WELCOME!!

2 POINTERS

1.) Different approaches

Intution → Pseudo code → Dryrun

90 - 100 mins

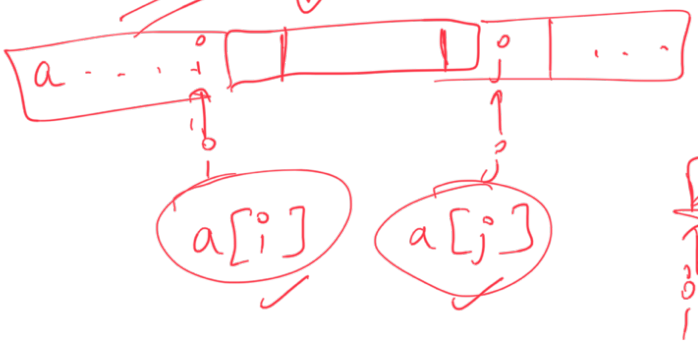
DOUBT

2 POINTERS

Technique

DS — Stack, Queue, Heap, BT..

AI — FW, Sorting, Binary Searc



Array → SORTED → +ve numbers

Ex: [1, 4, 5, 6] → K = 9

↑ ↑

1

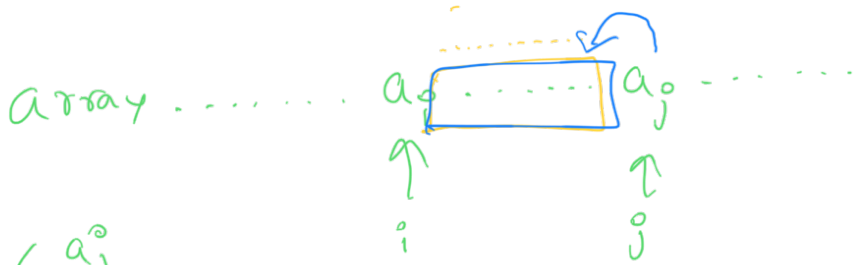
↳ Count of pairs that sum up K

B.F

array → a → size = n

```
for( i → 0 to n-1) {
    for (j → i+1 to n-1) {
        x = a[i]
        y = a[j]
        if(x+y == K) → count++
    }
}
return count
```

