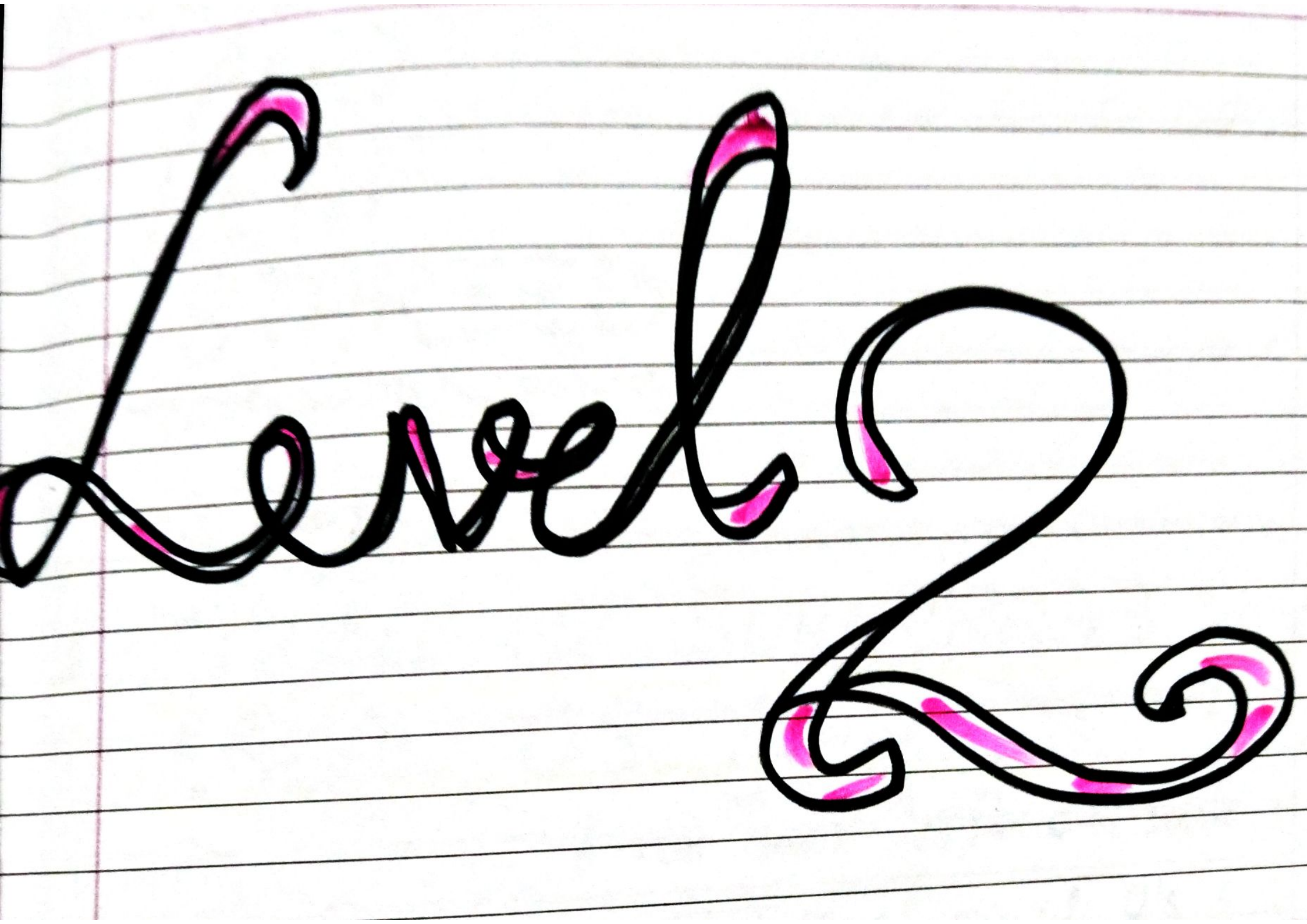

Level 2

HASHMAP & HEAP

HashMap & Heap Questions

1. Count of Subarrays with equal 0's & 1's. → gfg
2. Contiguous Subarray (Longest subarray with equal 0's & 1's). → gfg & leetcode

3. Count of Subarray with equal 0, 1, 2. → gfg.
4. Longest Subarray with equal 0, 1, 2.
   ↳ ~~leetcoding~~
5. Number of Subarrays sum exactly K.
   ↳ gfg
6. Longest Sub array with Sum K. → gfg.
7. Subarray Sum divisible by K. → gfg.
8. Longest Subarray with Sum divisible by K.
   leetcode, gfg.

Day 157

Level 2

HASHMAP AND HEAP STARTED

Date
Page No.

— Tamanna Verma

Date → 11 March.

