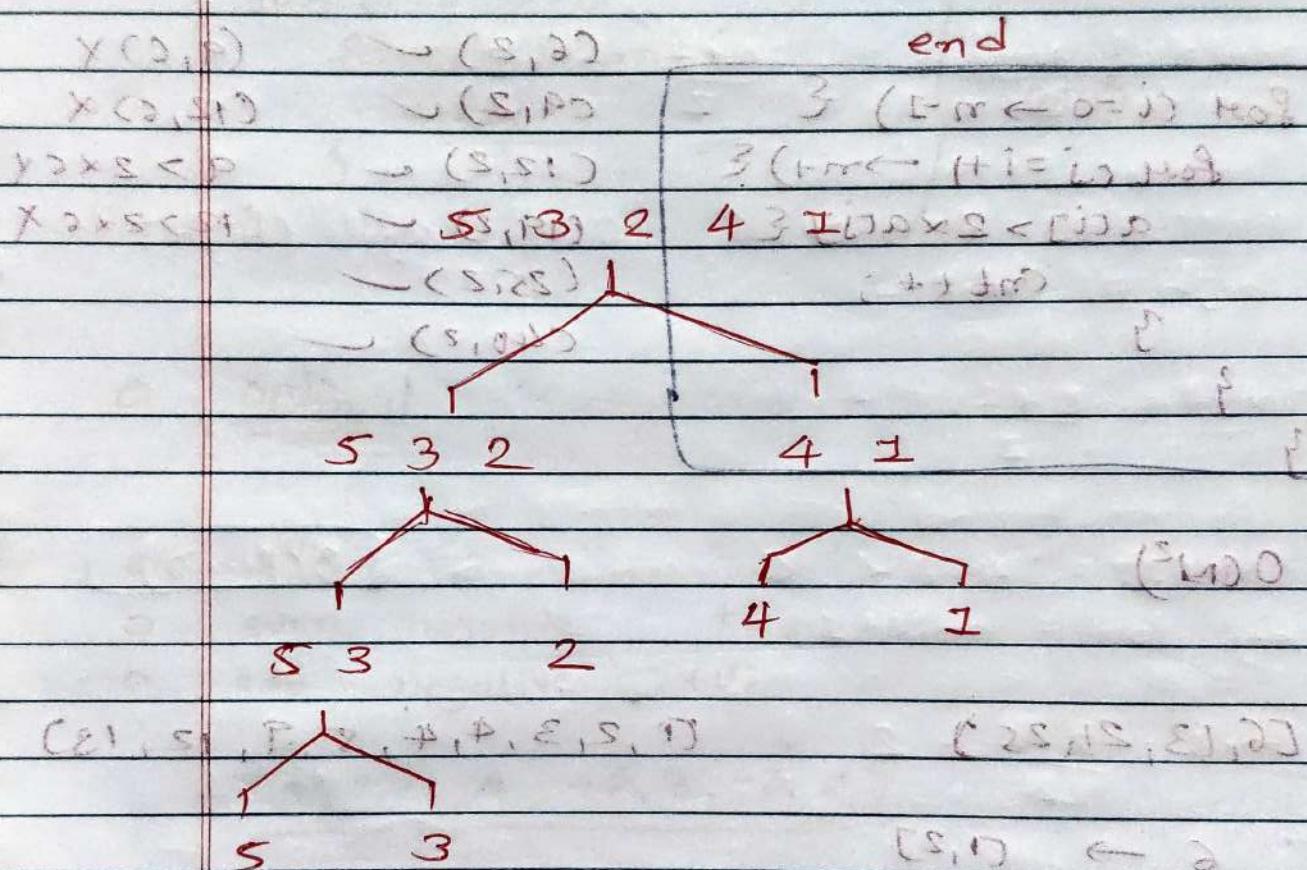
$3 > 2$

✓

j++

cnt + 3;

$3 > 2$	✓	$j++$	$Cnt + 3$	#
$3 > 4$	✗	$i++$		
$5 > 4$	✓	$j++$	$Cnt + 2$	
CS	$5 > 4$ P	✓	$j++$ P	$Cnt + 2$ P
	$5 > 8$	✗	$i++$	
(L)	$5 > 8$ < P	✗	$i++$)	$i = 5$



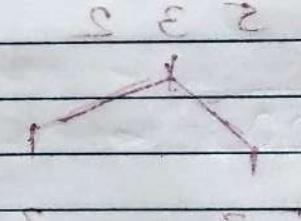
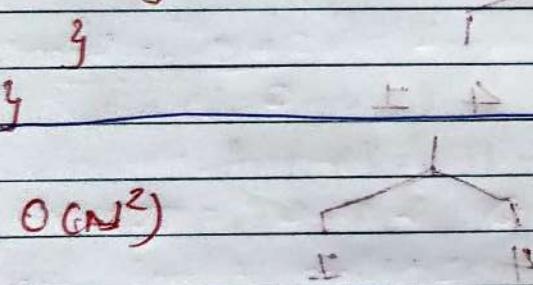
$T(c) \rightarrow O(n \log n)$

Reverse Pair Counting

$\Sigma + \text{for } i = n-1 \rightarrow 0$ $\Sigma + j$ $\Sigma < 0$
 $\Sigma + \text{for } j = i+1 \rightarrow n-1$ $\Sigma + i$ $\Sigma < 0$
 $\Sigma + \text{for } a[i] > 2 \times a[j]$ $\Sigma + j$ $\Sigma < 0$
 $i < j$ $\text{if } a[i] > 2 \times a[j]$ $a[i] > 2 \times a[j]$

 $b \rightarrow$

for $a[i] = 0 \rightarrow n-1$ {
 for $a[j] = i+1 \rightarrow n-1$ {
 $a[i] > 2 \times a[j]$ }
 $\text{cnt}++;$

 $(6, 2) \checkmark$ $(9, 2) \times$ $(9, 2) \checkmark$ $(12, 6) \times$ $(12, 2) \checkmark$ $a > 2 \times c \times$ $\Sigma = 19, 2 \checkmark$ $n > 2 \times c \times$ $(25, 2) \checkmark$ $(40, 2) \checkmark$  $[6, 13, 21, 25]$ $[1, 2, 3, 4, 4, 5, 9, 12, 18]$ $6 \rightarrow [1, 2]$ $\Sigma \quad \Sigma$ $13 \rightarrow [1, 2, 3, 4, 4, 5]$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $21 \rightarrow [1, 2, 3, 4, 4, 5, 9]$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $25 \rightarrow [1, 2, 3, 4, 4, 5, 9, 12]$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $\Sigma + \quad \Sigma < \Sigma \quad \Sigma \quad \Sigma$ $(n^2)O \leftarrow (20T)$

Maximum Product Subarray

$$\text{arr}[] = [2, 3, -2, 4] \quad \text{sol.} \quad \text{P.F.}, 3, +, 2$$

$$\text{ans} = 6 \quad \begin{matrix} & & \\ \text{start} & & \text{end} \\ \downarrow & & \downarrow \end{matrix}$$

for $i=0 \rightarrow n$ {

for $j=i+1 \rightarrow n$ {
 $\max C = \max \{ \text{current value}, \text{current value} * \text{maxC} \}$
 \downarrow

(sol) $\leq n^2$ (slow)

$\downarrow < F = O(n^2)$
 L-bound = slow

O Optimal

$$T = S + O = \text{bim}$$

(for positive) \rightarrow

$$\downarrow > P = O(1) \text{ HWD}$$

- o even negative, $+v_e \downarrow +\text{bim} = \text{sol.}$ fast
- o odd negative, $+v_e$

$$\underline{\underline{[3, 2, 1, 4, -6, 3, -2, 6]}} \quad \frac{S = S + S}{S}$$

Prefix avoid Suffix $\downarrow = \downarrow = O(n)$

o it has zeros (Toprint, dprint, sol, HWD) ?