$O(N^2)$



$a_i < a_j$

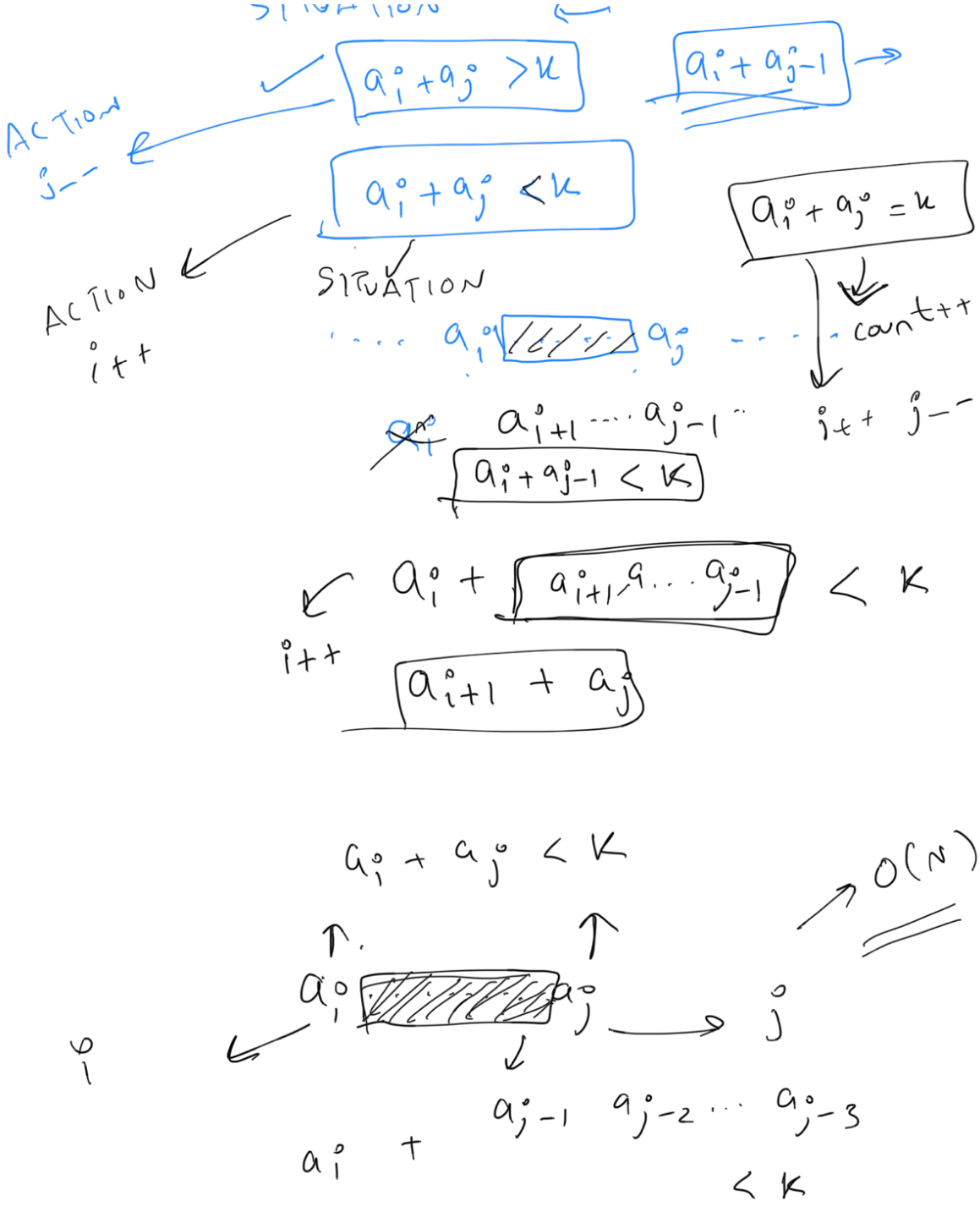
$a_i + a_j > K$

$a_{i+1} + a_j > K$

$a_{i+1}, a_{i+2} \cdots a_{j-1}$

i → 0

j → n-1


SITUATION
$a_i + a_j > k$
ACTION
j--
$a_i + a_{j-1}$
$a_i + a_j < k$
ACTION
i++
SITUATION
a_i ... a_j
$a_i + a_j = k$
count++
$a_{i+1} \cdots a_{j-1}$
i++ j--
$a_i + a_{j-1} < k$
$a_i + a_{i+1} \cdots a_{j-1} < k$
i++
$a_{i+1} + a_j$
$a_i + a_j < k$
O(N)
a_i
a_j
j
i
a_{j-1} a_{j-2} ... a_{j-3}
a_i +
< k
Dry run
1 4 5 6
k = 9
$O(n^2)$
i i j j
$a_i + a_j = 7$
$a_i + a_j$

O(N)

10 > 9

$a_i + a_j = 9$

count = 1

Array → a

i → 0

j → n-1

while (i < j) {

x = a[i]

y = a[j]

if (x+y > k) j--

elseif (x+y < k) i++

else { count++ i++ j-- }

}

count = 0

O(N)

Elements are not distinct

Example array a → [1, 4, 4', 5, 5', 5'', 6, 6', 11] k = 10

(4, 6) (4, 6')

(4', 6) (4', 6')

7

4

(5 5') (5' 5'') (5 5'') →

3

$a_i + a_j > k$ $a_i + a_j < k$

O(N)

LAND PROBLEM!

O(1)

$a_i + a_j == k$

count i, j, count

O(1)

[1 4 4 5 5 5 6 6 11]

2 2

k = 10

3

i j

$a_i + a_j = 10$

count(4's) = x

count(6's) = y

2 2

x * y → count += x * y

555

i j → x = count(5's)

