range: indices 2 to 6

	0	1	2	3	4	5	6
A[] =	7	6	1	4	5	2	4

Queries	Result
sum(2,4)	1+4+5 = 10 ←
min(1,5)	1 ←
mostfreq(2,6)	4, 2
⋮	

- range query   sum(i, j)
- point query   A[i]
- point update   A[i] = x, A[i] += x;
- range update   Add(i,j) = x, A(i,j) += x;

Queries

Offline (no update)
- prefix sum array (sum)
- Sparse Table (min/max)
- Stack + DSU (min/max)
- Mo's algorithm
- set or unordered_set, map

Online (with updates)
- O(QlogN) – Fenwick Tree (BIT), sum queries
- O(Qsqrt(N)) – SQRT Decomposition
- O(QlogN) – Segment Tree (Platinum)

For all types of queries

Offline Sum Queries:

1D Prefix Sum Array   O(N + Q)

2D " " Array (matrix)   O(NM + Q)

Offline min/max Queries (RMQ)

N elements and Q queries

	0	1	2	3	4	5	6	7	8	9	10
A[] =	5	2	9	1	10	1	4	7	8	4	7

Sol 1: Brute-Force   O(QN)

Queries	Answer
1 – 3	1
5 – 9	1
0 – 2	2
0 – 7	1
6 – 10	4

Sol 2: Look-up table. Precalculate all possible queries.

(end pos)

	0	1	2	3	4	5
0	5	2	2	1	1	1
1	–	2	2	1	1	1
2	–	–	9	1	1	1
				1	1	1

6 - 10   4

1	–	–	–			
2	–	–	9	1	1	1
3	–	–	–	1	1	1
4	–	–	–	–	10	1

(start pos)

$O(N^2 + Q)$, $O(N^2)$ space

Sol 3: Sparse Table (faster version of look-up table)

	0	1	2	3	4	5	6	7	8	9	10
A[] =	5	2	9	1	10	1	4	7	8	4	7

min(2,7)

len = j-i+1 = 7-2+1 = 6 (integer)

$$2^a = 2^{a-1} + 2^{a-1}$$

R1

R2   R3

min(R1) = min(min(R2), min(R3))

Construct Sparse Table

(length, 2^j)

	2^0 0	2^1 1	2^2 2	2^3 3
0	5	2	1	–
1	2	2	1	–
2	9	1	–	–
3	1	1	–	–
4	10	1	–	–
5	1	1	–	–
6	4	4	–	
7	7	7	–	
8	8	4		
9	4	4		
10	7	–		

(start pos)

ST[][]

len = 10-0+1 = 11

num of cols is lg(11) = 3

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

Construct the sparse table in column manner.

$$2^2 = \underline{2^1} + 2^1$$

$$st[i][j] = \min(st[i][j-1],\ st[i+2^{j-1}][j-1])$$

0 2   2   2 1

st[1][2] = min(2, 1)

st[2][3] = min(st[2][2], st[6][2])

Answering one Query

$\min(i, j) = ?$ $\quad$ $len = j - i + 1$ (len of the range query) $\quad$ $len = 7$

$k = \lg_2(len)$ $\quad$ (len of the sub-range is 2^k) $= \underline{4}$

i ⊢——— $len = 7$ ———⊣ j

⊢— $len = 4$ —⊣ $\quad$ ⊢— $len = 4$ —⊣

?

$$\min(i,j) = \min\left(\min(i, k),\ \min\left(\underline{i} + \underline{len - 2^k},\ k\right)\right)$$

$$\min(1,7) = \min\left(st(1, 2),\ st(\underline{4}, 2)\right)$$

	0	1	2	3	4	5	6	7	8	9	10
A[] =	5	2	9	1	10	1	4	7	8	4	7

st (1,2) $\quad$ st (4,2)

$O(N \lg N + Q)$ time

$O(N \lg N)$ space

Sol 4: Stack + DSU

	0	1	2	3	4	5	6	7	8	9	10
A[] =	5	2	9	1	10	1	4	7	8	4	7
Next smaller el. (index)	1	3	3	-							

2-1 $\quad$ 4-1 $\quad$ 1-2 $\quad$ 3-1 $\quad$ 5-4

Stack: 3, ~~2~~, ~~1~~, ~~0~~

stack

DSU (compress paths)

Queries	Answer
1 - 3	1
5 - 9	1
0 - 2	2
0 - 7	1
6 - 10	4

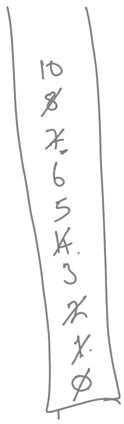
1) Sort queries by end points in increasing order

2) Find the next smaller element with a stack in $O(N)$ time, at the same time when we are at the ending point of a query; answer that query with

a query; answer that query with the help of DSU in almost linear time.