(sol > dprint HWD) ?

$$\underline{-2 \ 3 \ 4 \ -1 \ 0 \ \underline{-2 \ 3 \ 1 \ 4 \ 0 \ 4 \ 6 \ -1 \ 4}} \quad \frac{S \mid \text{sol} + \text{dprint}}{S}$$

Prefix = $(3 \times -6 \times -24 \times (-24 \times -6)) \text{ HWD} = 1$

bime minimum

* github.com/vinaygakkal/maximum-product-subarray

"Toprint, L-bound, Subarray.cpp"

Binary Search

0 1 2 3 4 5 6 7
 $\{3, 4, 6, 7, 9, 12, 16, 17, 18, 20\}$ target = 6
 ↑ ↑ ↑ ↓ = 20
 low mid high

$$\text{mid} = \frac{\text{low} + \text{high}}{2} = \frac{0+7}{2} = 3.5 \approx 3$$

$$\text{arr}[3] = 7 > 6$$

while ($\text{high} \geq \text{low}$)

$$\text{high} = \text{mid} - 1$$

$$\text{mid} = \frac{0+2}{2} = 1$$

low = 0

$$\text{arr}[1] = 4 < 6$$

$\leftarrow O(\log n)$

$$\text{high} = \text{mid} + 1, \text{if } \text{arr}[\text{mid}] < \text{target}$$

new arr

$$\text{mid} = \frac{2+2}{2} = 2$$

$\underline{\underline{[2, 5, 7, 8, 10, 11, 12, 15, 16]}}$

$$\text{arr}[2] = 6 = 6$$

target found exact

$f(\text{arr}, \text{low}, \text{high}, \text{target})$ {

if ($\text{arr}[\text{high}] < \text{low}$)

return -1; // overflow error

$$\text{mid} = \text{high} + \text{low} / 2;$$

if ($\text{arr}[\text{mid}] == \text{target}$)

return mid;

if ($\text{arr}[\text{mid}] > \text{target}$)

$f(\text{arr}, \text{low}, \text{mid}-1, \text{target});$

else

$f(\text{arr}, \text{mid}+1, \text{high}, \text{target});$

lower Bound

→ Smallest index, "i", such that

$\text{arr}[i] \geq \text{target}$ [2, 8] = arr

[3, 5, 8, 15, 19]

~~x=5~~

ans = 2

(mid)

x = 9

ans = 3

x = 16

ans = 4

x = 20

ans = 5

[0 1 2 3 4 5 6 7 8 9]
 [1 2 3 4 5 6 7 8 9 10]
 low < mid < high

ans = 10 mid = $\frac{0+9}{2} = 4$, $4 \geq 1, 4, 5$

[4 10]
 [1 10] high = mid - 1

0 = x root

mid = $\frac{0+3}{2} = 1$, $1 \geq 1, 0$
 [2 2 3 3 3 3 4 5]

high = mid - 1

(1) 11 = cond
 01 = c

mid = $\frac{0+10}{2} = 0$ (H), d (1 ≥ 1) = c

high = mid + 1 11 m = dd if
 11 = 11 m = dd

lower bound = 0

point 0

First last occurrence of X

$$\text{Ques} = [2, 4, 6, 8, 8, 8, 11, 13] \text{ arr} \quad \text{target } X = 8$$

$$\text{ans} = \{3, 5\} \text{ to print} = [3, 5] \text{ ans}$$

O Brute Force

$$[21, 21, 8, 2, 8]$$

Linear Search

$$S = 2 \text{ MB}$$

$$O(N)$$

$$E = 2 \text{ MB}$$

$$P = S$$

$$D = 2 \text{ MB}$$

$$21 = 5$$

$$Z = 2 \text{ MB}$$

$$05 = 5$$

O Optimal

$I = \infty$, lowest-bound $\Rightarrow 8 \text{ actind}] \geq \infty \quad 1]$
 upper-bound $\rightarrow \text{actind}] \geq \infty \quad 1]$

$$[2, 4, 6, 8, 8, 8, 11, 13] \quad \text{lb} = \text{bim} \quad O(L) = 2 \text{ MB}$$

\downarrow \uparrow \leftarrow

$$\text{lb} \quad \text{up} \quad I - \text{bim} = \text{dpt} \quad \left\{ \begin{array}{l} \text{lb} \\ \text{ub} - 1 \end{array} \right\}$$

for $x = 8$

$$(2, 4, 6, 8, 8, 8, 11, 13) \quad I - \text{bim} = \text{dpt}$$

$$x = 10 \quad \text{lb}(10) = 11 \quad (6)$$

$$x = 14 \leq \text{lb}(14) = 11 + 6 = 17 \quad \text{bim}$$

if $\text{lb} == n$ 11 $\text{ans}(\text{lb}) = 11 \quad \text{return } \{1, 1\}$

O Optimal

D = bimod normal

\Rightarrow Pure Binary Search

Search in Rotated Sorted array

0 1 2 3 4 5 6 7 8
 $\text{arr}[] = [7 8 9 | 1 2 3 4 5 6]$ $\text{lo} = 1$
 low mid high = 8

Identify Sorted (Half) & Unsorted

→ $\text{arr}[\text{low}] < \text{arr}[\text{mid}]$ $\text{low} = \text{high}$
 ↓ ↓

Should be but isn't

left unsorted

right sorted

1. $[\text{low}, \text{mid}] \rightarrow [\text{mid}, \text{high}]$?

target 1 $\in [2, 6]$ false

↑ & ↑ eliminate

2. $[\text{low}, \text{mid}] \rightarrow$ right side

left = bim

$\text{low} = 0$

mid = 4 7

$\text{high} = \text{mid} + 1 = 4 + 1 = 5$ 1

$$\text{mid} = \frac{0+3}{2} = 1$$

$\text{arr}[\text{mid}] = 8 > \text{arr}[\text{low}] = 7$

1. $[\text{mid}, \text{high}] \rightarrow$ right

left sorted

$1 \in [7, 8]$ false

eliminate left

$\text{arr}[\text{high}] > \text{arr}[\text{mid}]$ should
Unsorted

Search between sorted in array #
 $f(\text{array}, m, \text{target})$

$\{$
 $\text{low} = 0$ $\text{high} = m - 1$
 $\text{while } (\text{low} <= \text{high})$ $\{\text{mid} = \lfloor \frac{\text{low} + \text{high}}{2} \rfloor \}$

$$\text{mid} = \frac{\text{low} + \text{high}}{2}$$

$\text{if } \text{current}[\text{mid}] == \text{target}$

return $\text{mid} + 1$

$\text{if left sorted} \rightarrow$

$\text{if } \text{current}[\text{low}] <= \text{current}[\text{mid}] \{$

~~get rid of~~ $\times \text{if } \text{current}[\text{low}] == \text{target}$

otherwise ~~if~~ FF

~~obviously target <= current[mid]~~ $\{$

$$\text{high} = \text{mid} - 1;$$

$\Rightarrow \text{else}$

$$\text{low} = \text{mid} + 1; \text{mid} = \text{mid} + 1$$

$\text{if right sorted} = \frac{\text{low} + \text{high}}{2} = \text{high}$

$\text{else} \{$

$\text{if } \text{current}[\text{mid}] <= \text{target} \text{ FF}$

$\text{target} <= \text{current}[\text{high}] \{$

$$\text{low} = \text{mid} + 1;$$

else

$$\text{high} = \text{mid} - 1;$$

$\}$

return -1

~~but now all~~

Search in Sorted Rotated Array II

$\text{nums} = [7, 8, 1, 2, 3, 3, 3, 4, 5, 6]$
 target = 1

Same as before But have
 Duplicated elements
 \rightarrow (True, False)

Identifying Sorted half doesn't
 work here

$\text{nums} = [3, 1, 2, 3, 3, 3, 3]$ The = emp

have to solve
 $\text{nums}[low] = \text{nums}[mid] = \text{nums}[high]$

Condition

Shrink whenever we get this
 Condition

if ($\text{nums}[low] == \text{nums}[mid]$) if

$\text{nums}[mid] == \text{nums}[high]$)

{

$low++$

$high--$

continue

,

Find minimum Rotated Sorted array

$\text{arr}[] = [4, 5, 6, 7, 0, 1, 2] = \text{ans}$

Linear time complexity.

