Introduction to Square Root Decomposition

Course on Square Root Decomposition

Utkarsh Gupta • Lesson 1 • June 21, 2021

Prefix Sums → A little reference

Segtree / ~~Fenwick~~ → Not covered

Intro to Sqrt Decomposition

Problem

Part 1

B

block size

we choose B

Part 2

No. of blocks $= N/B$

$N = 100$

$$O\Big(\underbrace{B}_{\sqrt{N}} + \underbrace{\frac{N}{B}}_{\sqrt{N}}\Big) \longrightarrow O(\sqrt{N})$$

$B = 1 \longrightarrow 101$

$B = 2 \longrightarrow 52$

$B = 3 \longrightarrow 36$

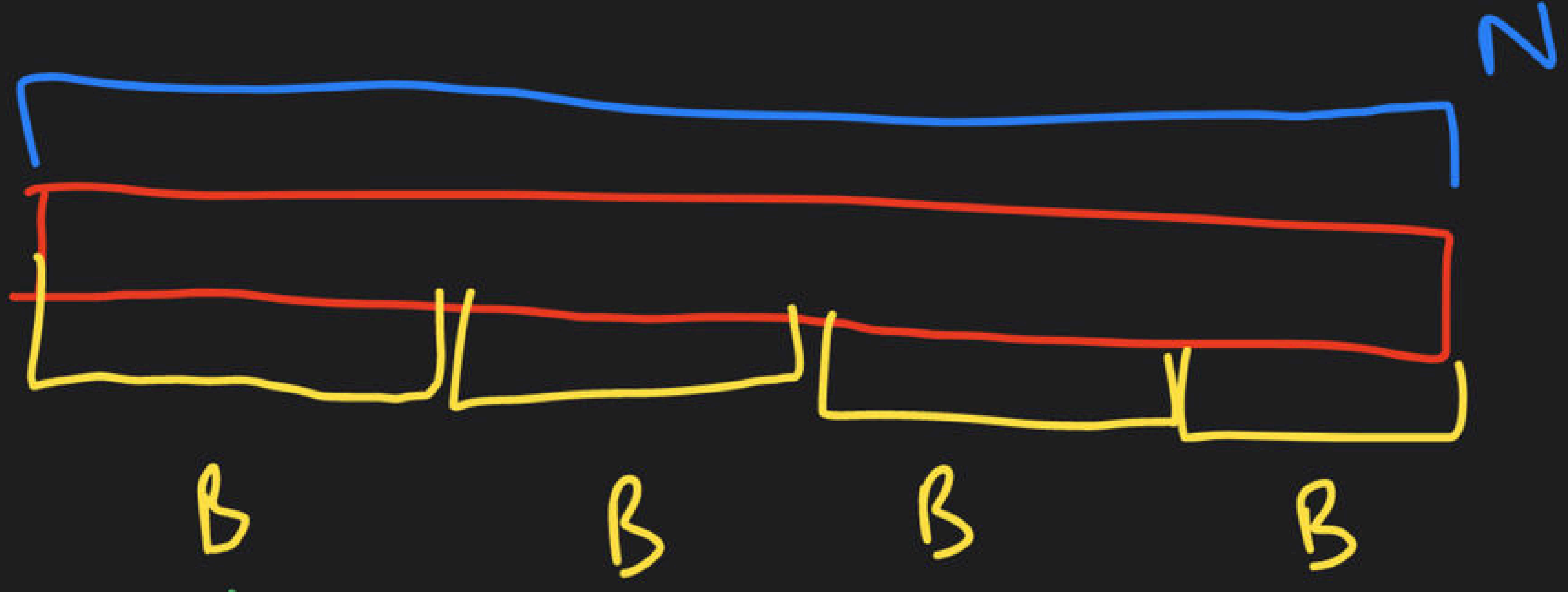
$B = 100 \longrightarrow 100 + 1 = 101$

→ Calculus

→ AM >= GM

$$\frac{B + \frac{N}{B}}{2} >= \sqrt{B \cdot \frac{N}{B}}$$

$$\boxed{B + \frac{N}{B} >= 2 \cdot \sqrt{N}}$$



$\left(\frac{N}{B}\right)$

$\sqrt{N}$

B

$\frac{N}{B}$

A $\quad N$ (length)

Q.

