Q. Given an array of 0s and 1s, bring all 0s before 1s.

Q. Check palindrome.

Q. Reverse array/string

} using two pointers

10

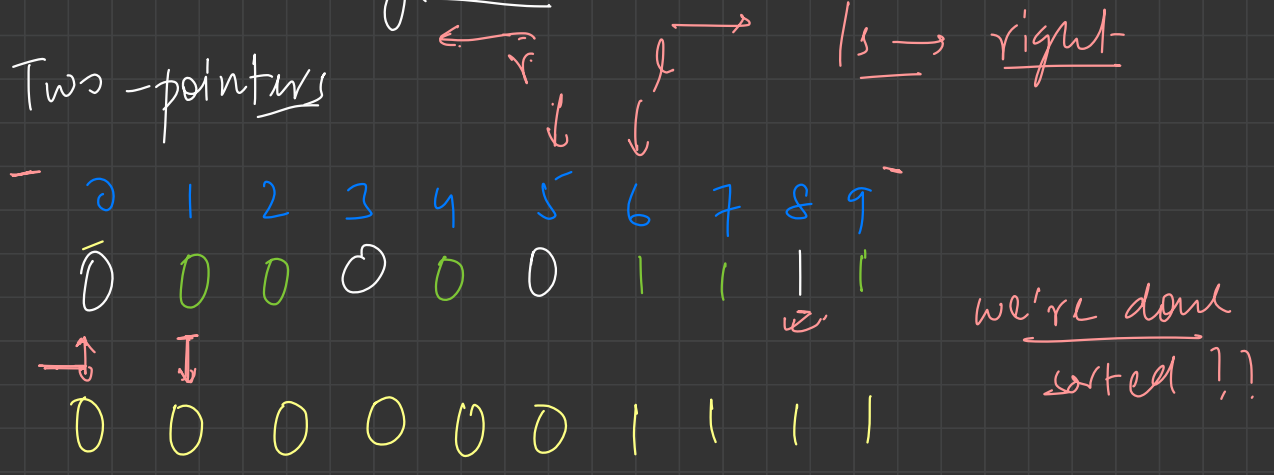
0 1 1 0 1 0 0 0 1 0

} input

output

0 0 0 0 0 0 1 1 1 1

1. Sorting.
2. loop nesting.
3. count (counting sort)
4. Two-pointers



we're done sorted !!

should i swap element at l and r?

Q. subarrays with exactly K values. → good subarrays

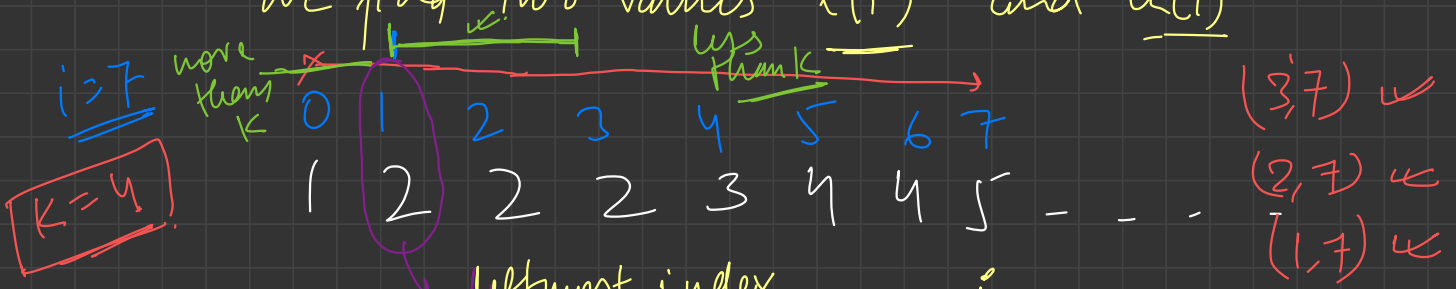
① Get all subarrays, and check them

→ time limit exceed

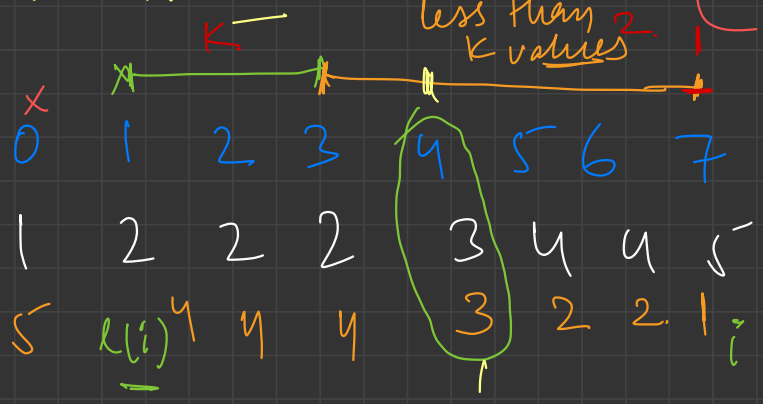
② We find the no. of good subarr ending at a certain index i