$O(N)$

$\text{arr}[] = [4, 5, 6, 7, 0, 1, 2]$ (Rotated sorted array)
 $\text{arr}[] = [4, 5, 6, 7, 0, 1, 2]$ (Unsorted array)

Identify sorted half

Sorted may/may not have minimum.
 Unsorted have minimum

$\text{ans} = \text{INT_MIN}, 4, 5, 6, 7, 0, 1, 2 = -100$

In Sorted arr[low : high] minimum is

Find minimum = Find maximum = $\text{arr}[l : h]$

$[4, 5, 6, 7, 0, 1, 2]$
 $l \text{ on } h$ (middle)

$[4, 5, 6, 7, 0, 1, 2]$ (middle)
 h
 l
 m (middle)

$\text{ans} = \text{INT_MAX}, 2, 7, 0, 1$

$[7, 8, 1, 2, 3, 4, 5, 6]$ (middle)
 l
 m
 h (middle)

$[7, 8, 1, 2, 3, 4, 5, 6]$ (middle)

$[7, 8, 1, 2, 3, 4, 5, 6]$
 h
 l
 m

$f(\text{ans}, \text{m})$

{

$\text{low} = 0$

$\text{high} = n - 1$

$\text{ans} = \text{INT-MAX}$

while ($\text{low} <= \text{high}$)

{

$\text{mid} = \text{low} + \text{high} / 2$

if ($\text{arr}[\text{low}] <= \text{arr}[\text{mid}]$) {

$\text{ans} = \min[\text{ans}, \text{arr}[\text{low}]]$

$\text{low} = \text{mid} + 1$

else {

$\text{ans} = \min[\text{ans}, \text{arr}[\text{mid}]]$

$\text{high} = \text{mid} - 1$

}

}

$O(\log n)$

if ($\text{arr}[\text{low}] <= \text{arr}[\text{high}]$)

direct return $\text{arr}[\text{low}]$;

#

Find out How many time

Array been Rotated

$\text{arr}[] = [3, 4, 5, 1, 2]$

$\text{ans} = 3$

Index of minimum element
be ans

0 1 2 3 0 = cool
 $[3, 4, 5, 1, 2]$ L.m = dpid

XMM_rhd = CxD

ans = 3 ($dpid \Rightarrow cool$) slides

3

$\{1, 2, 3\} dpid + cool = bim$

Single element in sorted array

$arr[2] = [1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 6]$

(Chim) even - odd = cool = CxD
 element L.bim = dpid

$arr[i-1] \neq arr[i] \neq arr[i+1]$

return i;

O(N)

(mpo) O

eliminate [1, 2] \rightarrow [cool, bim] fi

0 1 2 3 4 5 6 7 8 9 10
 $[1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 6]$

even, odd \rightarrow left half even, right half $\#$
 Odd, even \rightarrow even right half (ans-right)
 Odd, odd \rightarrow even left half (ans-left)

$\{1, 2, 3, 4\} - \{1, 2\}$

E = 2mD

$f(\text{array}, n)$

{

$n == 1$

 return $\text{array}[0]$;

 if ($\text{array}[0] \neq \text{array}[1]$)

 return $\text{array}[0]$;

 if ($\text{array}[n-1] \neq \text{array}[n-2]$)

 return $\text{array}[n-1]$;

$low = 1$

$high = n - 1$;

 while ($low <= high$) {

$mid = low + high / 2$;

 if ($\text{array}[mid] \neq \text{array}[mid + 1]$ ~~ff~~

$\text{array}[mid] \neq \text{array}[mid - 1]$)

 return $\text{array}[mid]$;

 if ($(mid + 1) * 2 == 1$ ~~ff~~ $\text{array}[mid - 1] ==$
 $\text{array}[mid]$) {

$low = mid + 1$;

 or $mid + 2 == 0$ ~~ff~~

$\text{array}[mid] == \text{array}[mid + 1]$

$low = mid + 1$;

 else

$high = mid - 1$;

 }

 return $\text{array}[mid]$;

$\text{array}[\text{array}] < [1, \dots, n]$;

$[1, \dots, n] < [\text{array}]$;

Peak element

$\text{array} = [1 2 3 4 5 6 7 8 5 1]$

$i \leftarrow 0$ $j \leftarrow n-1$
 $\text{mid} = \lfloor \frac{i+j}{2} \rfloor$ hi

(increasing curve)

$\text{num}[i] < \text{num}[j] > \text{num}[i+1] \dots h$

(decreasing curve)

Assume only 1 Peak is available

$4 < 5 < 6$

$\uparrow \quad \downarrow$ $\text{left} = \text{mid}$ $\text{right} = \text{high}$

$\uparrow \quad \downarrow$ Peak will be here ! $\uparrow \quad \downarrow$ $\text{left} = \text{mid}$ $\text{right} = \text{high}$

Peak can't be in mid

on increasing curve

$= \text{left} = \text{mid}$ $\uparrow \quad \downarrow$ $\text{left} = \text{mid}$

$\uparrow \quad \downarrow$ $\text{left} = \text{mid}$

$f(\text{array}) : l + b * m = \text{val}$

{ $\uparrow \quad \downarrow$ $\text{left} = \text{mid}$ $\uparrow \quad \downarrow$

$\text{if } m == 1 \text{ return } 0;$

$\text{if } \text{array}[m] > \text{array}[l] \text{ return } 0;$

$\text{if } (\text{array}[m] > \text{array}[m-1]) \text{ return } m;$

$low = 1; \quad l - high = \text{dpd}$

$high = n-2;$

while ($low <= high$), { $\text{mid} = \frac{low+high}{2};$

check if peak return mid

else if ($\text{array}[\text{mid}] > \text{array}[\text{mid}-1]$)

$low = \text{mid} + 1$

else if ($\text{array}[\text{mid}] > \text{array}[\text{mid}+1]$)

$high = \text{mid} - 1$



Find Squrt of Integer

$$n = 25$$

$$\sqrt{25} = 5 \quad = 5 \times 5$$

$$n = 36$$

$$\sqrt{36} = 6 \quad = 6 \times 6$$

$$n = 35$$

$$\sqrt{35} = 5 \quad = 5 \times 7$$

$$n = 36$$

$$\sqrt{36} = 6 \quad = 6 \times 6$$

→ Brute force

$$E = 2^{20}$$

$$80 > E \times E$$

→ check for all $1 \leq i \leq n$ or

if In Search

~~$$Z = b \times m$$~~

$$axa > n$$

$$E = 2^{20} \quad (a-1) = \text{ans} \quad S > 2 \times 2$$

```
def sol(c=0 → n) {  
    if (c * c == n) {  
        ans = c;  
    }  
    else  
        break;  
    return ans;
```

$$S = 2^{20}$$

$$80 < 2 \times 2$$

$$E = 2^{20}$$

$$L = 2^{20}$$

$$L - R = 2^{20}$$

→ BS

3 (cool < heat) \rightarrow binary

cool + heat = binary

$$\text{low} = 1$$

$$(m \rightarrow \text{binary}) \quad \text{high} = 2^8$$

$$\text{mid} = 14 \quad \text{bits} = 2^{10}$$

$$14 \times 14 > 2^8$$

go left \rightarrow bits = 10

$$14 \times 14 > 2^8$$

go left \rightarrow bits = 10



low = 1

high = 13

mid = 7

$$7 \times 7 > 28$$

go to R.L.

low = 1

mid = 3

28 < 6

$$3 \times 3 < 28$$

go to right

ans = 3

low = 4

mid = 5

high = 6

$$5 \times 5 < 28$$

(2-ans) ans = 5

low = 6

3 (m - o = 3) high = 6

mid(m = 6) i = 1 j = 7

$$6 \times 6 > 28$$

ans = 5

low = 1

high = n-1

while (high >= low) {

mid = high + low / 2;

if (mid * mid <= n)

ans = mid p = bim

low = mid + 1;

else

high = mid - 1 + 1;

28

I = val

x < + x 11



Find m^{th} root of N

$$N = 3^8 \quad M = 27 \quad \sqrt[3]{27} = 3$$

$$N = 4 \quad m = 69 \quad \sqrt[4]{69} = -1$$

* Implement using only recursion *

→ Brute force soln.