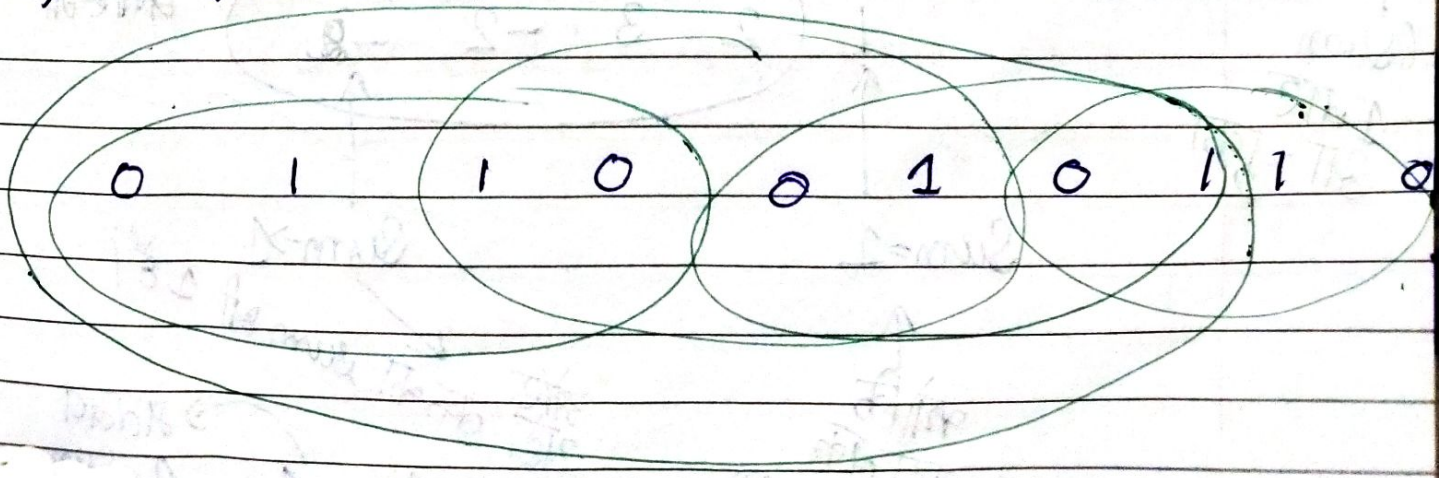
1. COUNT OF SUBARRAYS WITH EQUAL 0's AND 1's. → lyfg

→ subarrays are continuous

एक array given है। हमे बताना होगा कि ऐसे कितने subarrays हैं जिनमे equal no. of 0's & 1's हैं।

किसी भी array के $\frac{n(n+1)}{2}$ subarrays होते हैं।

→ O(n) में ये problem solve करनी है।



के लिए
0 → -1 add करो
1 → 1 add करो

0	1	1	1	0	0	1	0	1	1	0
Sum 0	-1	0	1	2	1	0	1	0	1	2	1

→ एक Sum variable को update करते चलो और हर बार जो Sum की value आएगी, उसको Hashmap में add करो as a key और उसकी value होगी frequency of this sum.

Sum

0	1	1	1	0	0	1	0	1	1	0
0	-1	0	1	2	1	0	1	0	1	2	1

0 के लिए -1 add किया
+1 add किया
+1 add किया
+1 add किया
+1 add किया
-1 add किया
+1 add किया
-1 add किया
+1 add किया
+1 add किया
+1 add किया

