BINARY SUBARRAY WITH SUM = K

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

output = 4 [1 0 1 0 1]

problem [0 0 0 1 0 1] goal = 2

counting subarray like this

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

answer = f(nums, goal) − f(nums, goal − 1)

no of array ≤ goal sub array ≤ goal − 1

```
fun (nums, goal) {
    l = 0, r = 0, sum = 0, cnt = 0;
    while (r <= nums.length) {
        sum = sum - nums[l];
        l++;
    }
```

```
fun (nums, goal) {
    if (goal < 0) return;
```

when goal = 0

then goal - 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;
}
```

O(n)

O(n)

TC → O(4N)

LONGEST REPEATING CHARACTER REPLACEMENT

S = A A B A B B A (K=2)

2 can change any to character to any other english character

A A A A B B A 4

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 the sequence starting from every char.

```
maxlen=0;
for (i=0 ⟶ n) {
    hash[26] = [0] , maxfrequency=0
    for (j=i ⟶ n) {
        hash[s[j]]++;
        maxf = max(maxf, hash[s[j]]);
        changes = (j-i+1) - maxf;
        if (changes ≤ K) {
            maxlen = max(maxlen, (j-i+1))
        }
        else break;
    }
return maxlen;
}
```

A - 3
B - 2

updating frequency

not we will update

5-3 = 2

len max freq

no of changes required

changes can be made

can't do the changes as ↑ > K

-'A'

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

no of changes

OPTIMAL:

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

(R)

→ A A A B B C C D

(l)

len - maxfqr <= K

maxlen = 0 1 2 3 4 5

maxfqr = 0

C - 1

B - 2 → (5 - 3) = 2 ≤ 2 (valid)

(6 - 3) = 3 ≥ 2 → invalid

B, 1 → (4 - 3) = 1 max SHU A

A, 3 → valid (3 - 3) = 0

A, 2 → 2 - 2 ≤ 2 → update len (valid)

A, 1 → 1 - 1 ≤ K

0 ≤ 2 (valid ∴ update len)

len

C - 1

B - 2

A - ~~3~~ 2

decrease increase till (valid) again

R

A A A B B C C D

l

= 4 - 2 = 2 ≤ 2 valid

→ Similarly

```
fn (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']);
                                in valid
    while ((r-l+1) - maxf > k) {
        length      hash[s[l] - 'A'] --;
                                    { updating max frequency
        maxf = 0;
O(n)    for (i=0 → 25) maxf = max(maxf, hash[i]);
            l = l+1;
O(26)   }
    if ((r-l+1) - maxf <= k) {
        maxlen = max(maxlen, (r-l+1));
        }
    r+1;
}
return maxlen;
```

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

→ MINIMUM COINS

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

demonnination
(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 → $\boxed{V = 49}$

Now 1000 > 49
∴ shift
500 > 49
100 > 49
50 > 49
but

20 < 49

use demonination 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 coins numbers exceeds the next one

$2+5 < 10$

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

ans $\boxed{\text{sum} = 11}$

$11-9 = 3$ 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 no coins previous can make the sum we can be sure