L, R

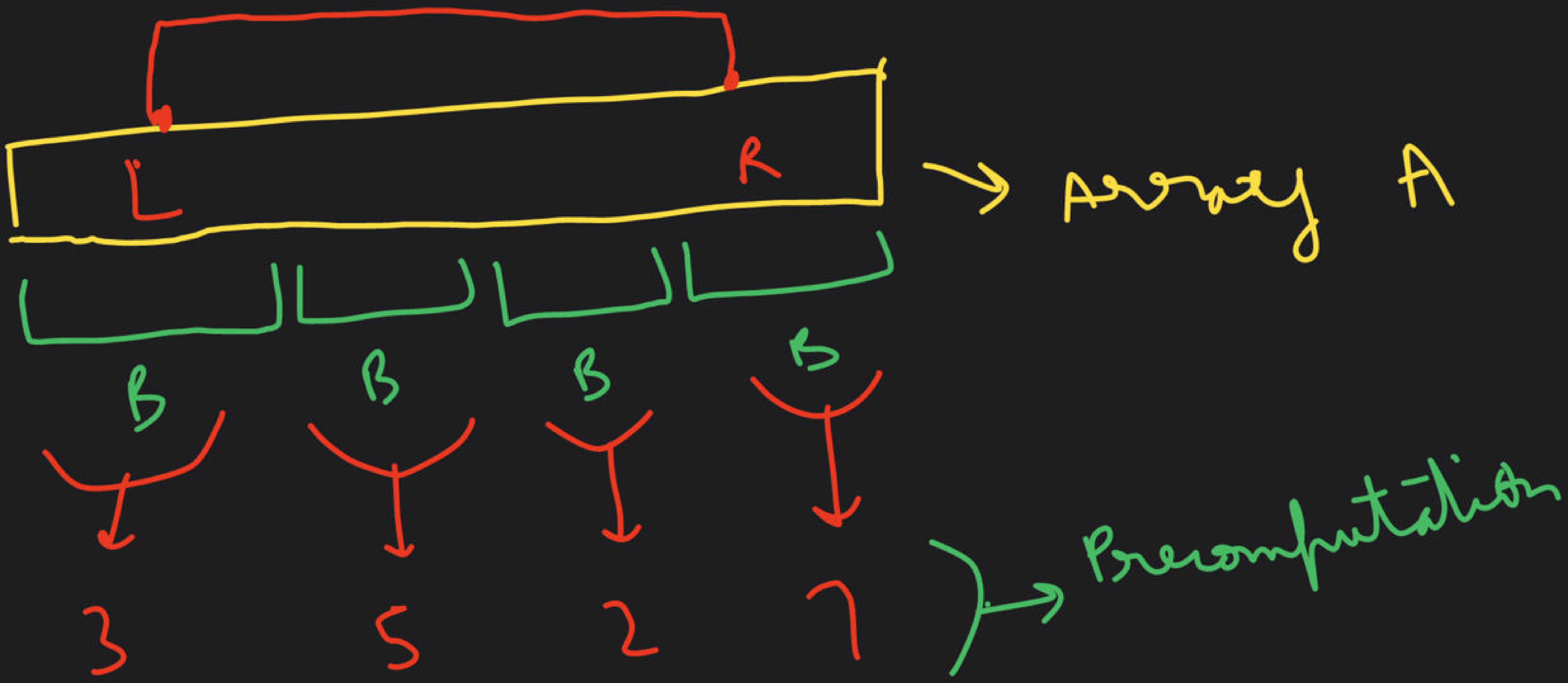
$(A[L] \ldots\ldots A[R])$

$\downarrow$

minimum value

$\boxed{O(N^2)} \rightarrow$ too slow

SQRT Decomposition - ans = min(5, 2, remaining values)



CSES Problem Set

Dynamic Range Sum Queries

TASK | STATISTICS

Time limit: 1.00 s **Memory limit:** 512 MB

Given an array of n integers, your task is to process q queries of the following types:

1. update the value at position k to u
2. what is the sum of values in range $[a, b]$?