Brute force \Rightarrow linear search

$$\text{low} = 1$$

$$\text{high} = 27$$

$$\text{temp prod} = 1$$

$$\text{mid} = 14 \quad [1, 2, 3, 4]$$

current

/ \ / \ / \ /

$$\text{temp prod} \times 14 > 14 > 27 \text{ not } \rightarrow \text{E Invalid}$$

current go left

current move

$$8 > \text{end} = 21$$

$$\text{low} = 1$$

$$\text{high} = 13$$

$$\text{mid} = 8 \quad \text{not valid}$$

$$8 \times 8 > 27 \quad [1, 2, 3, 4]$$

go left

/ \ / \ / \ /

$$\text{low} = 1 \quad 8 > \text{or} \quad \text{high} = 6$$

$$\text{mid} = 3$$

$$3 \times 3 \times 3 \times 3 = 81 \quad [1, 2, 3, 4]$$

return

/ \ / \ / \ /

8 5 5 5

final ans =

$$T(c) \rightarrow \log_2 n \times \log_2 m$$

# Koko eating Bananas

$$\text{Piles } C = [3, 6, 7, 11] \quad ES = 0 \quad h = 8 \quad E = 4$$

$$E = ES + \sum P_i \quad P_i = h \quad h = 4$$

→ Return the min Integer k such that \rightarrow koko can't eat all bananas, within h hours

$$ES = ES + P_i$$

$$[3, 6, 7, 11] \quad ES = 0 \quad P_i = h = 4$$

$$E = ES$$

$$k = \text{banana}$$

house

$$2 \text{ hours} \mid 3 \quad 4. \quad 6 \text{ hours} < + k \cdot h = E + \text{banana}$$

hour house

+ 8 = 12 or hours

$$= 15 \text{ hours} < 8$$

$$ES = ES + P_i$$

$$E = ES$$

$$\text{Now for } k = 3 \text{ hours}$$

$$[3, 6, 7, 11] \quad ES < -5 \times 3 \times 3 \times 4$$

$$+ 7 = 20 \quad \text{hours}$$

1 2 3 4 — — —

$$= 20 \quad < 8 \quad E = ES$$

$$E = 6 \text{ hours}$$

$$\text{Now for } k = 4 \text{ hours}$$

$$[3, 6, 7, 11] = E \times 2 \times 2 \times 2$$

$$+ 7 = 27 \text{ hours}$$

$$1 2 2 3 — —$$

$$= 8 \text{ hours}$$

$$E = 8 \times 2 \times 2 \times 2 = 64$$

O Brute Force of some sol.

~~Start from 1 and go to last~~

Start from

1 → Increase

where Satisfied

End

1 to maximum banana

(1, F, 2, E)

S = S + 1 + 1

for i=1 → max (files) :

2 reqTime = fun (arr, i) total - OR
fun (arr, hourly) :-

totalTime = 0

2 for (i=0 → n) :

End totalTime = ceil (arr[i]/hourly)

y

return (totalTime,)

3 S = F + E + S + 1

if (reqTime <= h) < < S

return i;

y

$\Theta(\max(\text{files}) \times n)$

(1, F, 2, E)

S = E + S + S + 5

global

2 > 4

F = 250P



here the same for checking
but loop can be replaced by
Binary Search.

1 low 6 mid 2 11 high

check for 6

[3, 6, 7, 11]

$$1 + 1 + 2 + 2 = 6$$

8 (> 6) \rightarrow Possible

go left (mid - 1) \rightarrow ans = 6

1 3 (3 < 6) \rightarrow ans = 6 5

1 low 3 mid 5 high

check for 3

[3, 6, 7, 11]

$$1 + 2 + 3 + 4 = 10$$

10 > 8 (\rightarrow ans = 6)

go right

4

low

4

mid 5 high

check for 4

[3, 6, 7, 11]

$$1 + 2 + 2 + 3 = 8$$

$$8 = 8$$

Possible

4 < 6

$$\text{ans} = 4$$



Min Days to make m bouquets

$$\text{bloomDay} = [7, 7, 7, 7, 11, 11, 12, 7]$$

1st flower $m = 2$
1 bloom on 01 no. of Bouquets
 7^{th} day $+ k = 3$
 $\Sigma t = 2 \text{nd}$ adjust flowers required

at 13th day all bloomed

Possible but we need
 $\Sigma t = 2 \text{nd}$ minimum

O Brute force

check for m -blooms to 1. $0 = 10$

pick 3 (min - possible)

$T(c) \rightarrow O(\text{mini} - \text{maxi} + 1) \times N$

O BS

Checking will be same bit days will be selected upon BS

$T(c) \rightarrow O(m \log n)$
 $(m = O(\log \frac{\text{maxi} - \text{mini} + 1}{1}) \times N)$



if $m^*k > \text{bloomDay.size}();$

return -1;

(F, S, H, H, E, F, F, F) = post mould

low = min(bloom) mid high = max(bloomday)

7 10 13

Not Possible

11 12 13

Possible

ans = 12

11 12 13

bloom not possible

12 11 13

end

bloom ans find

ans = 12

Possible (ans[], day, m, k) {

Cnt = 0; Conform = 0;

for (i=0 → n-1) {

if (ans[i] <= day) {

Cnt++;

} else {

Conform + = Cnt / k;

Cnt = 0;

}

Conform + = Cnt / k; } } }

return (Conform) ≥ = m);

g



bs (arr, k, m)

{

low = min (arr)

high = max (arr)

ans = high

while (high >= low) {

mid = low + high / 2;

if (possible (arr, mid, m, k)) {

ans = mid;

high = mid - 1;

else

low = mid + 1;

}

return ans;

}

Find Smallest divisor given a threshold

Find

div = 4

+ 1 2 5 9
4 4 4 4

+ 1 + 1 + 2 + 3 = 7 <= 6

Not

Possible



$$\text{div} = 5$$

(min, max, mid)

$$\begin{matrix} 1 & 2 & 5 & 9 \end{matrix} \quad 3$$

$\frac{1}{5} \quad \frac{2}{5} \quad \frac{5}{5}$ (mid) min = 1
5 = 1 point
 $1 + 1 + 1 + 2 = 5$ mid = 3
 3 (mid + 2 * dead) slides ✓

Possible slides: 1, 2, 3, 4, 5, 6, 7, 8, 9

3 (min, max, mid, slides) 9

min = 1
max = 9

o Brute force

linear search off Possibility

1 to 9

remove

1 to max (9)

o checking will be same

But on BS

o BS

loc mid high

1 5 9

Possible $2 + 1 = \text{ans} = 3$

1 2 4

Not Possible \rightarrow vib

3 2 1 4

Not Possible \rightarrow vib

4 2 1 1 4

* Not Possible

5 end

ans = 5

f (car, threshold)

卷之三

C 81

$\text{low} = 1$; $\text{high} = 17$; $\text{mid} = 9$

`high = max (ans);`

while ($high \geq low$) {

~~mid = high | low / 2~~

if (Possible (ans1, mid))

$\lambda = \text{threshold} \}$

ans = mid; cout < ans;

~~high = mid - 1~~

else

$$\text{low} = \text{mid} + 1 \geq 700$$

YES, I'm just here.

return (ans);

3

possible (ans1, mid) {

for $i = 0 \rightarrow n-1$) {

Sum += ceil((arr[i] / mid))

3

ягненок сын;

۳

$$(T(c) \rightarrow O(\log^{\max} x N))$$



Date _____ / _____ / _____

Page _____

Least Capacity to Ship Packages within D Days

$$\text{days} = 5$$

weights = {1 2 3 4 5 6 7 8 9 10}

Ship runs once per day
in 5 days Ship Cart Product
Min (Capacity) of ship needed

$$\text{Cap} = 100$$

In one day

$$\text{If Cap} = 10$$

- first day $\rightarrow \{1, 2, 3, 4\}$
- second day $\rightarrow \{5\}$
- third day $\rightarrow \{6\}$
- 4th day $\rightarrow \{7\}$
- 5th day $\rightarrow \{8\}$

{9, 10} remaining

Increase Cap

$$\text{Cap} = 15$$

- 1st $\rightarrow \{1, 2, 3, 4, 5\}$
- 2nd $\rightarrow \{6, 7\}$
- 3rd $\rightarrow \{8\}$
- 4th $\rightarrow \{9\}$
- 5th $\rightarrow \{10\}$

done.

