



BINARY SUBARRAY WITH SUM = K

$$arr = [1, 0, 1, 0, 1] \quad goal = 2$$

output = 4

[1 0 1 0 1]



problem $[0, 0, 1, 0, 1]$ goal = 2
 wanting subarray like this

we would optimize by not using any space as we have done previously in count subarray problem

$$\underline{\text{answer}} = f(\text{num}, \text{goal}) - f(\text{num}, \text{goal}-1)$$

no of array ≤ 3 and

$$\text{sub_array} \leq \text{goal} - 1$$
~~$$f_{\text{num}, \text{goal}} \{ l=0, u=0, s \}$$~~~~int sum = 0, cnt = 0;~~~~while (i <= nums.length) {~~

```
sum = sum - names[l];  
l++;
```

3

fun (nums, goal) {
if (goal < 0) return;
if (goal == 0) return;
if (goal > 0) {
 then goal - 1 = -1
 0 - 1 = -1
}

while (r <= nums.length) {

sum += nums[r];

while (sum > goal) {

sum = sum - nums[l];

l = l + 1;

}

cnt = cnt + (r - l + 1);

r = r + 1;

}

return cnt;

}

TC $\rightarrow O(4N)$

$O(n)$

$O(n)$

LONGEST REPEATING CHARACTER

REPLACEMENT

S = A A B A B B A

(K=2)

can change only 1 character to any other english character

4
A A A A B B A

A A B B B B B 5 longest

∴ tell the longest repeating character sequence

BRUTE

A A B A B B A

K=2

make sequence starting from every char.
maxlen = 0;

for (i = 0 → n) {

hash[26] = [0], max frequency = 0

for (j = i → n) {
hash[s[j]]++;

updating frequency

hash[s[j]]++;

maxf = max(maxf, hash[s[j]]);

changes = (j - i + 1) - maxf; changes can be made
if (changes ≤ K) {
maxlen = max(maxlen, (j - i + 1));

else break; can't do the changes as > K

return maxlen;

A - 3

B - 2

we will update
5 - 3 = 2

len max freq

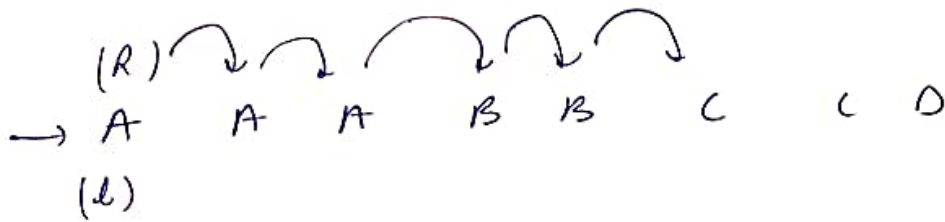
no changes required

storing freq of max appearing char so that rest can be changed and minimum changes will be required

