↳ Problem Solving → STL

Vectors (1)

Maps (1)

PBDS → Policy Based DS

Prerequisite → for loops, while loops, arrays

What is STL ?? Standard Template Library

- Stacks, queues, LL
- Maps, unordered_maps
- Priority Queue
- Policy Based DS
- sorting
- Binary Search
- next_permutation

Q ⇒ You have been given an array of all zeroes. Size of array will be N. Then you will get Q queries. In each query you get a no. 'L' & 'R'. Increase all value from index [L, R] by 1.

At last print the array.

[0 1 2 2 2]

0 1 2 3 4

query → 3

2 , 4 2

1 , 2,

3,

0 , 1 , 2 , 2 , 2

$N \leq 10^7$

$Q \leq 10^6$

$O(Q \cdot N)$

$N \le 10^7$

$Q \le 10^5$

Brute force

Difference vector/array

+2 +2 +2 +2 +2

[0, 1, 1, 0, 0, -1, -1]

0 1 2 3 4 5 6

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

3

2, 4

1, 5

3, 5

```
arr[L]++;
arr[R+1]--;
```

L, R

R+1

cumulative sum

[0, 0, 1, 0, 0, -1, 0]

0 1 2 3 4 5 6

[0 0 1 1 1 0, 0]

2, 4

We would have increase all value in right of 2 still under 4 by value 1

arr[L] ++;

L, R

arr[R+1] --

for fun

∀ i ∈ [1, n-1]

cumulative sum

arr[i] = arr[i] + arr[i-1]

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

0 1 2 3 4 5 6

2, 4

arr[b] = arr[s]

arr[l]++; arr[r]--

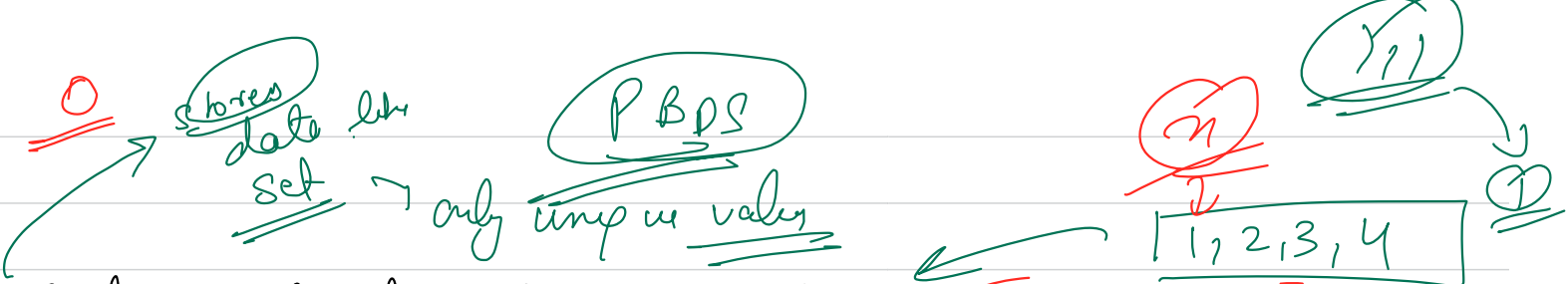
1, 4

2, 3

```
for (i=1; i<n; i++) {
    arr[i] += arr[i-1]; // cumulative sum
}
```

O(q + n)