14 Not Possible

ans = 14



min

bms

cool

Capacity = max (weight)

O1

max

Sum of all weight

O1

DaysReq (wt, cap) {

day = 1;

load = 0;

for ci = 0 → n-1 {

if (load + wt[i] > cap) {

day = day + 1;

load = wt[i];

else

load += wt[i];

}

return day;

}

B brute force

```
for (i = min(wt) → sum(wt)) {  
    if (dayReq(wt, i) ≤ day) {  
        return i;
```

(min(wt) ≤ sum(wt))

(sum(wt) ≤ sum(wt))

O((min(wt) + sum(wt)) × N)

(≈ O(N²))

QBs

low	mid	high
10	(high 32)	55
	Possible	ans = 32
10	20	ans 31
	Possible	ans = 20
10	14	Not possible
to	14	o = book
	Not possible	o = ref.
14	3 (90 < 58) 17 books	19
	Possible	ans = 17
14	15 books	16
	Possible	ans = 15
14	14 books	14
	Not possible	
15	end	14

$$\boxed{\text{ans} = 15}$$

$\text{f}(cwt, dys)$

{
 $3((\text{low}) \text{mid} \leftarrow (\text{low} + \text{high}) / 2)$
 $\text{low} \leftarrow \max(\text{cwt}), \text{high} \leftarrow \sum(\text{cwt})$;
 while ($\text{low} \neq \text{high}$) {

$\text{mid} = \text{low} + \text{high} / 2;$

$\text{noofdys} = \text{daysreq}(\text{cwt}, \text{mid});$

 if ($\text{noofdys} \leq \text{dys}$)

$(\text{high} = \text{mid} + 1) \text{ daysreq}()$;

 else

$\text{low} = \text{mid} + 1; \text{daysreq}()$;

y

return low;

# Find k^{th} Missing Number

$$\text{array} = [2, 3, 4, 7, 11] \quad k=5$$

1 2 3 4 5 6 7 8 9 10 11
1 1 1 1 1 1
1 2 3 4 5 6
1 1 1 1 1 1
 $\oplus =$ dead lock ans

O Brute force

$$[2, 3, 4, 7, 11] \quad k=8, 6, 7, 8, 9$$

↓ ↓ ↓ ↓ 1 1 1 1 1

for ($i=0 \rightarrow n+1$) {
 if ($\text{array}[i] \leq k$) $k++$;
 else break; }
 else
 dead lock

O BS

Search space = array

$$[2, 3, 4, 7, 11] \quad k=5$$

1 1 1 3 etc

$$11 - 5 = 6 \text{ missing}$$

$$7 - 4 = 3 \text{ missing} \quad \text{etc}$$



Date _____ / _____ / _____

Page _____

0 1 2 3 4

2 3 4 7 11

1 1 1 3 6

 $\frac{0+4}{2}$ 2 < 11 \uparrow (11 + 7 + 8 + 8) = 12

low mid high

= 2

11 0 1 2 3 4 5 6 7 8 9 10 11

1 < 5 \uparrow \uparrow \uparrow $\frac{8+4}{2}$

end low high

= 3

mid

start 0

3 < 5

0 1 2 3 4 5 6 7 8 9 10 11

low \uparrow \uparrow

high

= 4

mid

6 > 5 \uparrow \uparrow T \uparrow \uparrow \rightarrow nearby indexes
high low we geta[high] \rightarrow 7
8
9

missing = 3 more = 2

= 7 + 2

missing = 9 = 2 - 11

missing = 8 = 4 - 6

# Aggressive Cows

N stalls

$$\text{Arr} = [0, 3, 4, 7, 10, 9] \quad \text{Cows} = 4$$

→ place 4 cows in such way
 min distance b/w two cows
 is maximum

Arr is coordinates of stalls

$$(0, 3, 4, 7, 9, 10)$$

$$9 - c_2 - c_3 - c_4$$

$$\text{min } 3 \text{ unit} \quad (3 \text{ min} < 2 \text{ min}) \text{ min} = 1$$

$$c_1 - c_2 - c_3 - c_4 \quad \text{min} = 1$$

$$c_1 - c_2 - c_3 - c_4 \quad \text{min} = 3$$

$$\text{ans} = 3$$

Sort $\{0, 3, 4, 7, 9, 10\}$ $\rightarrow \{0, 3, 4, 7, 9, 10\}$

0	3	4	7	9	10	dis
9	c_2	c_3	c_4			dis = 1 ✓
9	c_2	c_3	c_4			dis = 2 ✓
9	c_2	c_3	c_4			dis = 3 ✓
9	c_2	c_3	c_4			dis = 4 X

$\approx O(N^2)$

ans = 3



dis
[I, max-min]

(int) N

bool canWePlace (arr, dist, cows)

{

CntCows = 1;

Coord = arr[0];

for (i = 1 → n-1) {

if (arr[i] - Coord) ≥ dist) {

CntCows++;

last = Coord = arr[i];

if (CntCows ≥ cows) return true;

return false;

E = min (P → E) → P → P

O Brute force

for (i = 1; i ≤ max-min; i++) {

if (canWePlace (arr, i, cows)) {

Continue;

else {

L = i; return i+1; }

S = i; }

E = i; }

O (max-min) × N)

($\approx O(N^2)$)

(Time =)

# Allocate Books bim
 2

$$\text{chart} = [25 \ 46 \ 28 \ 49 \ 24]$$

student = 4

Allocate such that maximum no. of pages student get is min

$$25 \ 46 \ 28 \ 49 \ 24 = 73$$

$$25 \ 146 \ 28 \ 49 \ 24 = 77$$

$$25 \ 146 \ 28 \ 149 \ 24 = 74$$

$$25 \ 146 \ 28 \ 149 \ 124 = 71$$

ans

Keep Pages start from max

49

$$1 \rightarrow 25 \quad +46 > 49 \quad X$$

$$2 \rightarrow 46 \quad +28 > 49 \quad X$$

$$3 \rightarrow 28 \quad +49 > 49 \quad X$$

$$4 \rightarrow 49 \quad +24 > 49 \quad X$$

$$5 \rightarrow 24 \quad : (0)P - (5-1)B = dPd$$

$S > \text{student} = 4$

Increase in Pages until it's possible

bim = 2nd

① Brute Force bim = 603

= Start with 1 more (Pages)

keep Increasing until

Sum (Pages)

(n)(n)O



• If possible (Pages = Max \rightarrow sum)

(1) SFE

(2) OII

(3) PP

cntstu = fun (arr, Pages)

