Arrays: Classical Problems

Course on Basic Data Structures (C++)

Pulkit Chhabra • Lesson 2 • Jan 23, 2021

Given an array of integers. (Size is N)

Print N numbers where: ans[i] = product(a[j])
$j \neq i$

Input → 5 1 2 3

Output → 3 5 5 5

pre_max[i] = max(a[0], a[1] ... a[i])

suf_max[i] = max(a[i], a[i+1] ... a[n-1]);

O(N)

```
for (i = 0; i < n; ++i)
    pre_max[i] = (i == 0) ? a[i] : max(pre_max[i-1], a[i])
```

[i-1]

i

pre_sum[i] = pre_sum[i-1] + a[i]

pre_min(i) = min(pre_min[i-1], a[i])

pre_fun(i) = fun(pre_fun[i-1], a[i])

gcd

lcm

product

and

or

xor

$gcd(a, b, c) \Rightarrow (gcd(a, b), c)$

$f(a[0], a[1] \ldots a[i]) = f(f(a[0] \ldots a[i-1]), a[i])$

Given an array with integers.

Queries : m

l r x

↳ a[l] + 2x, a[l+1] + 2x ---- a[r] + 2x

We want to retrieve the final array.

1 2 1 5 6 -3

1 2 3 1 5 7

l r x

1. 5 5 1
2. 1 6 -2
3. 3 4 2

1) 1 2 1 5 7 -3

2) -1 0 -1 3 5 -5

3) -1 0 1 5 5 -5

m = 3

Time - $O(N \times M)$

Required - $O(N+M)$

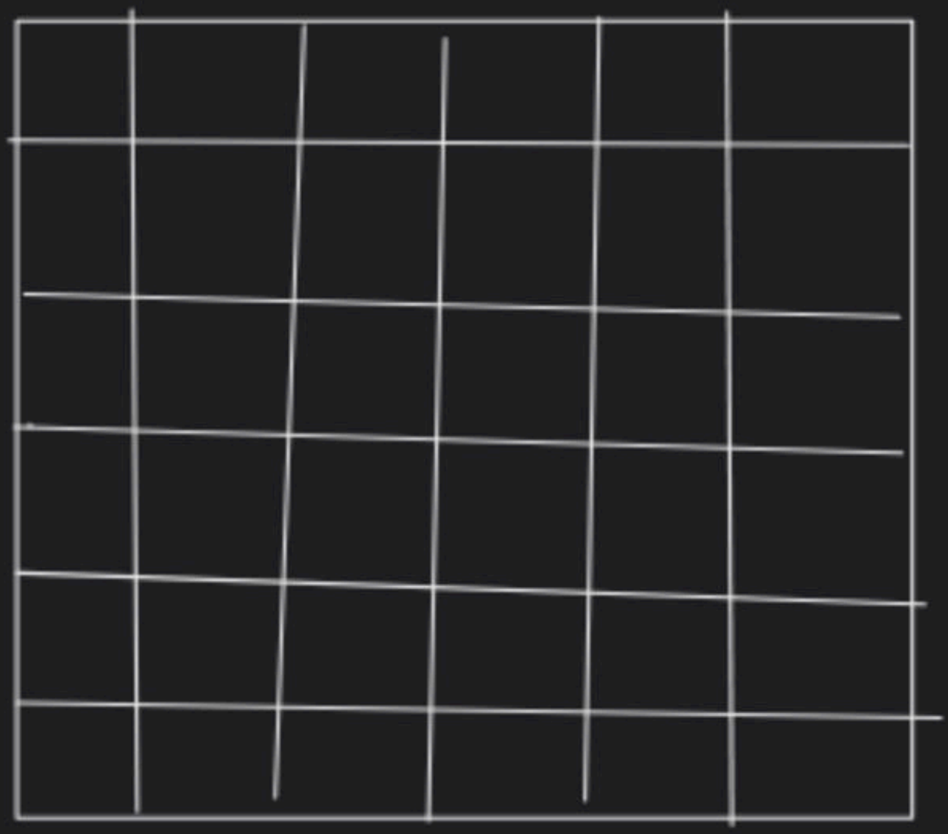
2D - vector (1-based indexing)