0→1 0→2 1→1 2→2 1→2 0→3 1→3 0→4 1→4 2→3 1→5

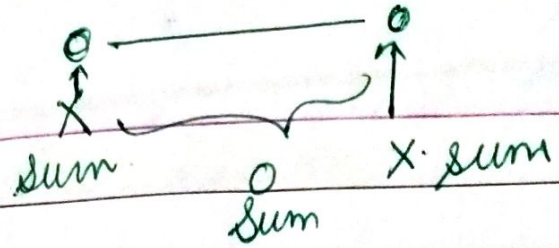
0 Sum 1 बार आ चुका

1 2 3 −2 −3 → ये Subarray भी 0 sum बनाएगा

Sum=1 क्योंकि यहाँ तक का sum भी 1 है

Sum→1 और यहाँ तक का sum भी 1 है i

1→1 मतलब 1 Sum बनाने वाला 1 Subarray है



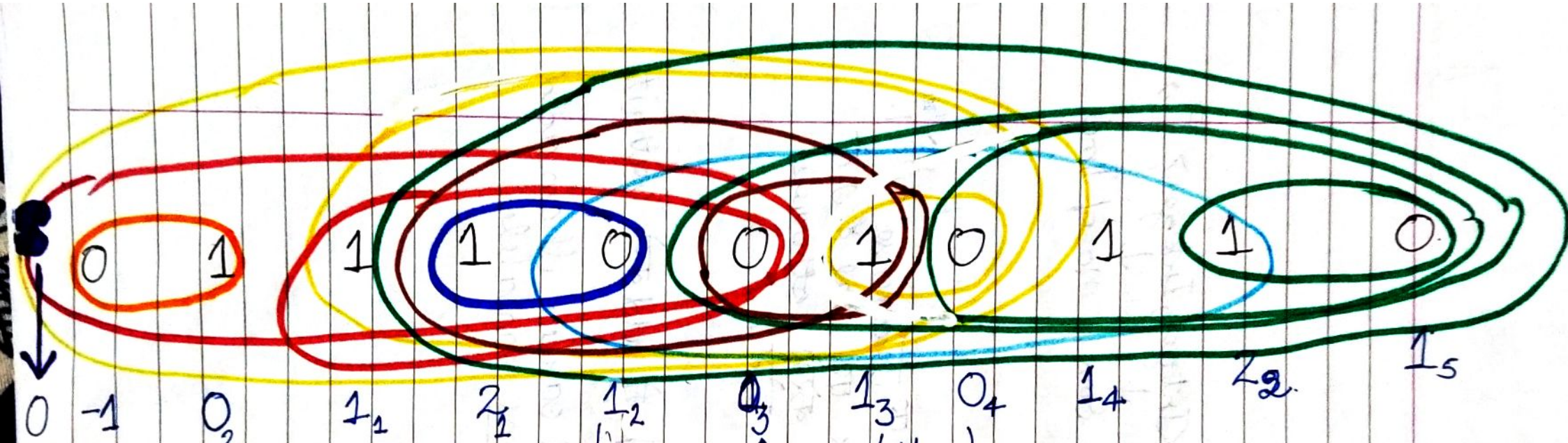
Approach:

① सबसे पहले एक Hashmap बनालो <Integer, Integer> का,
↓ ↓
Sum No. of Times.

② उसमे 0 → 1 डाललो
This denotes that
0 sum 1
बार आया है.

③ फिर हम पूरे array पर traverse करेंगे।
अगर array मे ith index पर val है 0 तो
sum में -1 add करेंगे otherwise
sum में +1 add करेंगे।

④ जो भी sum भी value आएगी
उसको Hashmap में add करो with its occurence.
If the sum has occured earlier than add the earlier occurence in ans variable.



(2-1) = 1.
add 1 in ans
because 1 subarray

2-1 = 1
add 1 in ans

add 2

add 2

add 3.

$1 + 1 + 2 + 2 + 3 + 3 + 3 + 1 + 4$

$= 2 + 4 + 6 + 4 + 4$

$= 6 + 6 + 8$

$= 20$

```java
public class Solution
{
  static int countSubArrWithEqualZeroAndOne
            (int arr[], int n)
  {
    int ans=0;
    HashMap<Integer, Integer> map=new HashMap<>();
    map.put(0,1);      // initially 0 sum है जो कहीं नहीं add किया

    int sum=0;         // Initially sum 0 रहेगा
    for(int val: arr)  // array की हर value loop चलेगी
    {
      if(val==0)
      {
        sum+=-1;       // अगर val 0 है तो -1 add करदो otherwise 1 add करदो
      }
      else{
        sum+=1;
      }

      if(map.containsKey(sum))   // अगर Hashmap में sum already है तो value get करो और ans में add करदो
      {
        int x=map.get(sum);
        ans+=x;
        map.put(sum,x+1);        // और फिर Hashmap में इस sum की value +1 करदो
      }
      else{
        map.put(sum,1);          // otherwise इस sum को Hashmap में डालदो with value 1.
      }
    }
    return ans;
  }
}
```

leetcode 525

lyfg

Date
Page No.

2. Longest Subarray with Equal 0 & 1.

	0	0	1	0	1	0	1	1	0	0	1	1	1
	0	1	2	3	4	5	6	7	8	9	10	11	12
Sum	-1	-2	-1	-2	-1	-2	-1	0	-1	-2	-1	0	1

Hashmap में sum और उसके index add करो
Then when again the same sum occur again, then chuck the difference in their indexes.

→ जब सबसे पहले कोई sum आए तो उसे add करदो Hashmap में, with its current index & when the same sum occur again, just check the differences in their index & that will be there length of subarray

→. Keep on updating the length of max subarray

```
class Solution
{
    public int findMaxLength(int[]arr)
    {
        int sum=0;
        int MaxLength=0;
        HashMap<Integer, Integer> map = new HashMap<>();
        map.put(0,-1);

        for(int i=0; i<arr.length ;i++)
        {
            int val=arr[i];
            if (val==0)
            {   sum += -1;
            }
            else{
                sum += 1;
            }
            if(map.containsKey(sum))
            {
                int x= map.get(sum);
                int currentLength=i-x;
                if(currentLength > MaxLength)
                {  MaxLength =currentLength;
                }
            }
            else
            {map.put(sum,i);
            }
        }
        return MaxLength;
    }
}
```