(3) if (cntstudent == m) {
 return pages
}

(4) SF

y

(5) OII

(6) PP

y

student[i] = 1

(7) SFE

(8) OII

(9) PP

student[i] = 0

(10) SFE

(11) OII

(12) PP

student[i] = 1

(13) OII

(14) PP

student[i] = 0

(15) OII

(16) PP

student[i] = 1

(17) OII

(18) PP

student[i] = 0

(19) OII

(20) PP

student[i] = 1

(21) OII

(22) PP

student[i] = 0

(23) OII

(24) PP

student[i] = 1

(25) OII

(26) PP

student[i] = 0

(27) OII

(28) PP

student[i] = 1

(29) OII

(30) PP

student[i] = 0

(31) OII

(32) PP

student[i] = 1

(33) OII

(34) PP

student[i] = 0

(35) OII

(36) PP

student[i] = 1

(37) OII

(38) PP

student[i] = 0

(39) OII

(40) PP

$O((sum - max) \times N) = O(N)$

$\approx O(N^2)$

$\rightarrow \text{bin} = \text{dp[1]}$

get

wall creation

(40) O(N)



low	mid	high
49 (5)	110 (2)	172 (1)
49 (8)	78 (3)	log (e)
49 (8)	63 (5)	77 (3)
62 (5)	69 (5)	77 (3)
70 (5)	73 (4)	ans = 73
70 (5)	71 (4)	ans = 72
70	70 (5)	ans = 70
71	Not Possible	70
	end ()	

$$O = \text{constant}$$
$$\boxed{\text{ans} = 71}$$

return low;

while (low <= high) {

 mid = low + high / 2

 nostu = fn (arr, mid)

 if (nostu > m)

 low = mid + 1; // ans = mid) O

 else

 high = mid - 1; // (m) O = 1

}

return low;

$\approx O(N \log N)$



Painters Partition / SPLIT array
largest sum

$$\text{arr}[] = [10 20 30 40] \quad k=2$$

[10] [20 30 40] 90 min Painters

[10 20] [30 40] 70 min

[10 20 30] [40] 60 min

↑

↑

Painter 1 Painter 2

max Subarray Sum + should be min

→ Same as before

Minimize maximum Distance to Gas Station

$$\text{arr}[] = [1 2 3 4 5] \quad k=4$$

place 4 new gas station

$$[1 2 3 4 5 6 7 8 9] \quad 2$$

1 1 1 1 1 1 1

$$[1 \quad 1.25 \quad 2 \quad 2.5 \quad 3 \quad 3.5 \quad 4 \quad 4.5 \quad 5]$$

0.5 0.5 = 0.5



Date _____ / _____ / _____

Page _____

$$a_{9191}(7) = [1 \ 17] \text{ odd index} \quad k=2 \quad \#$$

$$\begin{matrix} 1 & 3 & 5 & 7 \\ \backslash & \backslash & \backslash \\ 2 & 2 & 2 \end{matrix} \quad \text{ans} = 2$$

$$a_{9191}(7) = [1 \ 13 \ 17 \ 23] \quad [08 \ 08 \ 08] \quad [01]$$
$$\begin{matrix} 1 & 13 & 17 & 23 \\ \backslash & \backslash & \backslash \\ 12 & 4 & 5 \end{matrix} \quad k=5$$

$$\begin{matrix} 1 & 7 & 13 & 17 & 23 \\ \backslash & \backslash & \backslash & \backslash \\ 6 & 6 & 4 & 6 \end{matrix} \quad k=4$$

$$\begin{matrix} 1 & 7 & 13 & 17 & 20 & 23 \\ \backslash & \backslash & \backslash & \backslash & \backslash \\ 6 & 6 & 4 & 3 & 3 \end{matrix} \quad k=3$$

$$\begin{matrix} 1 & 5 & 9 & 13 & 17 & 20 & 23 \\ \backslash & \backslash & \backslash & \backslash & \backslash & \backslash \\ 4 & 4 & 4 & 4 & 3 & 3 \end{matrix} \quad k=2 \quad \#$$

20 to 27 Baki

20 24 28 32 36 Skip

$$\begin{matrix} 2 & 2 & 4 & 4 & 8 & 8 & 2 & 2 & 1 & 1 \\ \backslash & \backslash \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 \end{matrix}$$

Two Pointers

= Sliding Window



Date _____ / _____ / _____

Page _____

1. Constant Window

$$O = \Theta(1) \quad O = 1$$

$$O = \Theta(nm)$$

$$\text{arr}[] = [-1, 2, 3, 3, 4, 5, -1] \quad k = 4$$

(n > m) slides

$$l = 0 \quad r = k - 1$$

Shift window to find max sum
of consecutive (4 elements)

$$(l + 1) \text{max} - \text{max} = \text{new}$$

$$\text{Sum} = 7$$

while ($r < m - 1$)

{

$$(r + 1) \text{Sum} = \text{Sum} - \text{arr}[l]$$

l++;

r++;

$$\text{Sum} = \text{Sum} + \text{arr}[r];$$

$$\text{max} = \max(\text{sum}, \text{max});$$

return max;

2. longest Substring/Subarray (Condition)

longest Subarray with sum $s = k$

$$\text{arr}[] = [2, 5, 1, 7, 10]$$

$$k = 24$$

longest Subarray minimum | subarray



$l = 0 \quad r = 0 \quad \text{Sum} = 0$ (initially) \leftarrow
 $\max = 0$

$t = 1 \quad [1 \geq t \leq n-1] = \text{TRUE}$

while ($t < n$)

{

$\text{Sum} = \text{Sum} + \text{arr}[t]$

while ($\text{Sum} > k$) { (if $\text{Sum} > k$) \leftarrow } to

$\text{Sum} = \text{Sum} - \text{arr}[l]$

$l++$;

g

$(t < n > k)$ \leftarrow if $\text{Sum} > k$

if ($\text{Sum} <= k$)

$\max = \max(\max, g - l + 1)$;

$r++$;

g

$(t < n > k) + \max = \max$

$(\text{current max}) \text{ same} = \times 1000$

for optimizing :- if we get max length
4 don't let the window
shifting more than 3
as we only want
ans ≥ 3 \leftarrow original

3. No. of Subarray Where $\langle \text{condition} \rangle$
 $\underline{\underline{P}} \underline{\underline{S}} \underline{\underline{E}}$ $\underline{\underline{Q}} \underline{\underline{F}} \underline{\underline{E}} \underline{\underline{S}} \underline{\underline{T}} = \underline{\underline{C}} \underline{\underline{E}}$

4. Shortest \mid minimum Window $\langle \text{condition} \rangle$
 $\underline{\underline{W}}$ $\underline{\underline{M}}$ $\underline{\underline{W}}$ $\underline{\underline{C}}$

#

max Points you can obtain

$$\text{arr}[] = [6 \ 2 \ 3 \ 4 \ 7 \ 2 \ 1 \ 7 \ 1] \quad k=4$$

Subproblems:

Pick up from start or end

$$[6 \ 2 \ 3]$$

$$1] = 12$$

$$[6 \ 2]$$

$$7 \ 1] = 16 \text{ ans}$$

$$[6 \ 2 \ 3 \ 4]$$

$$] = 15$$

$$\left. \begin{array}{l} lsum = 0 \ 15 \\ lsum = (11 - [(i+k) \text{ head}]) \ rsum = 1 \\ lsum = 8 \ i + j - i = rsum = 8 \\ lsum = 6 \\ lsum = 0 \end{array} \right\} \begin{array}{l} 15 \\ 12 \\ 16 \\ 15 \\ 11 \end{array}$$

f(nums, k)

{ (228) 0

lsum = 0, rsum = 0, mind = n;

for (i=0 → k-1) {

 lsum += nums[i];

}

 maxsum = lsum;

 for (i=k-1 → 0) {

 lsum -= nums[i];

 rsum += nums[mind];

 mind -= 1;

 }

 maxsum = max(maxsum, lsum +

 rsum);

 }



longest Substring without Repeating

$S = cadbzabcd \rightarrow \text{ans} = cadbz$

no hash max S in 2019
Ans

Brute force

Generate all []
and check [] \rightarrow []
[] \rightarrow [] \rightarrow []

for C i=0 \rightarrow m-1 {

hush[256] = {0}

z1 | for ~Cj=0 \rightarrow m-1 { z1 & = m-2 }

z1 | j = m if hush[S[j]] == 1 = break;

z1 | & = m-1 = j - i + 1 & = m-2

z1 | & = maxLen = max(maxLen, len) ;

i | i = m if hush[S[i]] == 1; o = m-2

}

g

(>, 2m+1) ?