3

y = 3

x * y = 9

$a_i + a_j = k$

$a_i \neq a_j$ → $a_i a_i a_i \ldots a_j a_j a_j a_j$

x = 3 y = 4

count += x * y

i → incr x

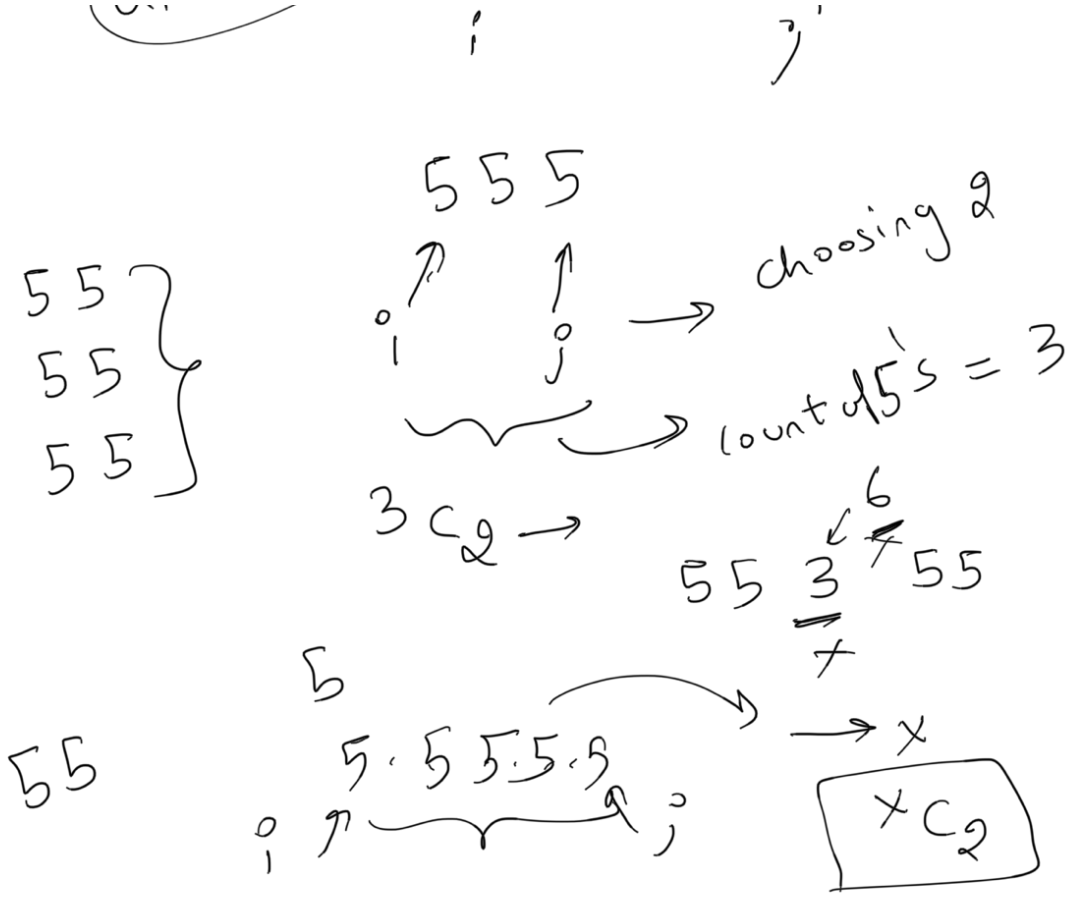
j → decr y

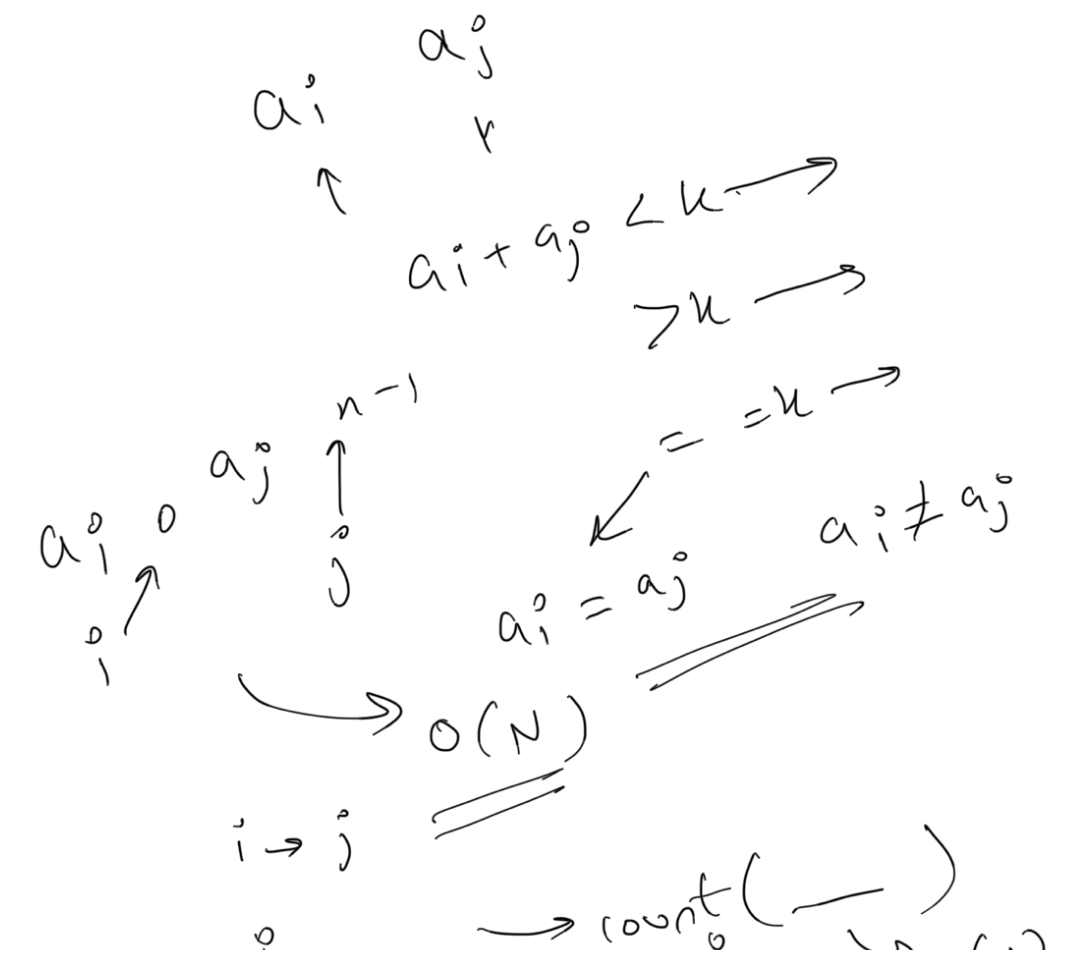
i j

array

$a_i == a_j$

0 n-1


i
j
5 5 5
choosing 2
count of 5's = 3
3 C 2 →
55 3 55
55
5
5 . 5 5 5 . 5
x
x C 2


a i
a j
r
a i + a j < r →
> r →
= = r →
a i ≠ a j
a i = a j
n - 1
O(N)
i → j
count(—)

555555 → 10

count( ) count( )

↳ 6 O(N) $a_i + a_j = 10$ → $5+j$

↳ i → j →

SORTED ARRAY

↳ +ve numbers ↳ K 0 to n-1

array → a

subarray [i...j] →

true or false ↳ sums up to K

[1 2 3 3 4 5 6] K = 10

B.F

↳ Find out all sub arrays