→ sum की value initially 0 है (i.e 0 sum has occured at -1 index).

initially Maximum length of subarray = 0

एक हमें एक Hashmap लेना।

हम array की सभी values पर loop लगाएंगे, अगर value 0 है तो sum में -1 add होगा otherwise sum में +1 add होगा

अगर ये sum already hashmap में है तो check the previous index of this sum

current index − previous index होगा subarray की length

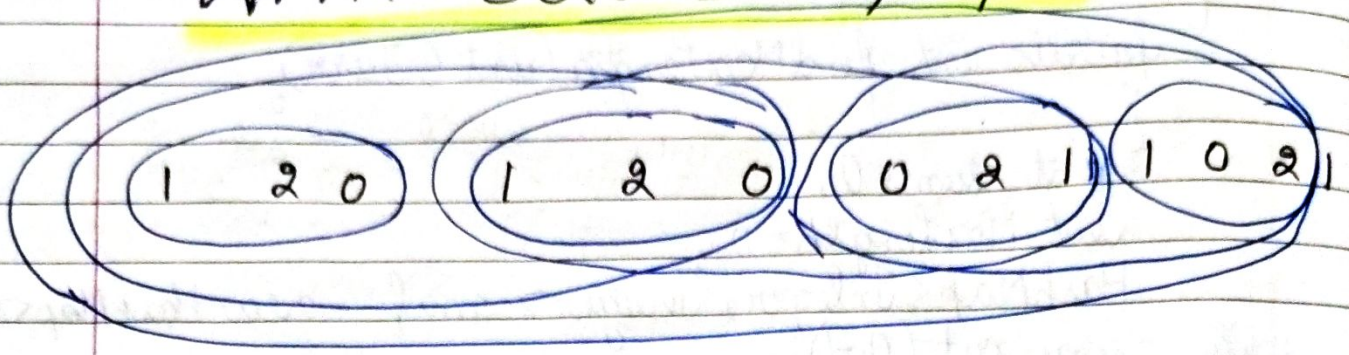
If the length of current subarray is greater than MaxLength then update MaxLength

if the sum occured for the first time, then put it in the Hashmap & its index

return max length

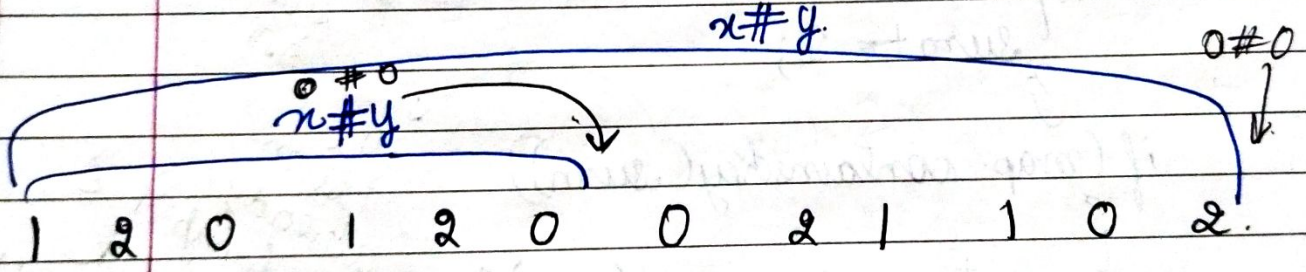
→ lyfg

3. COUNT OF SUBSTRING WITH EQUAL 0, 1, 2.



gap 1,0 # gap 2,1

ये होगी हमारी keys in the Hashmap.
& the no. of occurence of a particular key will be its value.



If we see same key occur again, it means the no. of 0, 1, 2 in between are equal

So we will keep a count of 0, 1, 2 and get the key →

gap 1 0 # gap 2 1

gap in the number of 1 & no. of 0

gap in no. of 1 & no. of 2

	1	2	0	1	2	0	0	2	1	1	0	2	1
count0	0	0	0	1	1	1	2	3	3	3	4	4	4
count1	0	1	1	1	2	2	2	2	2	4	4	4	5
count2	0	0	1	1	1	2	2	2	3	3	3	4	4
	0#0	1#1	1#0	0#0	1#1	1#0	0#0	-1#0	-1#1	0#0	1#-1	0#-1	0#0	1#-1
	1	1	1	2	2	2	3	1	1	4	3	1	5	4

add1 add1 add1 add2 add3 add2 add4 add3

ans = 1 + 1 + 1 + 2 + 3 + 2 + 4 + 3

= 3 + 2 + 3 + 2 + 4 + 3

= 5 + 5 + 7

= 17.