$O(N^2)$

$O(256)$

o = write o = m-2r o = m-2l

Optimal { (i \leftarrow o = m-2r)

$S = cadbzabcd$

l m-2r = m-2m \rightarrow m-2m

r i \leftarrow r = 11

len = r - l + 1

max = 0

2, 4

3, 6

4, 2

5, 1

6, 0

a is at 1st index l have to go

l = mppc[S[1]] + 2 + m-2 + m-1

b is at 3rd index l have to go

l = mppc[S[3]] + 2 + m-2 + m-1

2

3

4

5

char L index



fun (String s) { ~~bad~~ bad
 hush[256] = {-1};

 l = 0 r = 0 maxlen = 0

 while (r < n)

{

 if (hush[s[r]] != -1) {

 if (hush[s[r]] >= l) {

~~zeros~~

 l = hush[s[r]] + 1;

 }

 }

 len = r - l + 1;

 maxlen = max(len, maxlen);

 hush[s[r]] = r; ~~length~~

 r++;

 }
 return maxlen + 1; } = O(n)

}

Time = O(n)

O(n)

1 + O(256) = O(1)

1 + 0 - 0 =

1 + 0 - 1 =

Max Consecutive ones III
4

arr[] = [1 1 1 0 0 0 1 1 1 0] k = 2

allowed to make k zeros to ones

[1 1 1 0 0 0 1 1 1 0]

5

6

6



Goal that find longest Subarray
with K zeros = $O(N^2)$ time

$O(n \times m)$ $O = k$ $O = l$

Brute Force ($\approx N^2$) time

Time (for loops) \approx

$3 \times \text{all combination}$ \approx

$4 \times \text{containing } K \text{ zeros}$

$O(N^2)$

~~$(1+2+\dots+n)^2$~~

$(m \times n) \times (m \times n) = m^2 n^2$

O optimal $\approx O(2^k)$ time

$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10$

$arr[0] = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0]$

$\underbrace{\quad}_{l} \quad \underbrace{\quad}_{r} \quad \underbrace{\quad}_{l} \quad \underbrace{\quad}_{r}$

$\text{Zeros} = 81232$

$$\begin{aligned} \text{len} &= (9 - 1 + 1) & \text{maxLen} &\Rightarrow 0 \\ &= 0 - 0 + 1 & &= 1 \\ &= 1 - 0 + 1 & &= 2 \\ &= 2 - 0 + 1 & &= 3 \\ &= 3 - 0 + 1 & &= 4 \\ &= 4 - 0 + 1 & &= 5 \end{aligned}$$

$$c = * 1 0 1 1 1 0 0 0 1 1 1 = 01100$$

$$l++, l++, l++, l++ \quad \text{Zeros} = 2$$

sum of zeros \times sum of lengths

$$= 6 - 4 + 1 = 3$$

$$= 7 - 4 + 1 = 4$$

$$= 8 - 4 + 1 = 5$$

$$= 9 - 4 + 1 = 6$$

$0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1$

fun (nums, k)

maxlen = 0, l = 0, r = 0, zeros = 0

while (r < m)

{
 if (nums[r]) == 0)
 zeros++;

 while (zeros >= k) {

 if (nums[l] == 0)

 zeros--;

 l++;
 left++;

y

 if (zeros < k) {

 len = r - l + 1;

 maxlen = max(len, maxlen);

y

 r++;

y

return maxlen;

y

TCC = O(N+N)

SCC = O(1)

space complexity = O(2N)

→ optimality :- Do not allow window to shrink more than maxlen - 1, if zeros <= k update else not

TCC = O(N)

SCC = O(1)



func (nums, k) { (\star , zeroSum) mid.

 l = 0, r = 0, zeros = 0, maxLen = 0

 while (l < m) { 0 - maxLen

 if (nums[l] == 0) zeros++;

 if (zeros > k) { }

 if (nums[l] == 0) zeros--;

 l++

 } (\star , zeroSum) right

 if (zeros <= k) { }

 len = r - l + 1;

 maxLen update

 }

 r++ (\star , zeroSum) fi

 }

 l + len - 1 = r

 return maxLen;

 }

Fruit into Basket

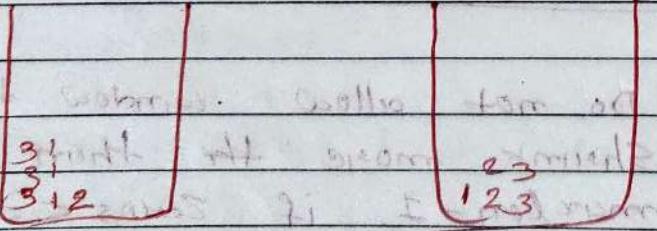
arr[] = [3 3 3 1 2 1 1 2 3 3 4]

(n = 11)

(m + m) = 6

(m) maxLen with

two types of



from 0 to 3 fruits

one Basket

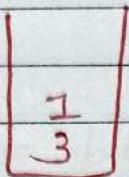
stone only one type

(n = 11)

(m) = 6

$\text{arr}[] = [3, 3, 3, 1, 2, 1, 1, 2, 3, 3, 4]$

Set



Size should be
 $t = k, 2^m$

o Bonite force

Double for loop

$= O(N^2)$

$= O(2)$

$O = O(2^m)$

o Sliding Window

$[3, 3, 3, 1, 2, 1, 1, 2, 3, 3, 4]$

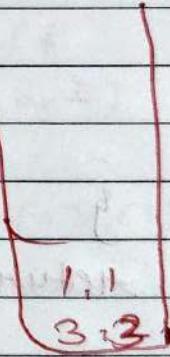
l, r or max = 0 to 4

maxlen = 0 to 4

if any num $\rightarrow 0$

remove

move l



num, lme

(3) 0 -> 02

(4) 0 -> 01
(4) 0 -> 02

sum to look for positions of windows

(2) 0 -> 02

(3) 0 -> 02



[P E S S I S I E E E] = O(N^2)

fun (arr) {

l=0, r=0, maxlen=0

t=2

mPP

while (r < arr.size())

{

mPP[arr[r]]++; sum = 0

if (mPP.size() > k) {

while (mPP.size() > k) {

(l) =

mPP[arr[l]]--;

if (mPP[arr[l]] == 0)

mPP.erase(arr[l]);

l++;

possible

y

y

if (mPP.size() <= k) {

maxlen = max(maxlen,

right - l + 1);

y

right++;

maxlen

return maxlen;

y

return maxlen;

;

Tcc) $\rightarrow O(N+N)$
 $= O(2N)$

Scc) = $O(L)$

\Rightarrow optimize by sticking window at max

Tcc) $\rightarrow O(N)$

Scc) $\rightarrow O(3)$

Number of Substring Containing All three char

 $S = *$ $A A A A A A A = 2$ $S = bbaabca$ $ans = 9$ $B = 200$ $A A A A A A A =$

Brute force
 two loops
 time: $O(N^2)$


 $\text{fun}(S) \{$ $\}$ $\} (m \leftarrow 0 = 3) \text{ not}$ $\text{lastSeen}(3) = \{ -1, 1, 1, 1 \};$ $\text{cnt} = 0;$ $\text{for } (i=0 \rightarrow n) \{ = 1 \text{ not}$ $\{ i : 4 + ['0' - \text{lastSeen}[0]] \text{ and }$ $\{ '0' - \text{lastSeen}[0] \text{ and } (\text{lastSeen}[S[i] - 'a'] = i)$ $\text{if } (\text{lastSeen}[0] != -1 \text{ iff}$ $(\Rightarrow \text{all are Present})$ $(2) \text{, return } \text{cnt} = \text{cnt} + (1 +$ $\min(\text{lastSeen}[0], [1], [2]))$ $\}$
 return cnt; $\}$ $\text{maximum complexity}$ $(as) O$ $(as) O$



Date _____ / _____ / _____



Date _____ / _____ / _____

Page _____

longest Repeating Character replacement

$$S = AABABA$$

$$P = 2 \text{ ms}$$

$$= \underbrace{AAAAAA}_{\text{P}} BA$$

$$K = 2$$

$$\text{P} = \text{P} \text{ per period}$$

$$\text{ans} = 5$$

$$A \rightarrow 3$$

$$B \rightarrow 2$$

$$\rightarrow$$

Convert this

(short string
long out)

for ($i = 0 \rightarrow n$) {

$hash[s[i]] = hash[s[i]] + 1$ (or $\max_f + 1$)