i j

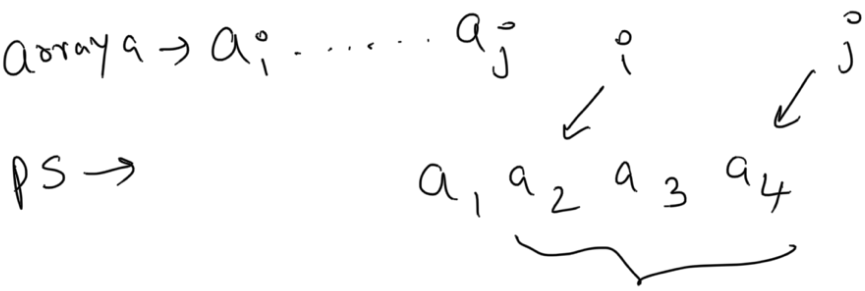
0,0 1 0 2 ..... n-2, n-1

B.F → T.C → $O(N^3)$ $O(N^2)$ → $[a_i \dots a_j]$ ↓ is summing to K

↳ O(N)

i,j → O(N²) (i, j) → O(N)

array a → a_i ..... a_j



ps[0] = a_1

ps[1] = $a_1 + a_2$

ps[2] = $a_1 + a_2 + a_3$

ps[3] = $a_1 + a_2 + a_3 + a_4$

ps[3] - ps[0]

array a → a_1 a_2 ... a_n

prefixsum ps → p_1 p_2 ... p_n

for (a → a_1 to a_n)
sum += a
ps[index] = sum
index++;

sum = 0

→ O(N)

→ O(N)

Extra space.

B.F → O(N³)

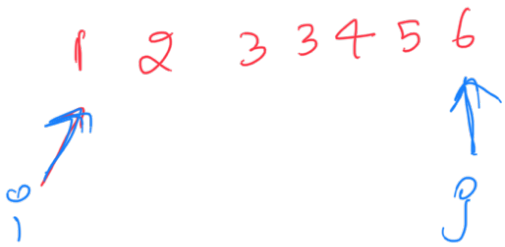
P.S. B.F →

i, j → O(N²)

sum [i ... j] O(N)

→ O(1)

T.C O(N²) S.C O(N)



@ Sum = $a_1 + a_2 + \cdots a_n$

O(N) ← Sum

if(sum < k)