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

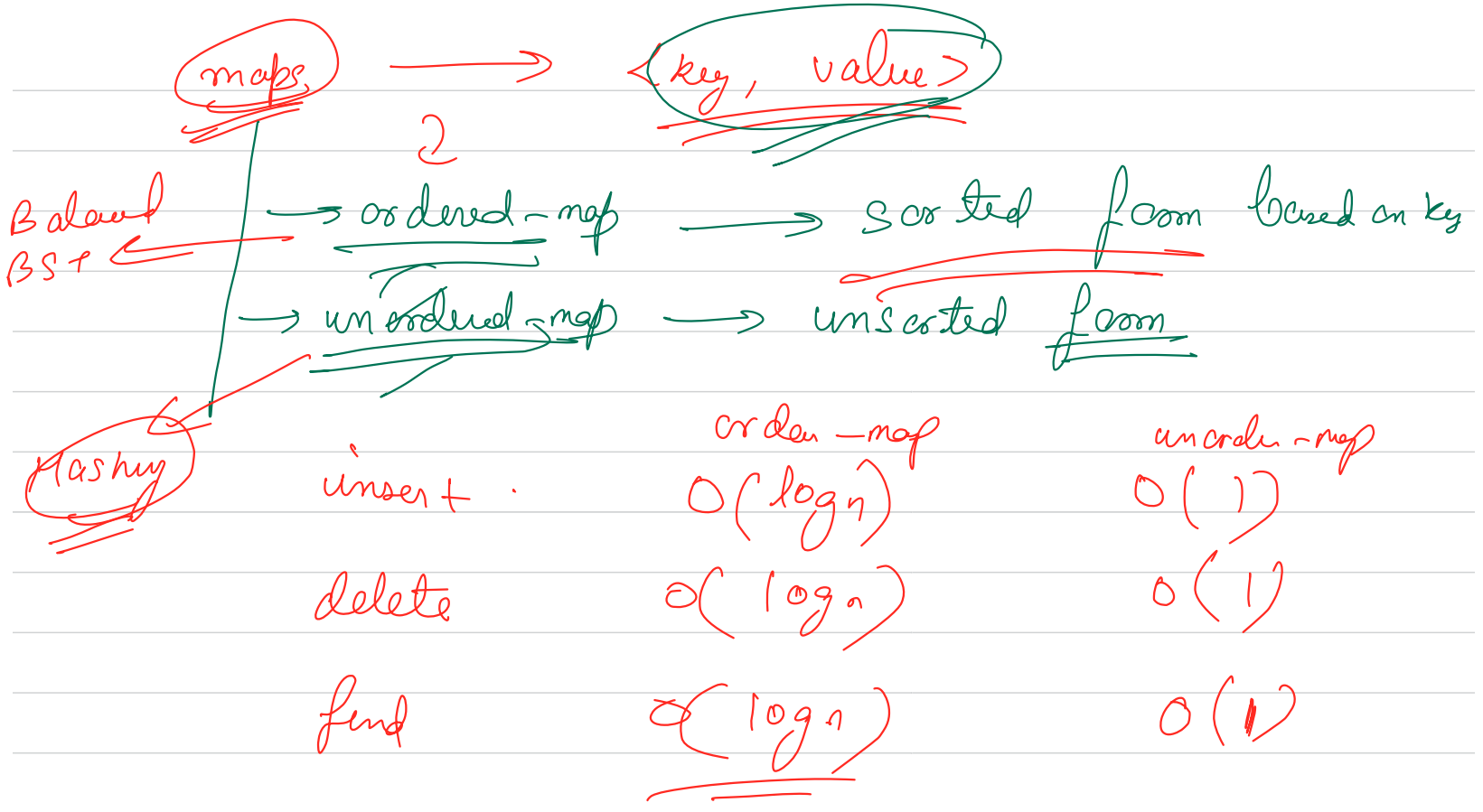


Policy Based Data Structure

↳ Special data structure which answer the following things

- find_by_order(K) → return the pointer to the K^{th} largest element
- order_of_key(X) → return how many elements are strictly lesser than X