$\max_f = 0$: $0 = +n$

 for ($j = i + 1 \rightarrow n - 1$) {

$hash[s[i] - 'a']++$

$i = \max_f (= max_char, hash[s[i] - 'a'])$

$\max_f = max_f + 1$ (changes $= \max_f - \max_f$)

$+ 1$ if (changes $<= K$)

$+ 1$ if $\max_f = \max_char, j - i + 1$

 } (if $\max_f > 0$) else trim

Break:

g

: too many

return maxLen;

$\approx O(n^2)$

$O(26)$



Date / /

Date / /
Page _____

$S = A \bar{A} A B B C C D \times \dots \quad (k=2)$

$\downarrow l \downarrow$

$\text{len-maxf} \leq k$

$\text{maxf} = 0 \times 2$

$\text{maxlen} = 0 \times 2 \times 3 \times 4 \times 5$

$A, 1$

$B, 12$

$A, 12 \times 2$

$P = \text{empty} \quad (1, 0, 0, 0) = (0, 0, 0)$

fun (String s, k)

Σ

$S = \text{loop } l=0, r=0, \text{ maxlen}=0, \text{ maxf}=0$

$\text{hush}[\text{sc}0] = \text{soj};$

$\text{while } (r < s.\text{size}())$

$\text{loop} = \{ \}$

$\text{hush}[s[r]] - 'A']++;$

$\text{maxf} = \max(\text{maxf}, \text{hush}[s[c]] - 'A'])$

$\text{while } ((r-l+1)-\text{maxf} > k)$

$\text{loop} \Rightarrow \text{maxf} \{ \text{shard} \}$

$\text{hush}[s[r]] - 'A']--;$

$S = \text{loop}$

$\{ \quad \text{maxf} = 0$

$\text{for } (i=0 \rightarrow s) \{$

$\text{maxf} = \max(\text{maxf}, \text{hush}[i]);$

$l++;$

$\text{if } (s[i] == 'A')$

g

$\text{if } (s[i] == 'A')$

$\text{if } (r-l+1-\text{maxf} \leq k) \{$

$\text{maxlen} = \max(\text{maxlen},$

$\text{loop} \Rightarrow r-l+1);$

g

g

$$Tcc = O(N+N) \times 2^k \Rightarrow O(2N) \quad Tcw = O(2^k)$$

Binary Subarray With Sum k

$$\text{arr}[\bar{c}] = [1 \ 0 \ 1 \ 0 \ 1] \quad \text{ans} = 4$$

1 1 1
1 1

$$[1 \ 0 \ 0 \ 1 \ 1] \quad 1 - 0 = 1 \quad \text{goal} = 2$$

1 1 1
 1 1

$$\text{Sum} = 012B2 \quad \text{cnt} = 0 \times 2$$

Not working 'A' = (0)2 it don't work

- (0)2 don't know if move left or right

(+ < 0)2 - (1+2+1)2 it don't work

→ No. of Subarray where $\exists \text{Sum} \leq \text{goal}$

$$[1 \ 0 \ 0 \ 1 \ 1 \ 0] \quad \text{goal} = 2$$

1 1 1 1 1

$$\text{Sum} = 012B2 \quad \text{cnt} = 0 \times 2 + 3$$

$\exists (+ \rightarrow 0)2 + 1 + 1 + 1 = 4$

+4 +4

Total 19 = Subarray

+5

$\leftarrow = \text{goal}$



fun C nums , goal)

{

l = 0, r = 0, sum = 0, cnt = 0

while (r <= n)

{

sum += nums[r];

while (sum > goal)

{

sum = sum - nums[l];

l++;

g brakit : partite

cnt = cnt + (r - l + 1);

r++;

g

return fun (nums, goal) -

fun (nums, goal - 1);

O(2N) x O(2N) + O(1)

O(4N)



Date / /



Date / /

Page _____

Subarray with k diff Integers

$$\text{arr} = [1 \ 2 \ 1 \ 3 \ 4] \quad k=2 \quad \text{ans} = 3$$

1 2 1 3 2 3

Brute Force (n²)

Two for loops

$$O(n^2)$$

① Optimal Sliding Window

$$[2 \ 1 \ 1 \ 1 \ 3 \ 4 \ 3 \ 2] \quad k=3$$

1 1 1 3 4 3 2

cnt = 2 (loop counter)

$$\text{cnt} = 2$$

Same Problem to move x (0 → 3 → 12 or 0 to move l)

$$\begin{array}{l} 4 \rightarrow 1 \\ 3 \rightarrow 12 \\ 1 \rightarrow 123 \\ 2 \rightarrow 1 \end{array}$$

num → freq



no. of Subarray whose
different Integer $\leq k$

$\cdot (1-n) \text{ m}$ - $(n) \text{ m}$ presentation

[2 1 1 1 3 4 3 2]

9 ← ← ← ← ←

$$\text{Cnt} = 0 + 1 + 2 + 3 + 4 + 5 +$$

+

$$\text{Cnt} + = 9 - l + 1$$

combine subarray

$4 \rightarrow 1$
 $3 \rightarrow 2$
1, 123
e, 1

dd = f

fun (nums, k)

l = 0, r = 0, Cnt = 0, mPP;

x = 0000

printed while (r < n) {
 mPP[nums[r]]++;
 while (mPP.size() > k)

 mPP[nums[l]]--;

 if (mPP[nums[l]] == 0)
 mPP.erase(nums[l]);

 l++;

g

 Cnt + = r - l + 1;

}

 combine

return Cnt;

g

return fun(k) - fun(k-1);

$$T(c) \rightarrow 2(Ocn) + Ocn) + P + E : T(c) \rightarrow Ocn)$$

$$\rightarrow O(4n)$$

minimum Window Substring

$s = ddऱaaabbca$ $t = abc$

daaa X

Smallest

aaabb b c $O = ln^3$ $B = 12$, $O = Substring$

Smallest \rightarrow (baa) string containing all this

ans = bca

Brute force

Two loop

$O(n^2)$ + other

O optimal

Sliding window

$s = dd\cancel{aa}abbcc$

\cancel{a} \cancel{a}

$t = abc$

$cnt = 0 \ 8 \ 2 \ 8 \ 8 \ 3 \ 2$

$d \rightarrow 2$
 $c \rightarrow 20^{-1}$
 $b \rightarrow 20$
 $a \rightarrow 1$

$minlen = 10^9 8768443$

$SInd = -108283486$

$fun(s, t)$

{

$hash[256] = 0$

$l = 0, r = 0$

$min = 10^9, SInd = -1, cnt = 0$

$m = s.size(); n = t.size();$

~~code below~~

$for (i = 0 \rightarrow m)$

$hash[s[i]]++;$

$while (r < n) \{$

$if (hash[s[r]] > 0)$

$cnt = cnt + 1;$

$hash[s[r]]--;$

$while (cnt == m)$

$minlen = min(minlen, r - l + 1);$

$SInd = l;$

$hash[s[l]]--;$

$if (hash[s[l]] > 0) \{$

$cnt = cnt - 1;$

g
 $++;$

g.

return $\sin d = -1?$ S.Substring ($\sin d$, $\min len$)

j

$$T(c) \rightarrow O(N) + O(N) + O(m)$$

$$= O(2N) + O(m)$$

$$S(c) \rightarrow O(256)$$

(f, g) mif

o3 = consider

o3r o = l

o = +oo, l = oo, r = rrrr

o3r + o = o3r o3r r = oo

3 slides

(m - o = 3) not

(C++(3)) dead

3 (m - o) slides

(o < t[0]) dead

l + t[0] = too

l - (t[0] - 1) dead

(m - o + t[0]) slides

(l - o + t[0]) slides m - o + t[0]

l = brrr

l - (t[0] - 1) dead

3 (m - o + t[0]) slides

l - (t[0] - 1) dead

C++H