Input

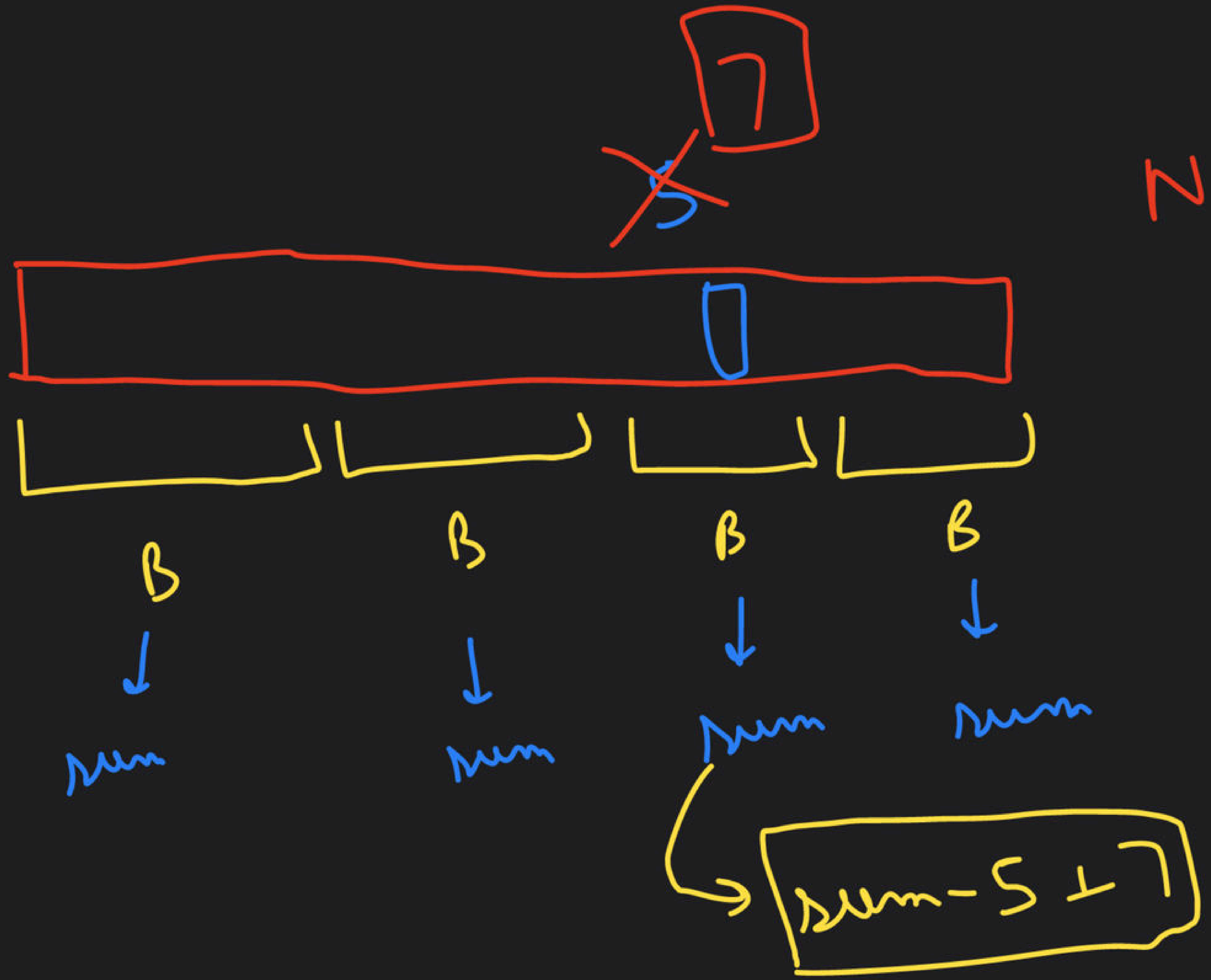
The first input line has two integers n and q: the number of values and queries.