<code>

```
public static long NumberOfSubArrayWithEqualOne
                    (int arr[]).
{
    int count0=0;
    int count1=0;
    int count2=0;
    String key = "0#0";
```

Initially count of 0, 1, 2 is 0

Initially the difference (gap) of 1 & 0 and gap of 2 & 1 is 0.

gap 10 ↑ gap 21 ↑

```
HashMap<String, Integer> map = new HashMap<>();
map.put(key, 1);
```

0#0 key को put किया है HashMap में Because initially में count of 0, 1, 2 are all zero.

```
int ans=0;
```

```
for(int i=0; i<arr.length; i++)
{
    int val = arr[i];
    if(val==0)
    {
        count0++;
    }
    else if(val==1)
    {
        count1++;
    }
    else{
        count2++;
    }
int key = (count1-count0) + (count2-count1);
    if(map.containsKey(key))
    {
        int x = map.get(key);
        ans += x;
        map.put(key, x+1);
    }
    else{ map.put(key, 1); }
}
return ans;
}
```

gap of 1 & 0

gap b/w 2 & 1

Get the value corresponding to the key

Add the earlier value in answer

add the 1 to the already present value

If this key is not already present, then add it with value 1.

return the ans

4. Longest Subarray with equal 012.

	0	1	0	2	0	1	0
Count0	0 1	1	2	2	3	3	4
Count1	0 0	1	1	1	1	2	2
Count2	0 0	0	0	1	1	1	1
	0#0	-1#0	0#-1	-1#-1	-1#0	-2#0	-1#-1	-2#-1
index -1		0	1	2	3	4	5	6

Put the key & index in Hashmap

Subtract 0 from 3 = 3
Length = 3
Max Length = 3

Subtract 2 from 5 = 3
Length = 3
Max Length = 3

हमे Subarray की length return करना है जिसमे count of 0, 1, 2 equal हो और length of this subarray should be maximum of all subarrays with equal 0, 1, 2.

→ **Approach**

1. हम हर point पर 0 का count,
1 का count,
2 का count
calculate करते चलेंगे और ये check करेंगे कि $x_1 - x_0$ और $x_2 - x_1$ की value क्या है.