i → 0 j → 0 .... n-1 → compute sum

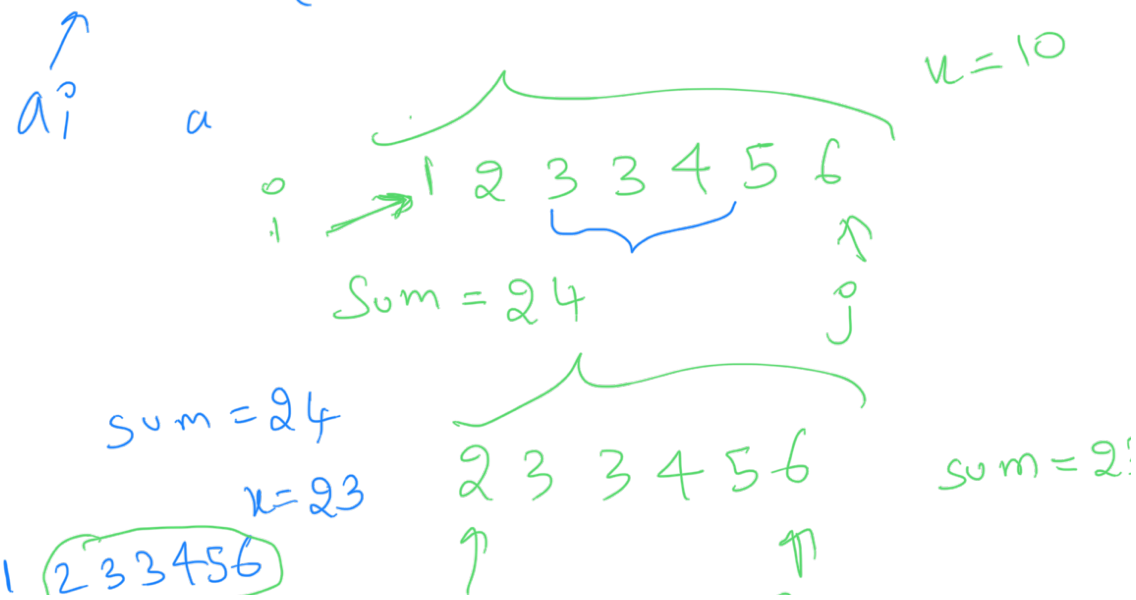
Sum.

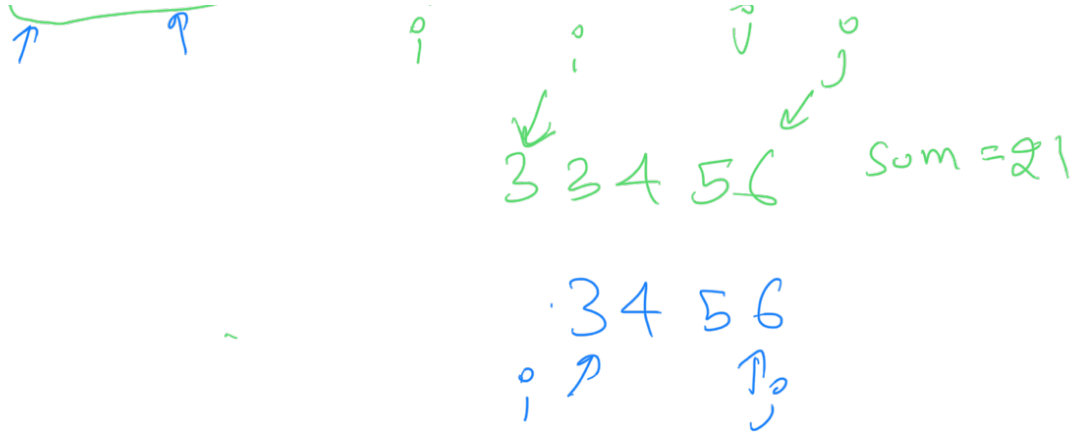
i → 0 j → n-1

if (sum < k) return false;

if(sum == k) return true;

if (sum > k)





SORTED

1 2 3 3 4 5 6 $k = 10$

→ 1 sum = 1 $k = 10$

$k = 10$

1 2 sum = 3

Sum = 6

sum = 13 → Sum = 12 → sum = 10

↓ O(N)