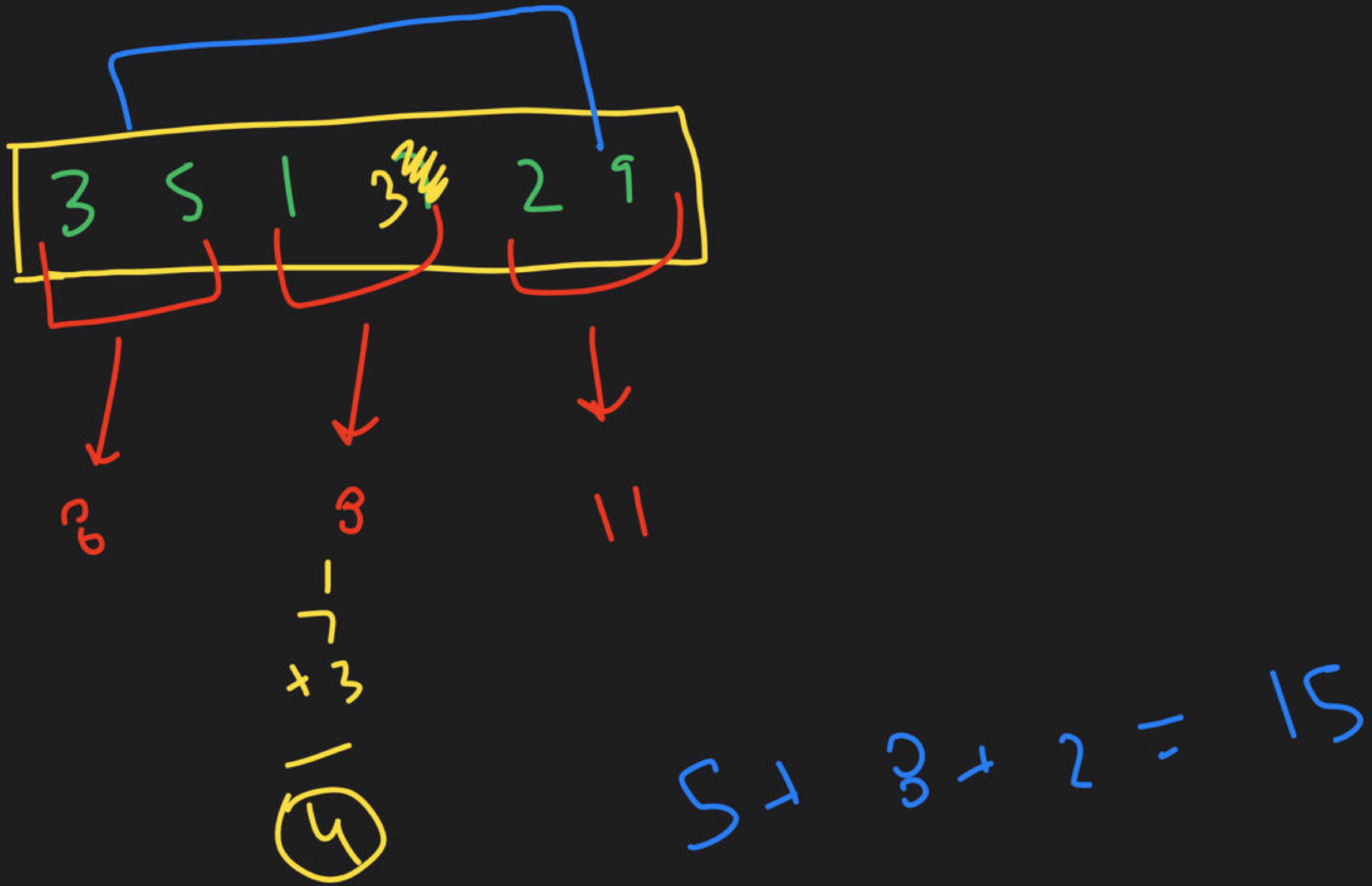
The second line has n integers $x_1, x_2, \ldots, x_n$: the array values.

Finally, there are q lines describing the queries. Each line has three integers: either "1 k u" or "2 a b".

Output

Print the result of each query of type 2.


7
5
N
B
B
B
B
sum
sum
sum
sum
sum-5+7



3 5 1 3 2 9
8
8
11
1
7
+ 3
4
5 + 8 + 2 = 15

Implementation details

int getBlock(id);

for i : [1, n]:

sum(getblock(i)) += x[i];

all blocks have correct sum.