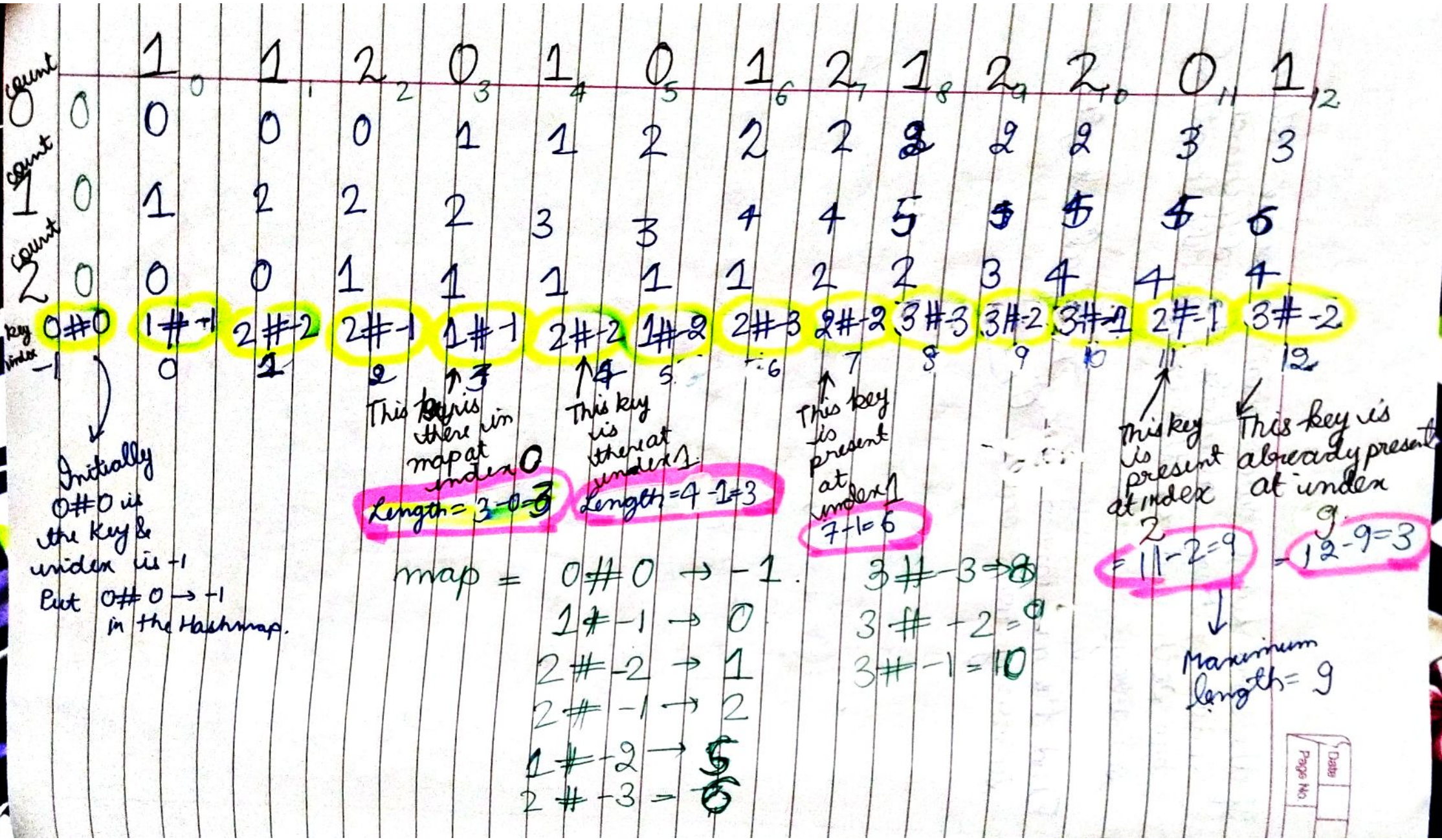
$x_1 - x_0 \# x_2 - x_1$

This Key is a string

→ If the same gap occur again at some other index it means the count of 0, 1, 2 in b/w 2 gaps are equal

2. If the same key, occur again at some other index, then calculate the difference b/w both index, this will be the length of subarray

3. Find the maximum length of all lengths of subarrays with equal 0, 1, 2.


count
1 0 1 1 2 2 0 3 1 4 0 5 1 6 2 7 1 8 2 9 2 10 0 11 1 12
0 0 0 0 1 1 2 2 2 2 2 2 3 3
1 0 1 2 2 2 3 3 4 4 5 5 5 5 6
2 0 0 0 1 1 1 1 1 2 2 3 4 4 4
key index
0#0 1#-1 2#-2 2#-1 1#-1 2#-2 1#-2 2#-3 2#-2 3#-3 3#-2 3#-1 2#-1 3#-2
-1 0 1 2 3 4 5 6 7 8 9 10 11 12
Initially 0#0 is the key & index is -1
Put 0#0 → -1 in the Hashmap.
This key is there in map at index 0
Length = 3-0=3
This key is there at index 1.
Length = 4-1=3
This key is present at index 1
7-1=6
This key is present at index 2
=11-2=9
This key is already present at index 9
-12-9=3
map = 0#0 → -1
1#-1 → 0
2#-2 → 1
2#-1 → 2
1#-2 → 5
2#-3 → 6
3#-3 → 8
3#-2 → 9
3#-1 = 10
Maximum length = 9

```
public static int solution(int arr[])
{
    int count0 = 0;
    int count1 = 0;
    int count2 = 0;

    String key = "0#0";

    HashMap<String, Integer> map = new HashMap<>();

    map.put(key, -1);
    int length = 0;
    int MaxLength = 0;

    for(int i=0; i<arr.length; i++)
    {
        int val = arr[i];
        if(val==0)
        {
            count0++;
        }
        else if(val==1)
        {
            count1++;
        }
        else{
            count2++;
        }
    }

key = (count1 - count0) + "#" + (count2 - count1);
```

Initially count of 0, count 1, count of 2 is 0

This is the key initially

This is the hashmap.

Indicating length and index by -1

Put the key & its index as -1

We will travel on all the values in the array

If the val is 0 count0 ++

If val is 1, increase count1 by 1

If val is 2, increase count2 by 1

The key will be gap of 1 & 0 # gap of 2 & 1

If this key is already present in Hashmap then find the defference between current index & key value.

```
if (map.containsKey(key))
{
    int length = i - x;
    MaxLength = Math.max(length, MaxLength);
}
else
{
    map.put(key, i);
}
}
return MaxLength;
}
```

This will be length of current substring.

The maximum length will be the max of length & maxlength.

If the key is not already present, then put the key with value as current index.

current index.

leetcode
gfg

5. Count of Subarray with Sum equal to K.

Sum = 14.

2 5 1 6 7 3 4 8 2

Sum=0 2, 7, 8, 14, 21, 24, 28, 36, 38

2 → check if 2-14=-12 sum is already present in Hashmap

7 → 7-14=-7 Sum is not present

8 → 8-14=-6 sum is not present

14 → 0 sum found (Add 1 (freq of 0))

21 → 21-14=7 Sum found. Add 1 (freq of 7)

24 → 24-14=10 sum not found

28 → 28-14=14 sum found. Add 1 (freq of 14)

38 → 38-14=24 found. Add 1 (freq of 24)

36 → 36-14=22 Not found

= 1 + 1 + 1 + 1.

= 4.