— how to find no. of good subarrays for a particular index.

we find two values $l(i)$ and $u(i)$



to find how many subarrays end at $i=7$, with exactly $4(K)$ values (start before or at 7)



while we were going left from i, either

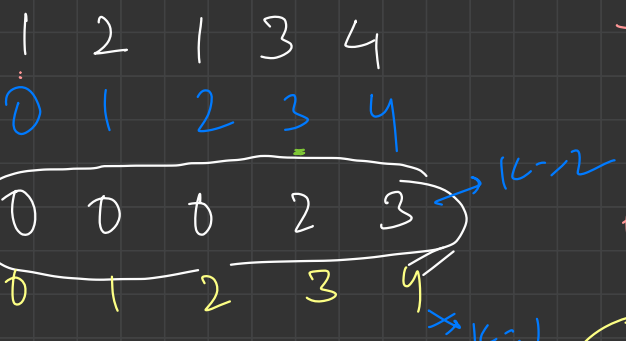
leftmost index from where exactly k-1 values exist wrt i.

↓
u(i)

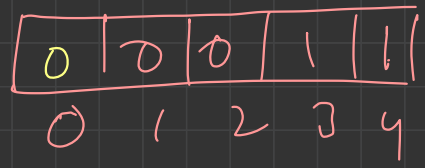
no. of distinct values ++;
or stayed same.

$u(i) - l(i)$ → no. of good subs ending at i.

k = 2



distinct = 2
(j → i) values



u(i) → l(i) (j → i)

// since, dist > k, we should shift j right until there are no more than k values

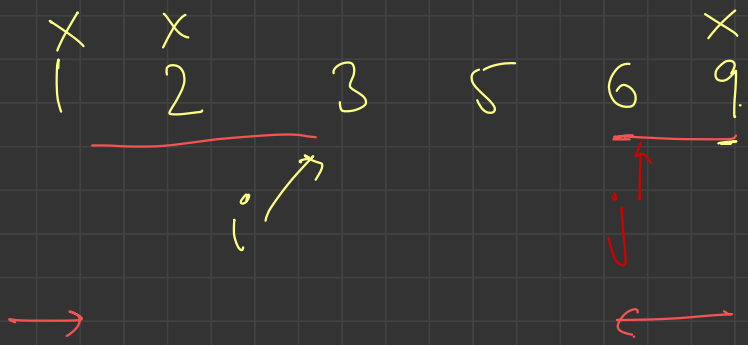
Q.1. Given a sorted array, find a pair of values sum to X.

eg: 1 2 3 5 6 9

X = 9

output: 3, 6

3 + 6 = 9 = X



$x = 9$

$a[i] + a[j] = 1 + 9 = 10 \geq x$

$j{-}{-}$

$a[i] + a[j] = 1 + 6 = 7 < x$

$i{+}{+}$

$a[i] + a[j] = 2 + 6 = 8 < x$

$i{+}{+}$

$a[i] + a[j] = 3 + 6 = 9 = x$

Q. Given a sorted array, find a pair closest to the sum X.

Eg: 1 3 8 15 19 25

X = 24

1 3 8 15 19 25

i j

X = 24

closest_pair = 23

1

if (abs(sum - X) < abs (closest_pair - X))

2

⇒ closest_pair = sum;

$a[i] + a[j] = 1 + 25 = 26 > X$

j--;

그리고 $a[i] + a[j] = 1 + 19 = 20 < X$

i++;

$a[i] + a[j] = 3 + 19 = 22 < X$

i++;

$a[i] + a[j] = 8 + 19 = 27 > X$

j--;

$a[i] + a[j] = 8 + 15 = 23 < X$

i++;

n = a.size()

a = 2 3 1 3 2 4 6 7 9 2 8

b = 2 1 4 3 9 6

freq[b[i]]

freq:

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

- store the freq of a[]

Q. the elements of a[] which do not exist in b[] be placed in descending order.

V =

2	2	2	1	4	3	3	9	6	7	8

- iterate on b[] and append the freq. of b[i] to v.