10 ← Reached optimal

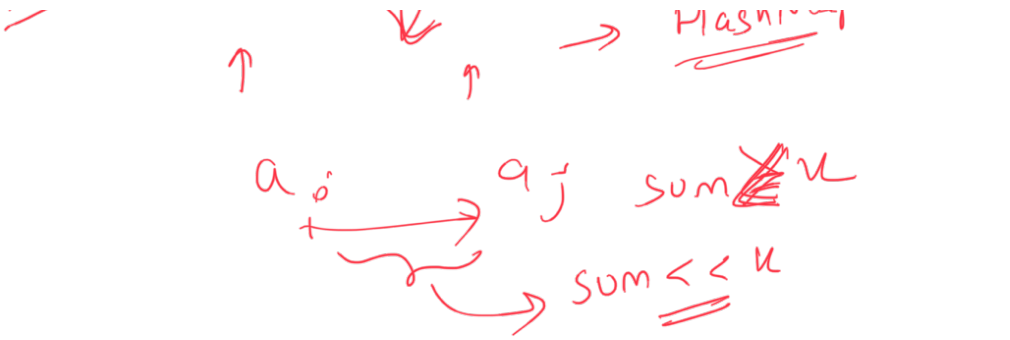
T.C S.C

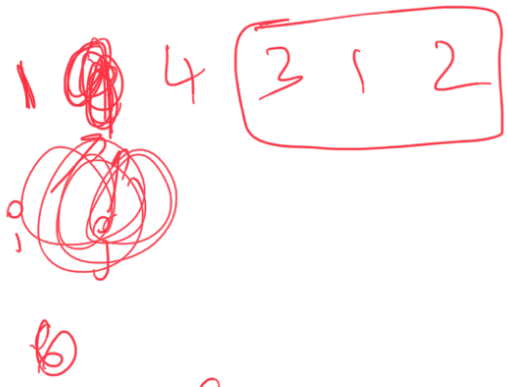
B.F $O(N^3)$ $O(1)$ → true

P.F $O(N^2)$ → $O(N)$

$O(N)$ $O(1)$

2 pts O.A





sum = 6

sum = 9

if (sum > k)