ईस Sum की frequency को hashmap में store करेंगे

(12-K) → मतलब के sum यहाँ तक है और इसके बाद है 12-6=6=K

1 5 2 -2 3 3

Sum 0, 1, 6, 8, 6, 9, 12

6-6=0 add freq of 0 = 1

6-6=0 add freq of 0 = 1

12-6=6 add freq of 6 = 2

1+1+2 = 4

Check if Sum-K is present in HashMap or not.

If yes, then add its frequency in answer.

<code>.

```
public int subarraySum(int[] nums, int k)
{
    int sum = 0;
    HashMap<Integer, Integer> map = new HashMap<>();
    map.put(0, 1);          // Initially 0 sum has frequency 1
    int ans = 0;

    for(int val : nums)
    {
        sum += val;
        if(map.containsKey(sum-k))
        {
            int freq = map.get(sum-k);
            ans += freq;    // add the freq of (sum-k) in ans
        }

        if(map.containsKey(sum))
        {
            map.put(sum, map.get(sum)+1);
        }
        else
        {
            map.put(sum, 1);
        }
    }
    return ans;
}
```

If the sum-k is present in the HashMap at 3rd index & current index & sum is 5

If this sum is already present, then increase its frequency in the map

else put this sum in the map with its freq = 1

6 Longest Sub Array with Sum K. → gfg

K=15

0	1	2	3	4	5
10	5	2	7	1	9

Sum 0 (10) (15) (17) (24) (25) (34)

Put 0 with 1 index in Hashmap

Check is Sum-K present?

if yes then subtract its index from current index

This will be length of current subarray.

We need to compare all the length & return Maxlength.

15-15=0 ✓

17-15=2 ✗

24-15=9 ✗

25-15=10 ✓

34-15=19 ✗.

subtract 1 from 1-2

4-0=4

If this sum is not present in map Put this sum with its index

map

0	-1.
10	0
~~15~~	~~1~~
17	2.
24	3.
25	4
34	5

length

2-(-1)=2

4-0=4

So maxlength = 4.

Approach ① add (0, -1) in map. (0 → sum, -1 → index)

② इस array के हर index की value को sum variable में add करेंगे.

③ if (sum - k) key is present in HashMap,
then
the length of the corresponding subarray will be currentIndex - (sum - k) index

④ If the current sum is not present in the Hashmap, put it with the current index as its value.

(sum, index)
↳ current index

⑤. Compare length of the subarrays & return the maximum length.

<Code>

```
public static int lenOfLongSubarr
                (int nums[], int k)
{
  int sum=0;
  HashMap<Integer, Integer> map = new HashMap<>();
  map.put(0,-1);

  int length=0;
  int MaxLength=0;

  for(int i=0; i<nums.length; i++)
  {
    int val=nums[i];
    sum+=val;
    if(map.containsKey(sum-k))
    {
      int index = map.get(sum-k);
      length=i-index;
      MaxLength=Math.max(length,MaxLength);
    }
    if(map.get(sum)==null)
    {
      map.put(sum,i);
    }

  }
  return MaxLength;
}
```