1 4 3 2

min → 1

(1,4) → 1

(1,3) → 1

(1,2) → 1

(1,4) + (2,3) → 2 = 3

(1,2) - (3,4)

1 + 3 → 4

a = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~

(1,2) (3,4) (5,6) (7,8)

1 3 5 7

$a_1, a_2, a_3, a_4, \ldots, a_n$

if they are sorted in non-decreasing order

pairs would be: $(a_1\ a_2)\ (a_3\ a_4)\ (a_5\ a_6)\ (a_7\ a_8) \ldots$

∴ bubble sort

∴ insertion sort

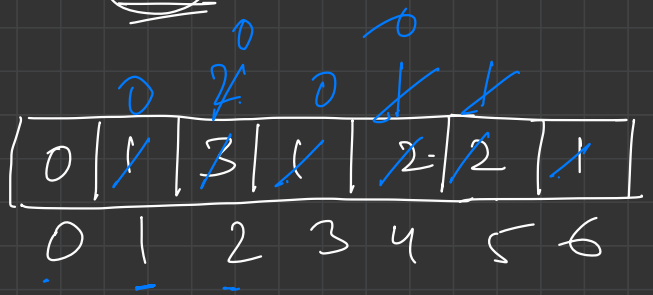
∴ selection sort

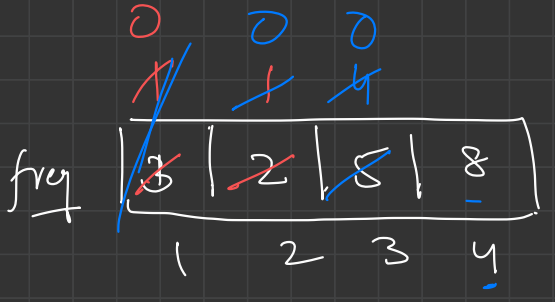
Frequency array ??

a = 1 2 2 2 3 4 4 5 5 6

(1 2) (2 2) (3 4) (4 5) (5 6)

1 + 2 + 3 + 4 + 5





(1, 1) (2, 3)

(1, 2) (3, 3) (3, 3)

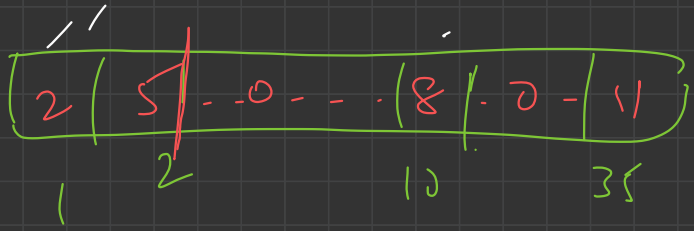
(4, 4) (4, 4)

find max. sum of min. of pairs : 1 + 2 + 3 + 4 + 5 (4, 4) (4, 4)

$4 \xrightarrow{freq} 8$

↳ pairs $\Rightarrow$ sum

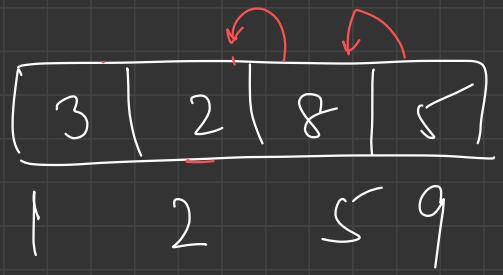
$4 * \frac{freq[4]}{2}$



(1, 1) + (2, 2) + (2, 2) + (2, _

```
bool left_over = false;

for(int i = 0; i < 2*n; i++){
    if(freq[i] > 0){
        sum += i*((freq[i]-left_over+1)/2);
        left_over = (freq[i]+left_over)%2;
    }
}
```



(1, 1) → 1

(1,) → 1

5

0 → (0,0) (0,0) (0,

freq[i]

sum += 1*((3-0+1)/2)

+= (4/2) = 2

freq[2] = 2

sum += 2*((2-1+1)/2)

+= 2

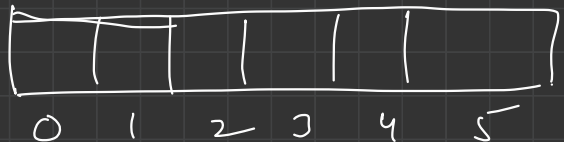
freq[5] = 8

sum += 5*((8-1+1)/2) = 5(4) = 20

freq[9] = 5

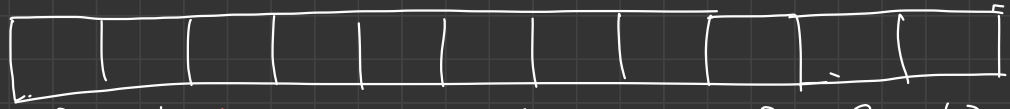
sum += 9 * (15 - 1 + 1) / 2) = 9 (5/2) = 9(2) = 18

0 → 5



-5 → 5

values → 11



i → freq[i]

i → freq[i - shift]

freq of 0 → freq[0] X (-5)

↳ freq[0+5]

shift = 5

freq of -3 → freq[-3] X

↳ freq[-3+5] = freq[2]

Q. Find the closest triplet to X in a sorted array.

Q. Given 3 sorted arrays, find three elements one from each such that they are closest.

(A, B, C) → minimize (max(|A-B|, |B-C|, |C-A|))

Binary Search

1. Aggressive Cows
2. Painter Partition
3. Murthal & Paratha

2, 3, 5 → (2, 2, 3) → 3

3, 4, 2 → (1, 2, 1) → 2