if (sum < k) → return false

Problem

String → characters

a, b

a a b a a b a

beauty (string) → number of contiguous same characters

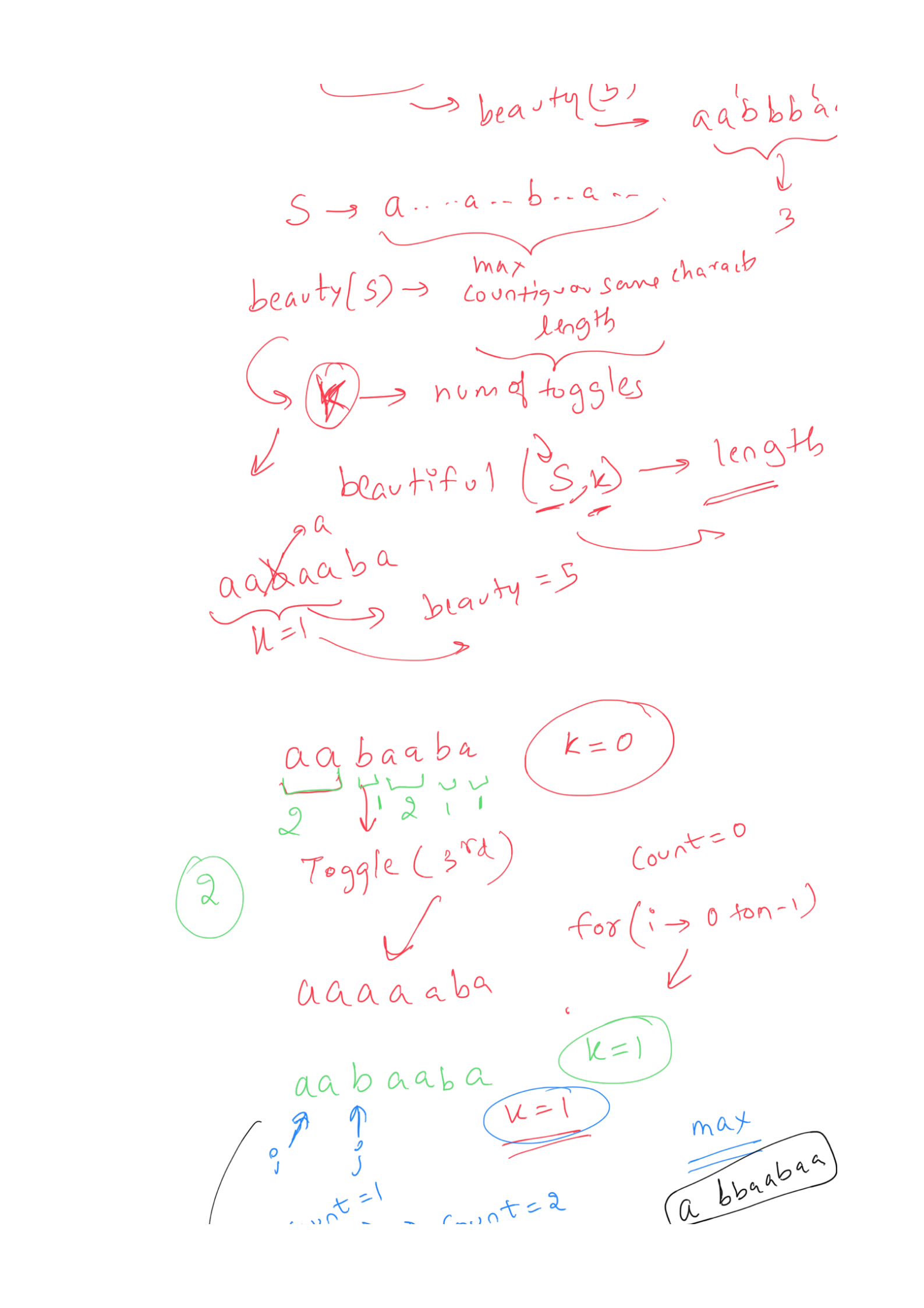
k = 0

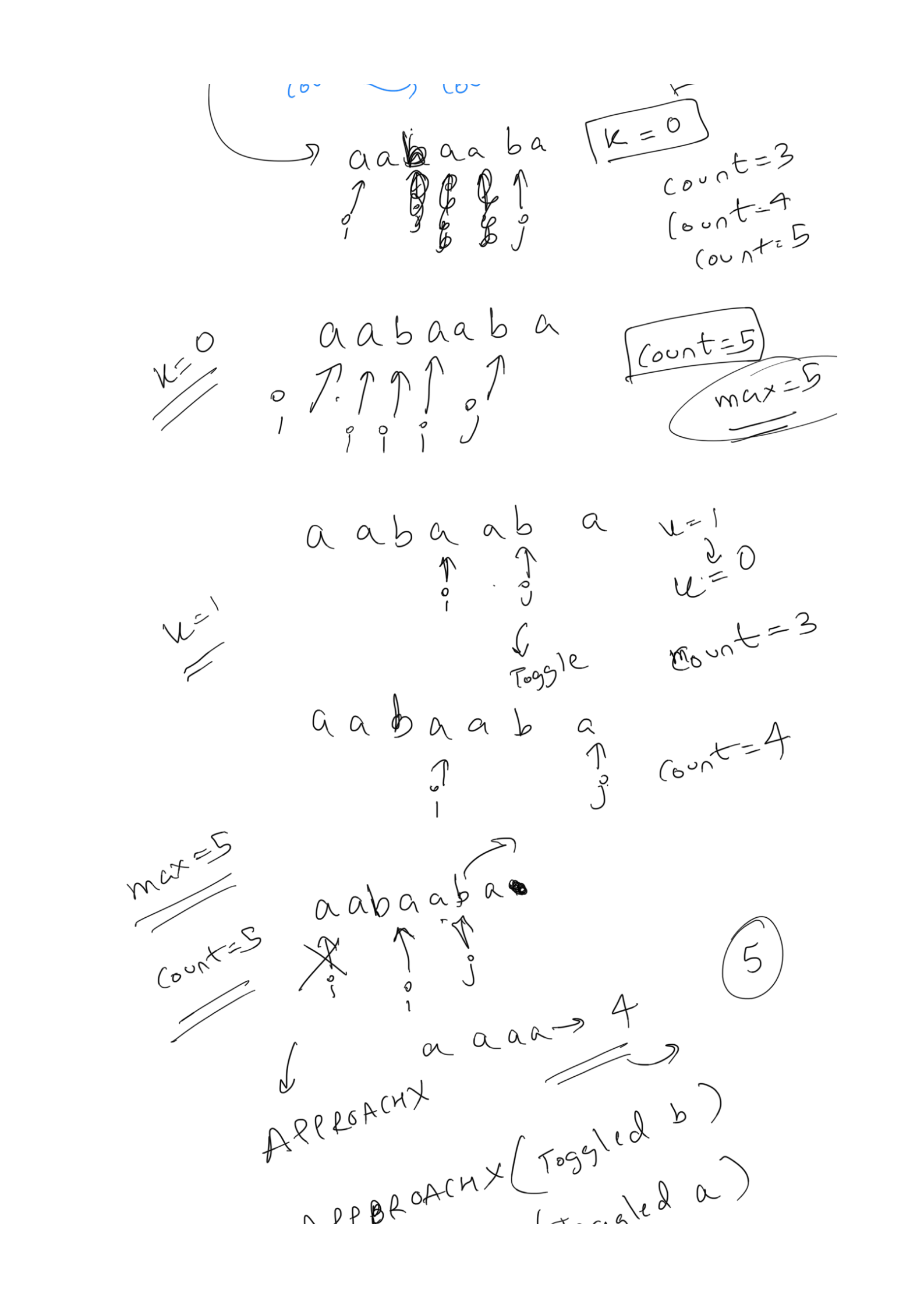
beauty (— 2)

Max bea

k = 1

beauty → k numbers toggle


beauty(b)
aabbba
3
S → a...a--b--a--
beauty(S) → max contiguous same charac length
num of toggles
beautiful (S, k) → length
aabaaaba
a
k=1
beauty = 5
aabaaaba
k=0
2
2 1 2 1 1
Toggle (3rd)
aaaaaaba
count=0
for (i → 0 to n-1)
k=1
aabaaaba
k=1
i
j
count=1
count=2
max
abbaabaa


a a b a a b a
K = 0
count = 3
count = 4
count = 5
a a b a a b a
K = 0
count = 5
max = 5
a a b a a b a
K = 1
K = 0
Toggle
count = 3
a a b a a b a
count = 4
max = 5
a a b a a b a
Count = 5
5
a a a a → 4
APPROACH X
APPROACH X (Toggled b)
(Toggled a)

x1 = APPROACH (log)

x2 = APPROACH x (log)

k=1

max (x1, x2) → O(N)