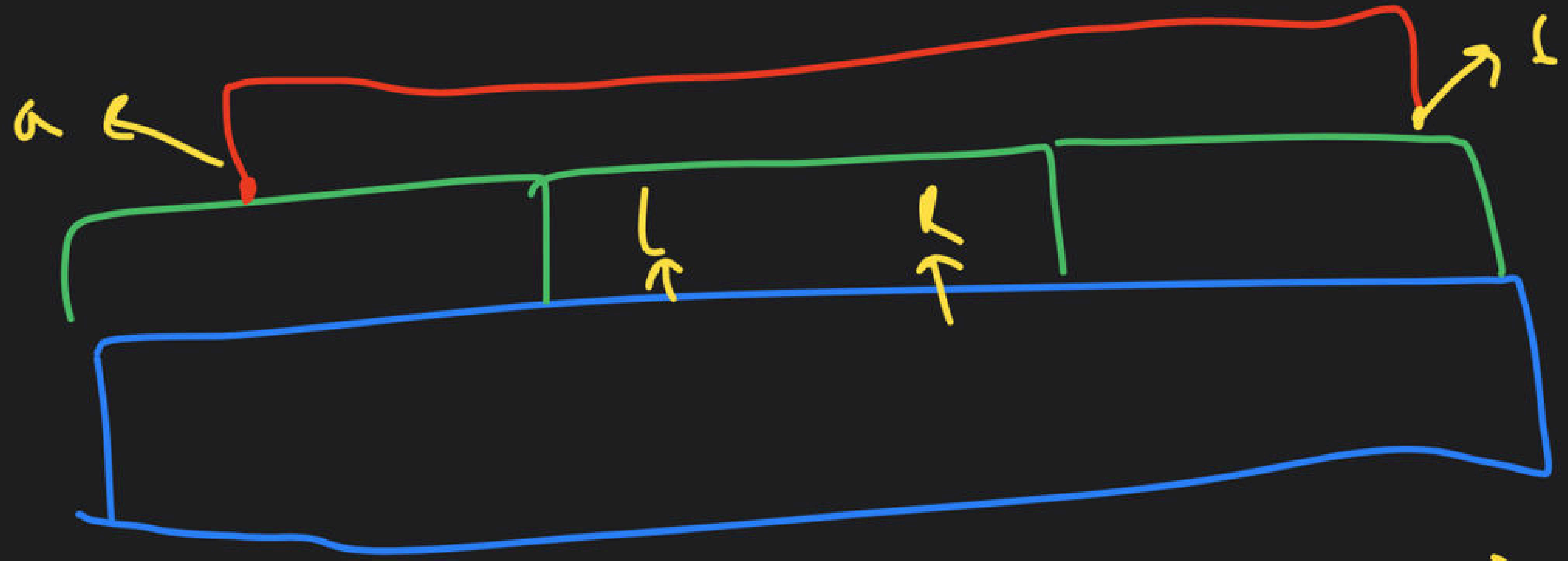
Updates (type 1 queries) :-

k, v

$sum[gB(k)] -= x[k]$

$x[k] = v$

$sum[gB(k)] += x[k]$




```
i = b;
while( gB(i) == gB(b)) {
    ans += x[i];
    i--;
}
```

take block a

take middle blocks

if block is same there are no middle blocks.

if(blocka != blockb){

take block b

}

$$O\left(B + \frac{N}{B}\right)$$

we don't know value of B.

POLL:

if B increases, then TC will

A. Increase

B. Decrease

C. Depends

why B is always 500 → works for maxN

worst case when N is maximum

2e5

$500 \simeq \sqrt{2e5}$

GIVEAWAY - Give Away

#tree #binary-search

You are given a **1-indexed** array **X**, consisting of **N** integers, and a set of **Q** queries. There are two kinds of queries:

1. **0 a b c**
   Here you are required to return the number of elements with indices in **[a,b]** greater than or equal to **c**
2. **1 a b**
   Here you are required to change the $\mathbf{a}^{\mathbf{th}}$ element of array to **b**.

Input Format:

First line contains **N**, the number of elements in the array **X**. The next line contains **N** space separated integers representing the elements of **X**. The third line of input contains a single integer, **Q**, the number of queries. The next **Q** lines of input each contain queries of two kinds as described above.

Output Format:

Q lines with the ith line contains the answer for the $\mathbf{i}^{\mathbf{th}}$ query

Constraints:

1 ≤ N ≤ 5*10^5
1 ≤ Q ≤ 10^5
1 ≤ X[i] ≤ 10^9
1 ≤ a ≤ b ≤ N for query type 0
1 ≤ a ≤ 10^5, 1 < b ≤ 10^9 for query type 1
1 ≤ c ≤ 10^9


3 1 7
sort
1, 3, 7
lower_bound ( )