↳ queries

$i_1\ j_1$ $\quad$ $i_2\ j_2$

↓ $\quad$ ↓

top-left $\quad$ bottom-right



$O(i_2 - i_1)$ or $O(j_2 - j_1)$

$\Rightarrow O(1)$

$b \Rightarrow 0, 0, 0, 0 \ldots - \ldots - 0 \Rightarrow N$

$b \geq 0, 0, 0, +X \ldots 0 \ldots -X, 0, 0, 0, 0 \ldots$

$\Downarrow$

$\uparrow$ $r+1$

	0	1	2	3
b =>	0	3	1	4

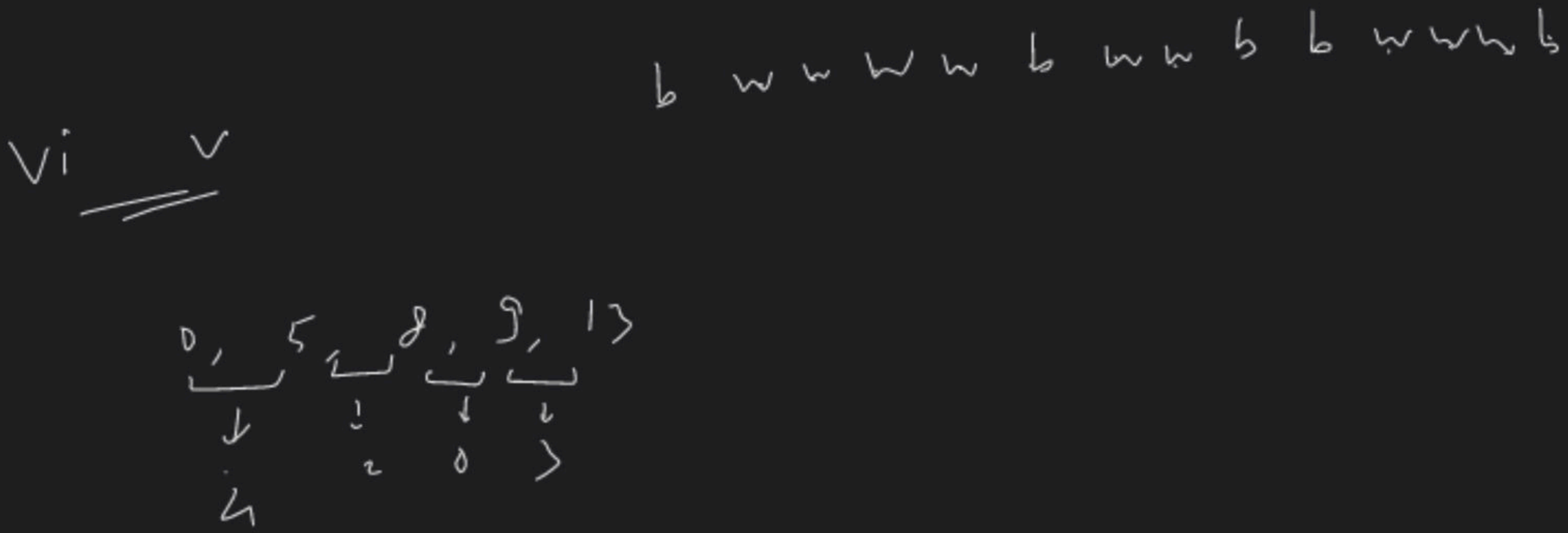
b[1] + b[0] => 3

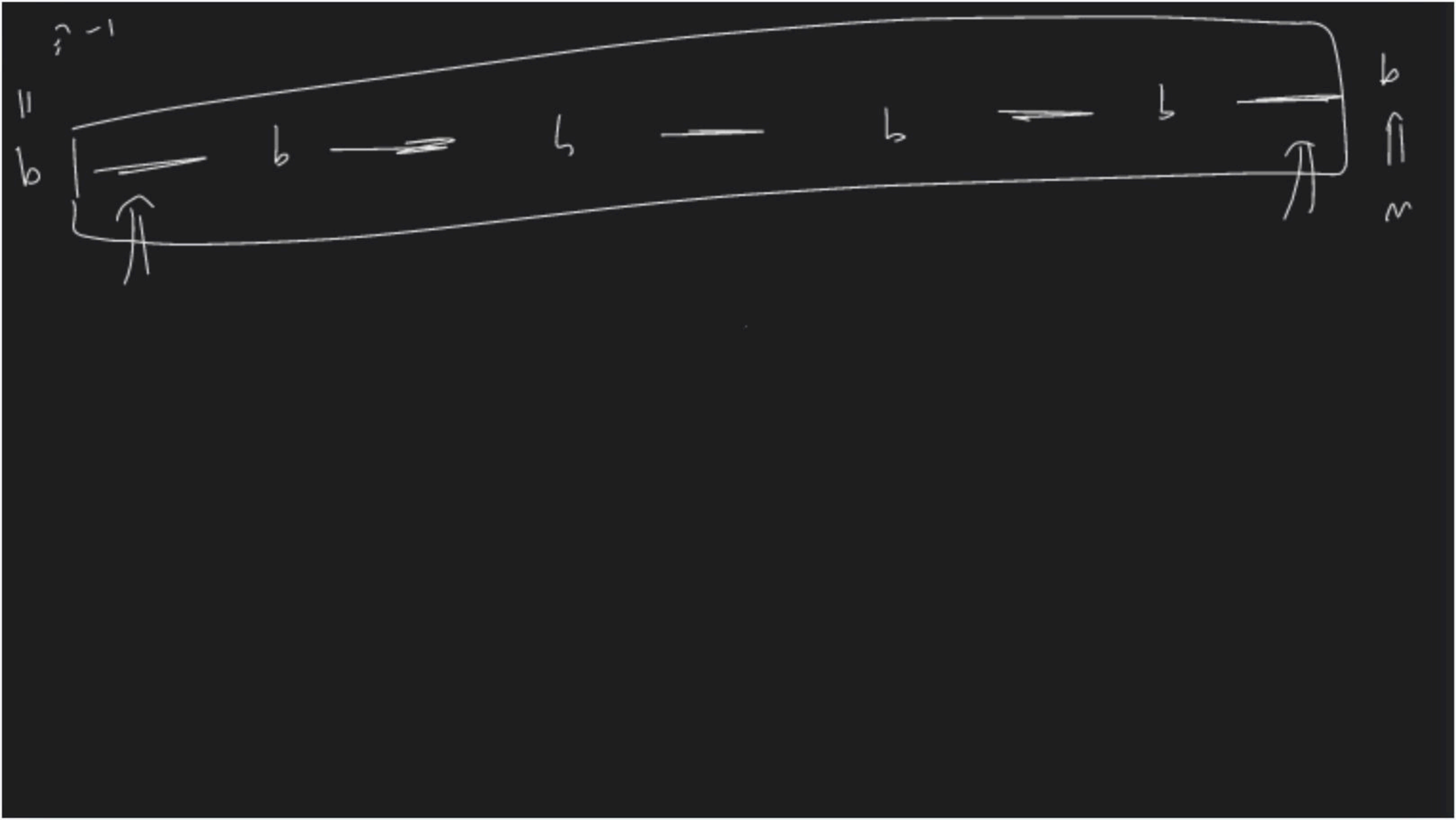
b[2] + b[1] => 4

b[3] + b[2] => 8

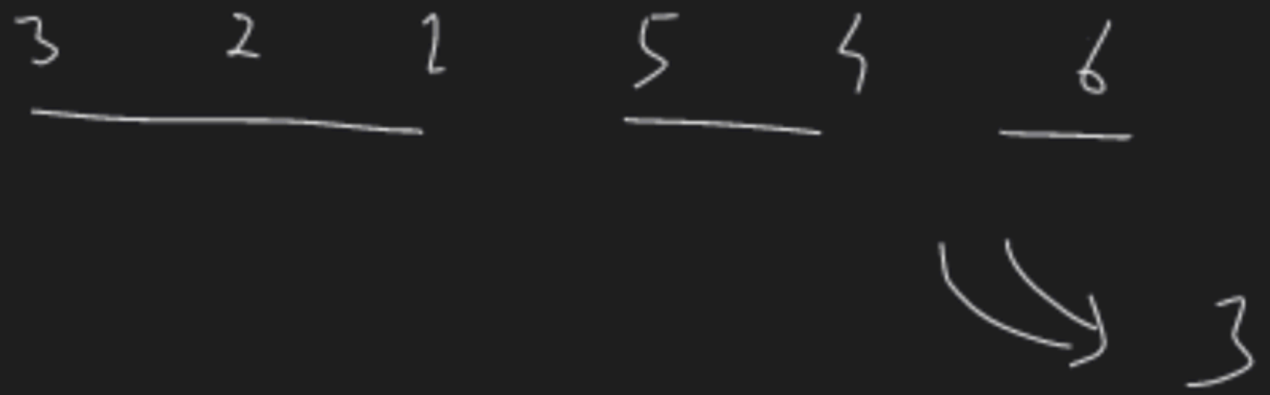
wwwww b ww b wwww b wwwww

$\frac{4*(4+1)}{2} - 1 \quad \frac{2*(2+1)}{2} + \frac{3*(3+1)}{2} + \frac{4*(4+1)}{2}$