String S

a ... a b a

abbaabaa

random x

k=0

k=1

k = Toggling - 1 ree

aaabababbaa

k=2

O(N)

aaabaaabbaa

x1 = APPROACH (toggle b)

x2 = APPROACH (k..a)

max(x1, x2)

Pseudo code

DOUBTS → P1
P2

String - S. K → K a ... b.

beauty (S, K)

```
beauty ( string S, int K) {
x1 = beauty util (S, K, a)
x2 = beauty util ( S, K, b)
return max (x1, x2).
}
```

beauty util (S, K, c)

S → array →
i → 0
j → 0
toggles_used → 0
for(j - ...) {

count
max

if (array[ ] equal to c)

↓

count++

↓

if (array[j] != c)

↓

if (toggle_used < k)

count++ ; toggle_used++

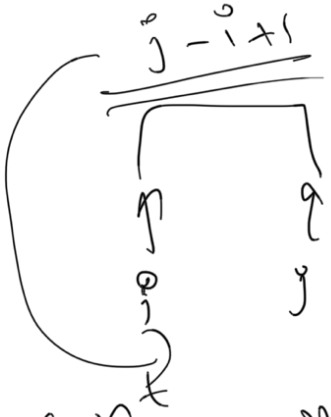
if (toggle_used == k)

↓

increment (i to the first)

↓

updating the max value



i i j

a a b a a b a b b b

k=1

a a a b a a a b

q i j

flipped (b → a) → max = 5

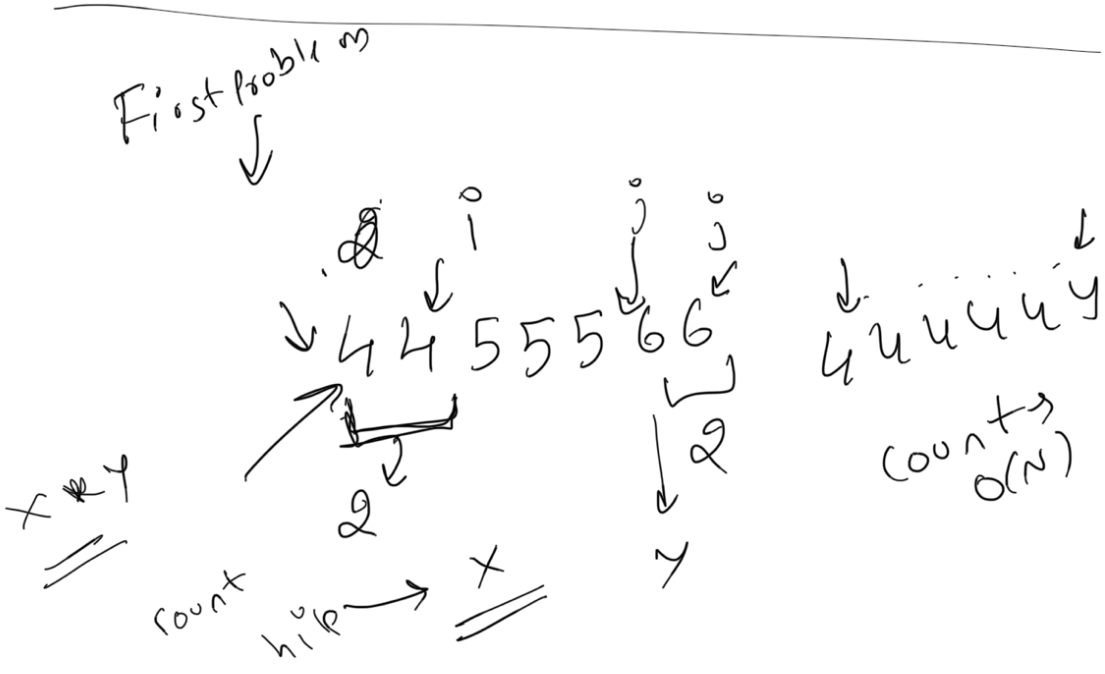
flipped (a → b) → max = 2

↓

max = 6

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


i
j
Linear


First problem
i
j
j
4 4 5 5 5 6 6
4 4 4 4 4 4
count
O(N)
x * y
2
2
y
count
hip
x