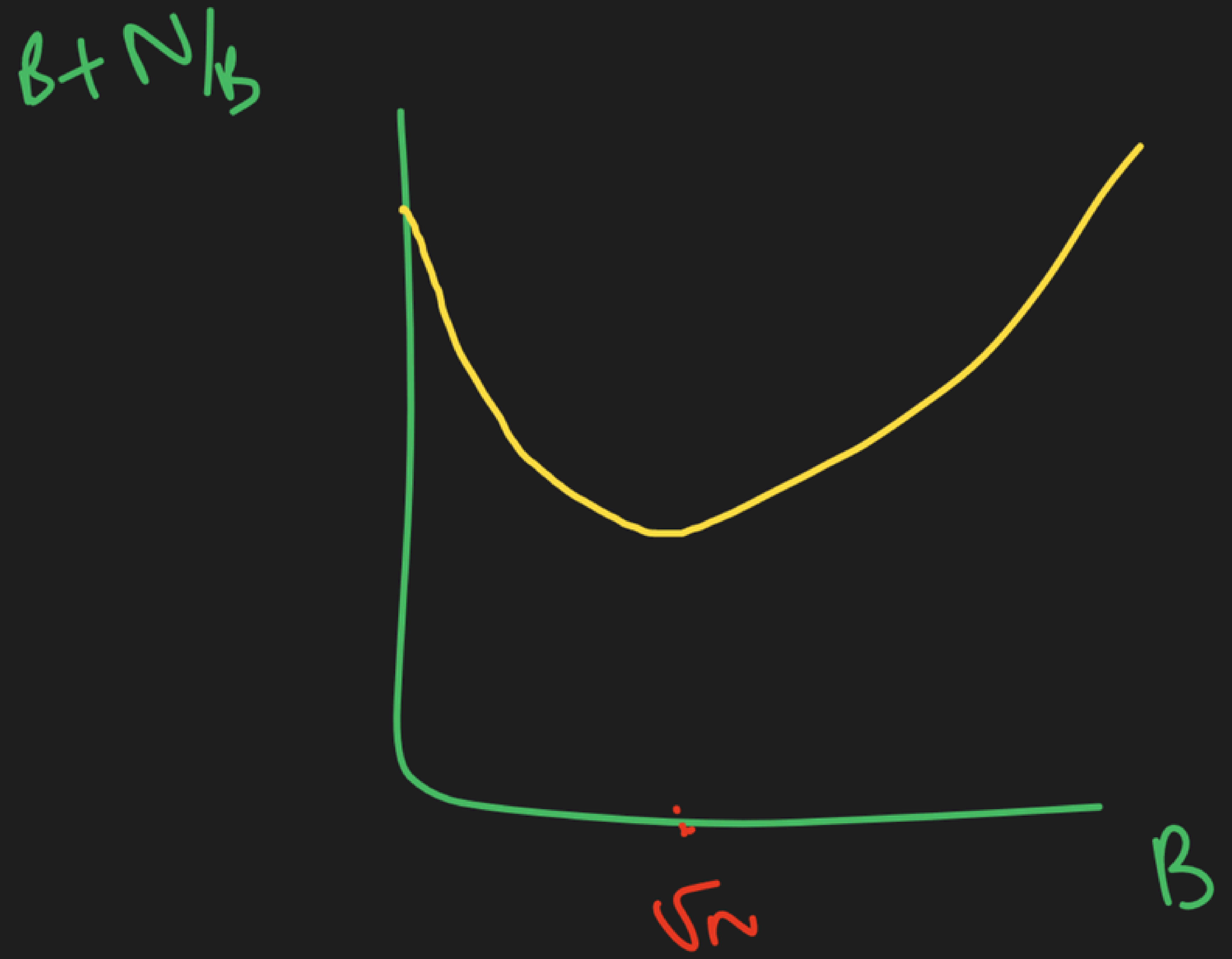
multiset

set $\rightarrow$

$\{1, 3, 7\}$

~~how many elements are smaller than 4~~

can't answer


B+N/B
√n
B

take block a :

```
i = a;
while(gB(i) == gB(a) && i <= b){
    ans += x[i];
    i++;
}
```

for i : [blocka+1, blockb-1]:

ans += sum(i)

$O(B)$ elements remaining in left

$O(B)$ " " " right

$O(N/B)$ blocks in the range

$B = \sqrt{N}$

$$O\left(2B + \frac{N}{B}\right) \equiv \underbrace{O\left(B + \frac{N}{B}\right)}_{O(\sqrt{N})}$$

5, 3, 1, 2, 7

5 3 1 1 1

pre[L-1] pre[R]

the operation is not invertible

min
max
OR
AND

$\underbrace{A + B}$ $= C$

$C - B = A$

$A + B = C$

$A + B - B = C - B$

$\Rightarrow \boxed{A = C - B}$

additive inverse

$B + (-B) = 0$

$$\min(A, B) = C$$