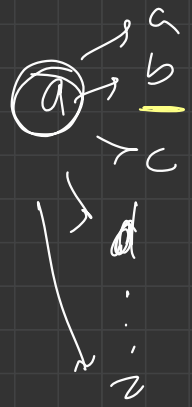
- max chunks to sorted
- largest time
- remove adjacent duplicates (2)

---

we change k - chars (any)



128 chars

babb

k = 1

b → c, d, e, f, z, q

a → b or b → a

a b a b

k = 2

b b b b

a a a a

b a a b

a a b b

Same chars as much as possible

option 1 change only a → b

option 2 change only b → a

option 3

Change some $a \rightarrow b$

some $b \rightarrow a$

$k = 2$

b a b a b a b b c a b

option 1 $a \rightarrow b$

a a b a b b b b b b (6)

option 2 $b \rightarrow a$

b a a a a a b b a b. (5)

✓ we will make a function which finds the maximum length substring of same chars, with k changes of char x possible.

max_len = 0
count = 3

b a b a b a b b a a a b

i ↑ ↑ i

x = 'a' → we should change any char not 'a'

k = 2

we're finding max len substring of all 'a'.

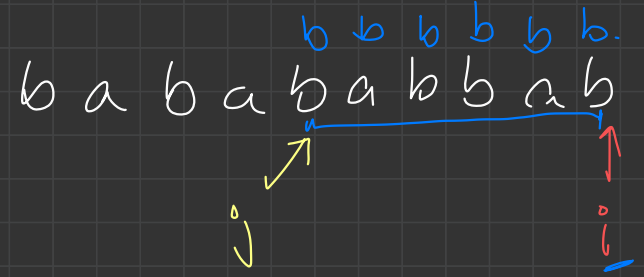
s[i] == x ? → false : change it (count++).

true : no issue → go ahead

ch = 'b' k = 2 count = 2

i - j + 1

max_len = 6



s[i] == 'b' ? → true : go ahead (calculate len)

→ false : count++; check (count > k) len calc

Problem

① understand
- what to do
- output
- constraints : complexity

→ decide on approach

② write, formalize (pen-paper)

subarray : $n = 100$

$O(n^2) = 10^4$

$O(n^3) = 10^6$

$O(n^4) = 10^8$

brute-force ??

approach
- brute-force
- efficient

③ Implement in C++

- it works
- error.

④ Debugging

- read the error.
- dry run.

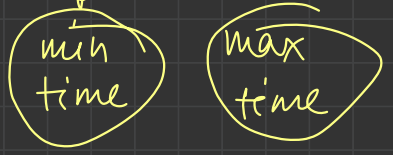
- when found a working approach
  - ① dry with sample case
  - ② try with some other TC

- Aggressive Cows
- Painter Problem
- Murthal Paratha

range: continuous

optimum output -

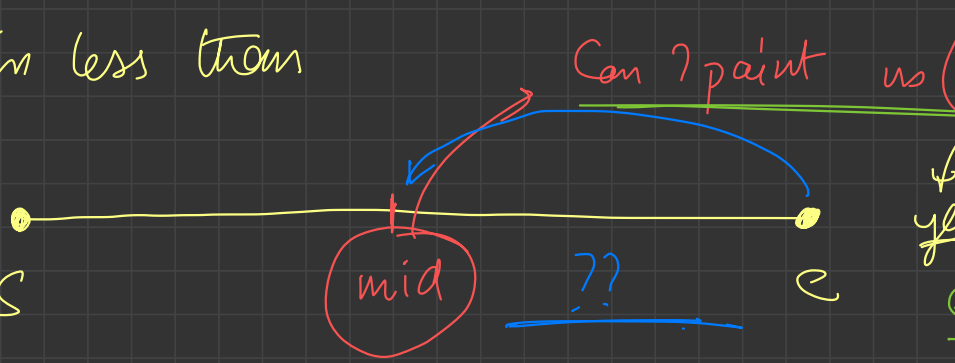
s - e identify



0 10^9

Q. if I can paint all paintings in T time. Can i do it in more than T time?

Q. If I cannot do it in T time. Can I paint in less than T time?



Can I paint in mid time? ⇒ no

s = mid+1


```
while(s <= e) {
    mid = (s+e)/2;
    if(check()) {
        e = mid-1;
    } else { s = mid+1; }
}
```

Q. Given an array, print another array b., where

$$b[i] = \sum_{j=0}^{i-1} a[j]$$

a = 1 2 3 4

output b = 0 1 3 6

Q. $b[i] = \sum_{j=i+1}^{n-1} a[j]$ ⇒ suffix

Q. cummulative sum ⇒ $b[i] = \sum_{j=0}^{i} a[j]$

Prefix sum / count / any related value

wrt. all elements before it

Suffix any value wrt all elements after it

Cummulative