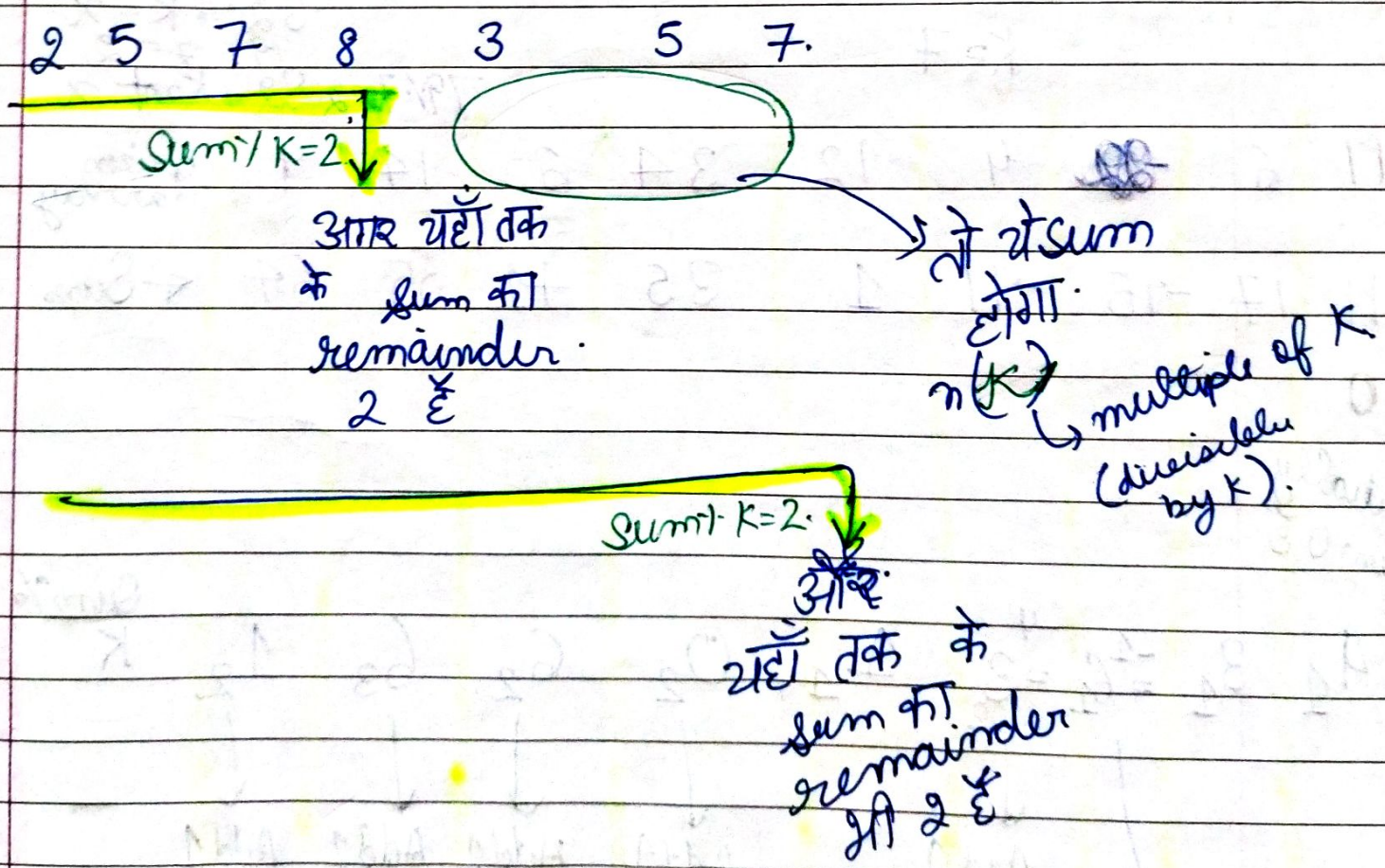
7.

Subarray sum divisible by K.

K = 4.

2 5 7 8 3 5 7

Hashmap में हम store करेंगे sum % K with its frequency
↓
remainder.



k = 7

a b c d e f g h i j

S1 - S1.

S1 % k = S1 % 7 = 2.

S1 = km + x.

S2 - S1 = k(x - m).

This is a subarray divisible by k.

S2

S2 % k = x

79 % 7 = 2

S2 = kn + x

k = 7

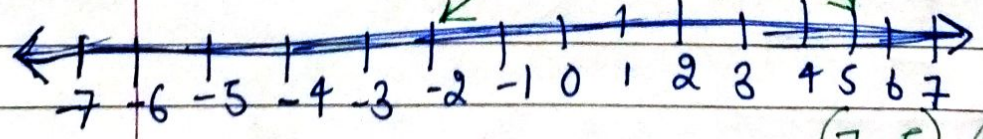
11	6	-32	4	12	34	6	14	9	given array
0 11	17	-15	-11	1	35	41	55	64	← Sum

Initially sum 0 ✓

0_1 4_1 3_1 $=6_1^{-1}$ $=3_2^{-4}$ 1_1 0_2 6_2 6_3 1_2 Sum % K

Add 1 Add 1 Add 1 Add 2 Add 1

= 1 + 1 + 1 + 2 + 1 = 6



Sum = kx - α + k - k

= k(x - 1) + (k - 2)

= k(x - 1) + (k - 2)

= 7 · 5 - 3

= (7 · 5) - (3 + 7) - 7

= 7(5 - 1) + 4

= 7 · 4 + 4.

```java
public static int solution (int[]arr, int K)
{
  HashMap<Long, Long> map = new HashMap<>();
  long sum = 0;
  long ans = 0;

  map.put(sum, 1L);
  for(int i=0; i<arr.length; i++)
  {
    long val = arr[i];
    sum = sum + val;
    long remainder = sum % k;
    if (remainder <0)
    {
      remainder + =k;
    }
    if(map.containsKey(remainder))
    {
      Long freq = map.get(remainder);
      map.put(remainder, freq+1);
      ans + = freq;
    }
    else
    {
      map.put(remainder, 1L);
    }
  }
  return ans;
}
```

} Initially sum & answer is 0

This is how we put 1 (Integer) as Long (यहां L लगाते Integer value के पीछे)

remainder की value निकालो

अगर remainder -ve है तो उसमें K add करतो ताकि remainder +ve हो जाए

अगर remainder map में already है

तो उसकी frequency ans में add करदो

frequency को increase करदो

अगर remainder already present नहीं है तो उसे add करदो with frequency 1

ans की value return करदो !.