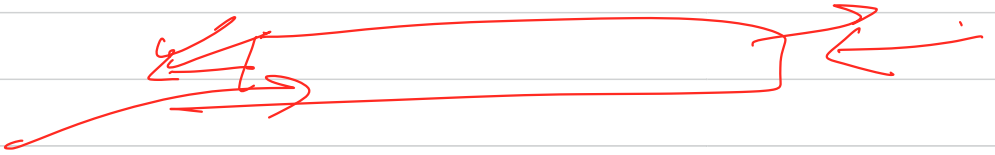
count

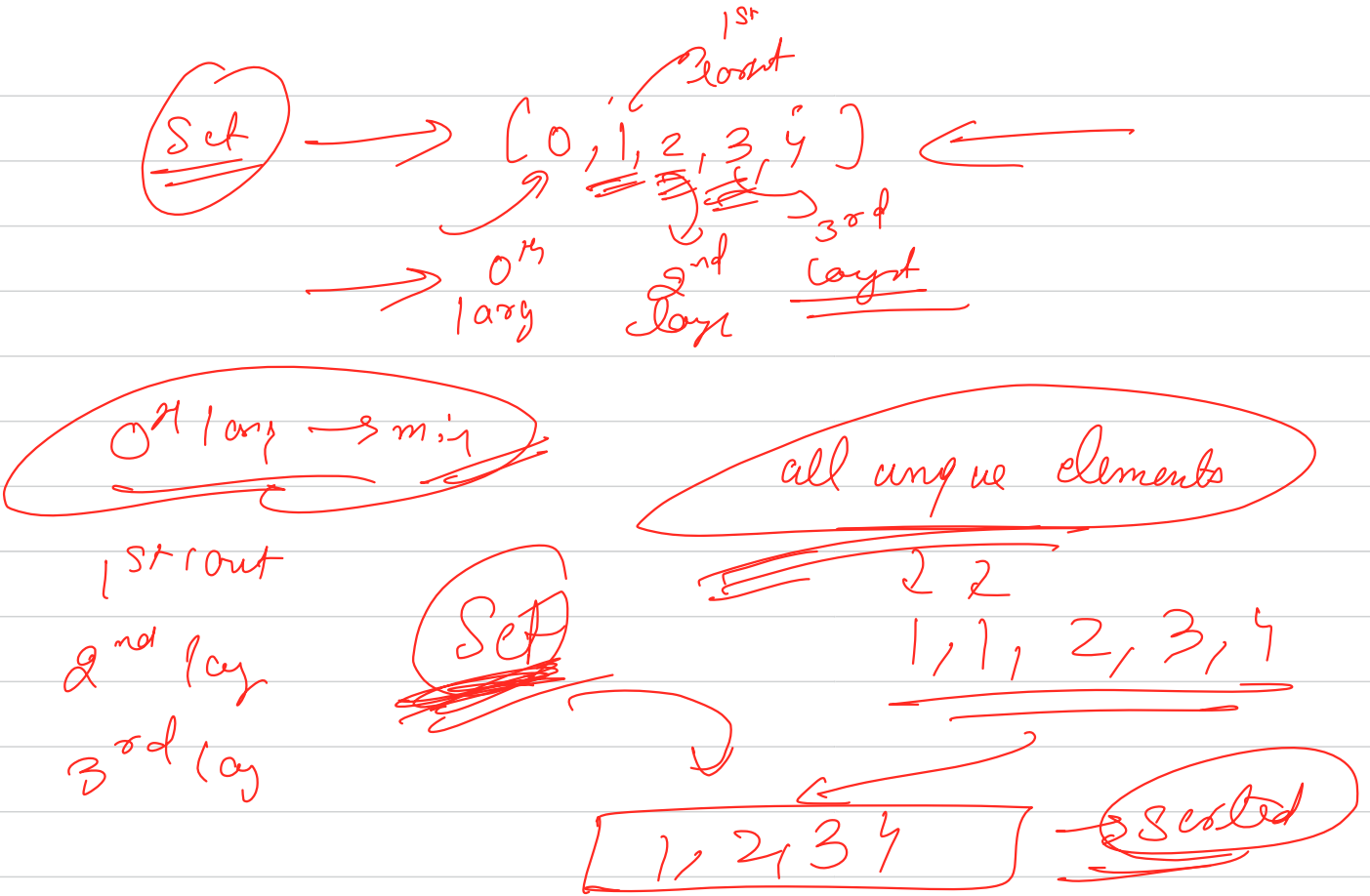


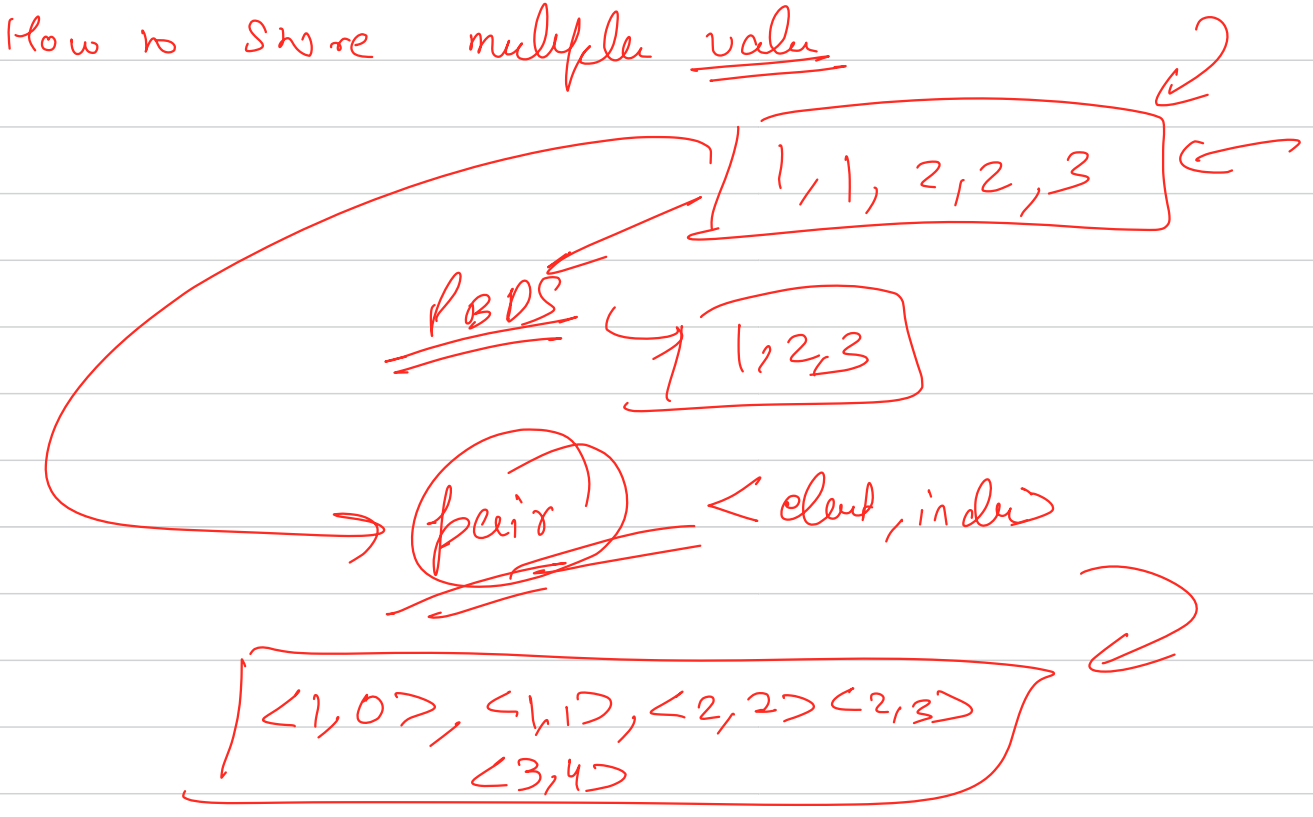
	ordered-map	unordered-map
insert	$O(\log n)$	$O(1)$
delete	$O(\log n)$	$O(1)$
find	$O(\log n)$	$O(1)$

Hashy or any compare is done on Key

Dequeue → double ended queue




Set
1st
Layout
[0, 1, 2, 3, 4]
0th
layer
2nd
layer
3rd
Layout
0th layer → min
1st layout
2nd layer
3rd layer
all unique elements
2 2
1, 1, 2, 3, 4
Set
1, 2, 3, 4
sorted


How to store multiple value
1, 1, 2, 2, 3
PBDS
1, 2, 3
pair
< elem, index >
<1, 0>, <1, 1>, <2, 2> <2, 3>
<3, 4>

Vectors // list / arraylist

→ dynamic arrays grow as & shrink as needed