Given an array, divide the array into maximum no. of contiguous parts, such that if we sort the individual parts, the array gets sorted.



1 , 2 , 3, 4 , 5 , 6

1 2 3 4 5 6

1

C = 2

3 1 2 ⇒ ans = 2

ans = 3

Ans = 5

3, 2, 1 | 4 | 5 | 6

3 2 5 1 0 6 $\Longrightarrow$ 3

1 2 3 4 5 6

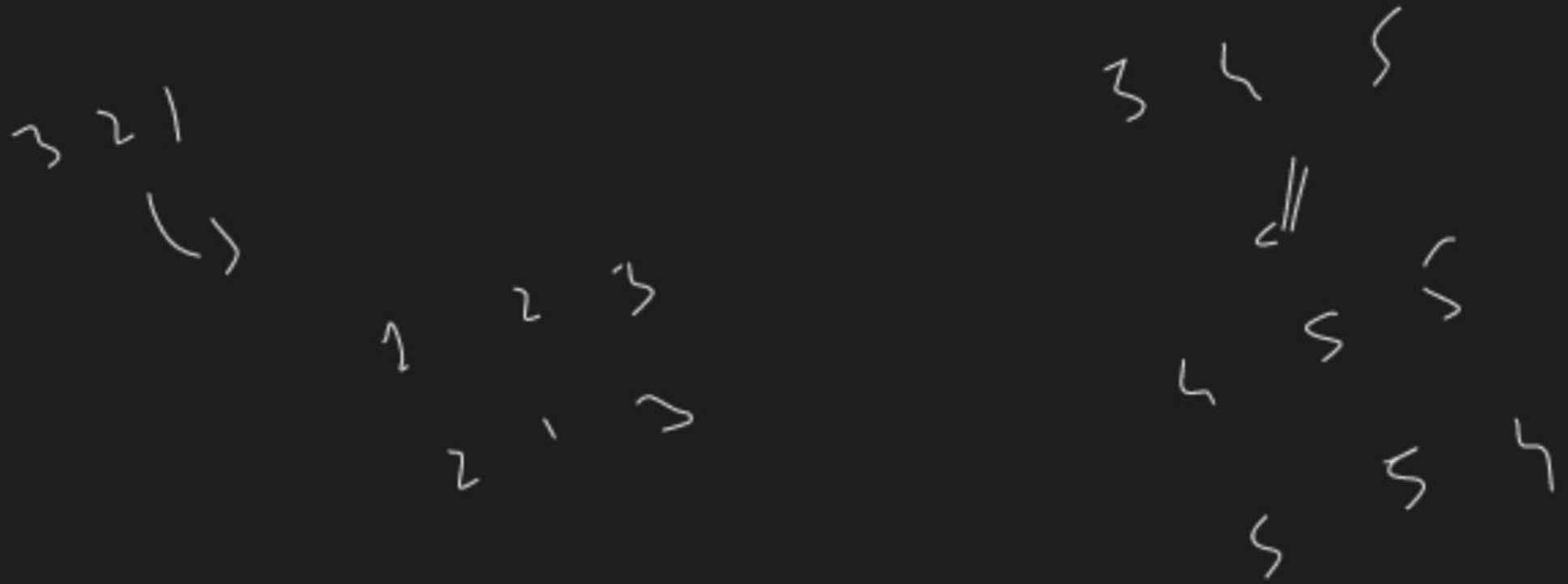
$pre\_max[i] \leq suf\_min[i+1]$

1.) Max parts

2) $i_1 j_1$ to $i_2 j_2$ $\Leftarrow$ sub-matrix $O(1)$

3.) Check if it is possible to rearrange the array s.t. sum of no 2 consec. elements is divisible by 3.

4) Maximum Subarray Sum



7 0 1

m^2 5