Finding next smaller element:

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



cur = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~ ~~7~~. ~~8~~ ~~9~~ 10

stack (index) (increasing order)

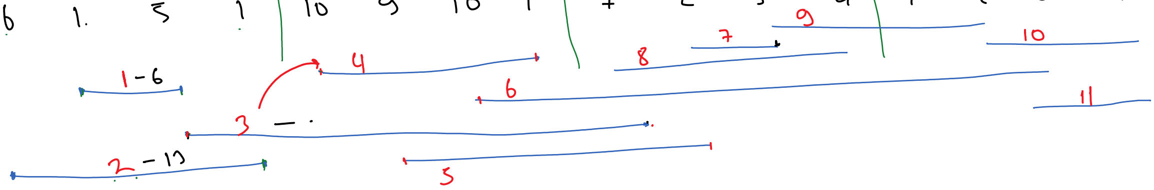
$O(Q \lg Q + N)$

Mo's Algorithm (For different types of offline queries)

len = 16, $\sqrt{len} = 4$

	0				1				2				3		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	1	5	1	10	9	10	1	7	2	3	4	1	2	6	7

left ↓ (at 1), right ↓ (at 2)



1) Sort queries. a) starting block
b) if in the same block – end position } in increasing order.

i ____ j

left = 0
right = −1

sum = 0 + 6 + 1 + 5 − 6 = 6 + 1 + 6 = 13 + 10 + 9 + 10 + 1 + 7 − 6 − 1 =

For one block: right → O(N)

left → Q√N

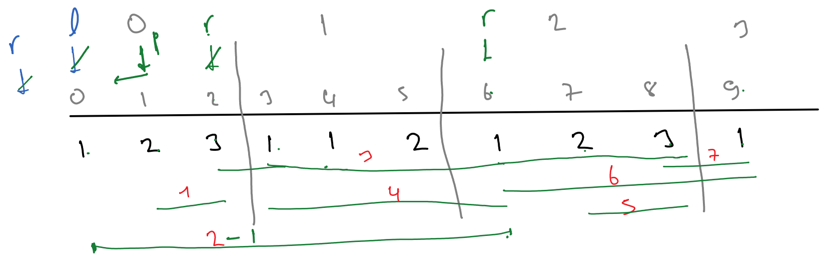
Example Problem

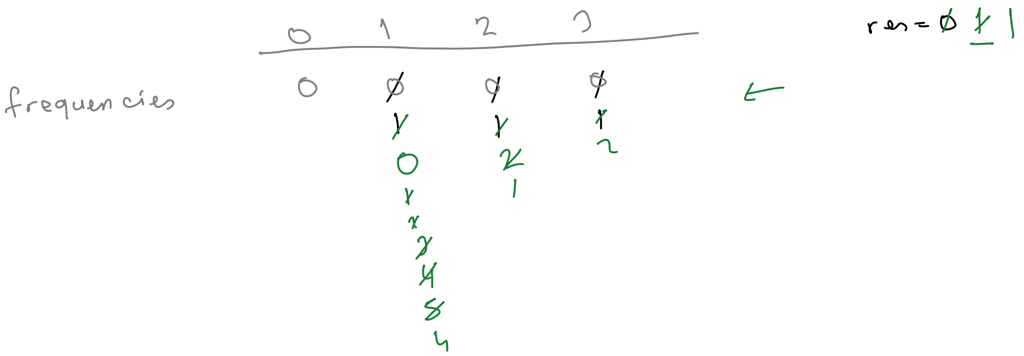
Given an array of size N. All elements of array <= N. You need to answer M queries. Each query is of the form L, R. You need to answer the count of values in range [L, R] which are repeated at least 3 times.

Example: Let the array be {1, 2, 3, 1, 1, 2, 1, 2, 3, 1} (zero indexed)

Query: L = 0, R = 4. Answer = 1. Values in the range [L, R] = {1, 2, 3, 1, 1} only 1 is repeated at least 3 times.

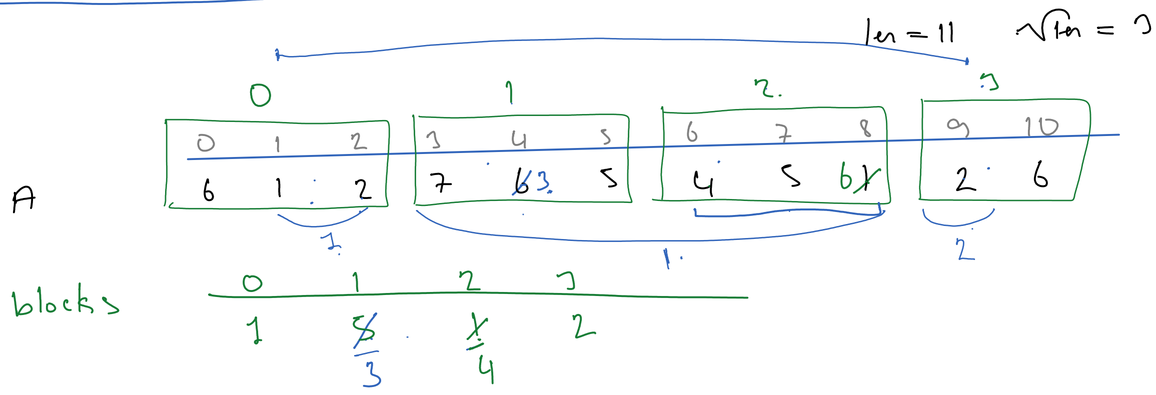
Query: L = 1, R = 8. Answer = 2. Values in the range [L, R] = {2, 3, 1, 1, 2, 1, 2, 3} 1 is repeated 3 times and 2 is repeated 3 times. Number of elements repeated at least 3 times = Answer = 2.





Answering Online Queries with SQRT Decomposition

Range Min Queries



min (1, 9) → 1   O(√N)   O(Q√N)

min (6, 8) → 1   O(√N)

update (4, 3)

update (8, 6)

update (4, 3)
min (2, 9)

update (8, 6)