no of changes reqd

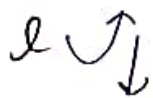
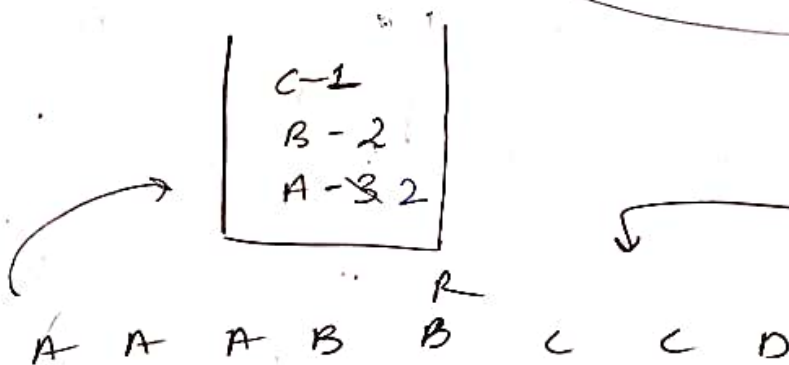
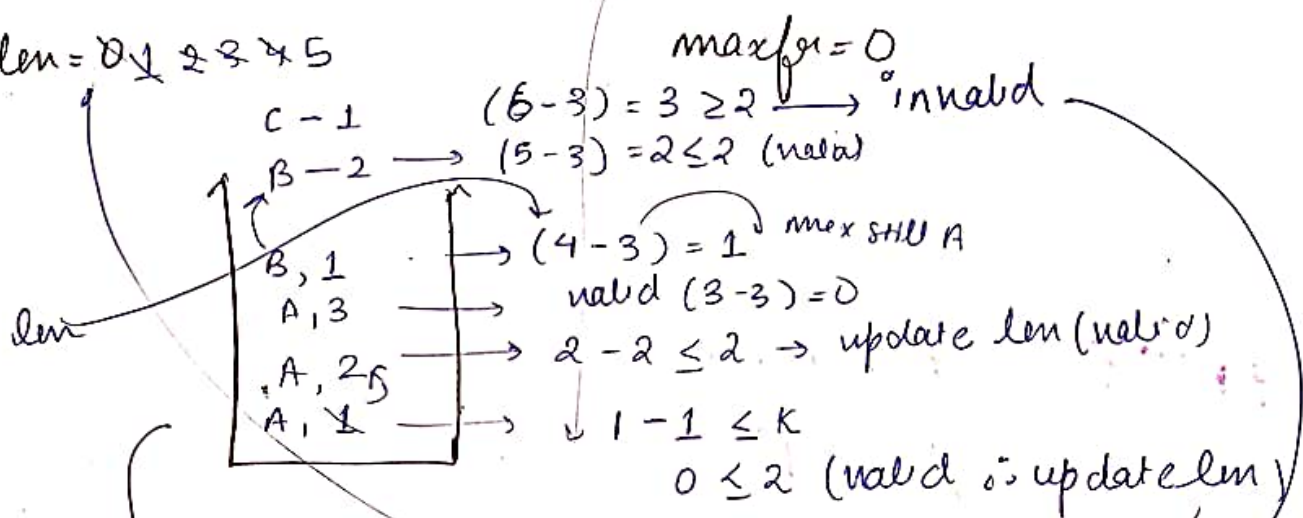
OPTIMAL:

$S = A A A B B C C D$ $K = 2$



$$\text{len} - \text{maxfreq} \leq K$$

maxlen = 0 1 2 3 4 5



$= 4 - 2 = 2 \leq 2$ valid

\rightarrow Similarly

decrease
increase
i till
(valid)
again

fun(s, k) {

l = 0, r = 0, maxlen = 0, maxf = 0;

hash(26) = {0};

$O(n)$

while (r < s.length) {

hash[s[r] - 'A']++;

maxf = max(maxf, hash[s[r] - 'A']);

\swarrow in val d

while ((r - l + 1) - maxf > k) {

\swarrow length

hash[s[l] - 'A']--;

maxf = 0;

updating max frequency;

for (i = 0 -> 25) maxf = max(maxf, hash[i]);

l = l + 1;

}

if ((r - l + 1) - maxf <= k) {

maxlen = max(maxlen, (r - l + 1));

}

r++;

}

return maxlen;

$\therefore O(2n \times 26)$

→ MINIMUM COINS

coins = {25, 10, 5} & Sum = 30

denomination

(infinite supply of each coin)

you have to pick minimum coins to get sum (30)

Greedy approach

coins = {1, 2, 5, 10, 20, 50, 100, 500, 1000}

Start from last

sum → $V = 49$

sum

now $1000 > 49$
∴ shift
 $500 > 49$
 $100 > 49$
 $50 > 49$
but

$20 < 49$

use denomination repeatedly $49 - 20 = 29 - 20 = 9$

now $20 > 9$ ∴ shift

$10 > 9$ shift

$5 < 9$ ∴ $9 - 5 = 4$

similarly 2, two time

2
2
5
20
20

answer
count

This approach only works when...

[1, 2, 5, 10, 20, 50, 100, 500, 1000]

sum of two wins numbers exceeds the next one

$$2 + 5 < 10$$

as if array = {1, 5, 6, 9}

ans [sum = 11]

↓
 $11 - 9 = 2$ not possible
but $5 + 6$ possible

when sum becomes greater the next
on $5 + 6 > 9$ ∴ we can't say be sure
if to pick 9 or not
but if we can't previous can
make the sum we can be
sure