992. Subarrays with K Different Integers

eg: nums = [1,2,1,2,3] K=2

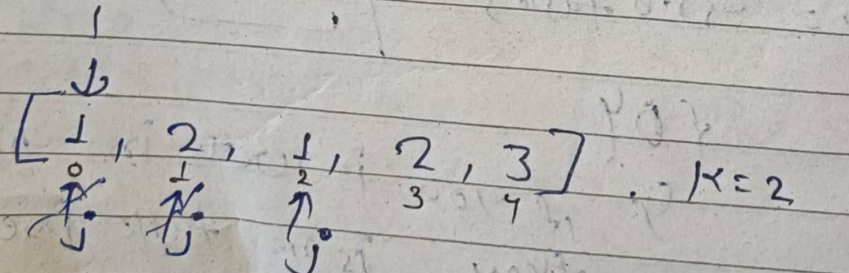
res = 7

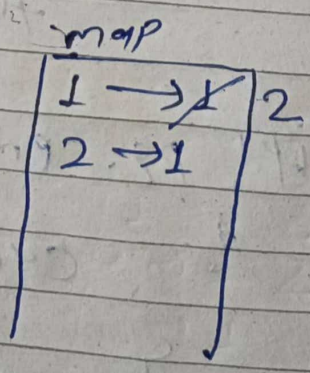
good subarrays = [1,2], [1,2,1], [1,2,1,2], [2,1], [2,1,2], [1,2], [2,3]

Note:- It seems like you can apply Sliding window approch but can it work with Khandani Slidinding window approch-

let's see.

Applying Standard template of Sliding window

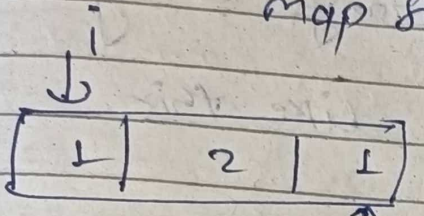




result :
at idx j=1
(1,2)

Teacher's Signature

at idx = 2 ⟶ i.e. j = idx - 2 &
map size still = 2



⟶ (1, 2, 1)

⟶ क्या इस valid window में only (1, 2, 1) good subarray है या or ~~कुछ~~ और भी subarray exist करता है.

yes ⟶ (2, 1] also exist in this valid window.

How can you find that (2, 1) is also a good subarray in the given window.

⟹ (j - i + 1) ⟶ इस से हम यह पता लगाते हैं कि window i to j में j पर end होने वाले total subarray कितने हैं।

If we apply this then we get

[1, 2, 1)

(j - i + 1) = 2 - 0 + 1 = 3

(1, 2, 1) (1, 2) (1)

this subarray is not valid.

Teacher's Signature

So By the formula $(j-i+1)$ we can compute the total subarrays ending at j.

So if your window is like this

$[1, 2, 1]$

then $(j-i+1) = 3$

तो इसका मतलब है कि हमने तो सारी subarray निकाल लिये जिसमें unique elements या तो K के equal हो या less than K हो।

Note :- Here we can do like this first we can count all the subarrays in which different integers at most K
i.e. $<= K$
then we cont for $<= (k-1)$ &
if we subtract this then we have only subarray in which cnt is equal to K.

e.g. $[1, 2, 1, 2, 3]$, $K = 2$

cnt += $(j-i+1)$

map

1 →	2
2 →	2
3 →	1

Teacher's Signature .....

res =

j=0 ⟶ {1}

j=1 ⟶ {1,2}, {2}

j=2 ⟶ {1,2,1}, {2,1}, {1}

j=3 ⟶ {1,2,1,2}, {2,1,2}, {1,2}, {2}

j=4 ⟶ {2,3}, {3}

~~total Cnt~~ = 12

Now cnt ~~for~~ <= (k-1) : k=1

total cnt = 5

12 - 5 = 7

Approch -2

हमारी कि दिमाग में थी कि

k=2

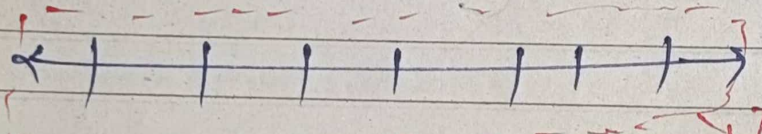
[1, 2, 1, 1, 2] (i and j at index 0)

j++ → [1, 2, 1, 1, 2] (i at 0, j at 1) → valid subarray with k different integers.

j++ → [1, 2, 1, 1, 2] (i at 0, j at 2) → This is also a valid subarray but you have (1,2,1) & (2,1) is also valid.

but you don't have any formula to finding that (2,1) is also valid but if you apply $(j-i+1)$ you get all subarrays ending at j.

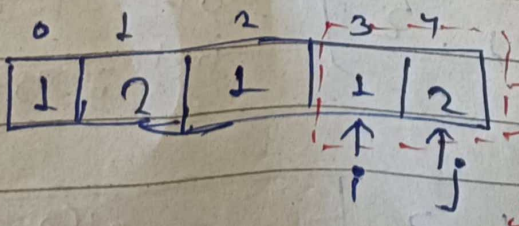
Then our problem $(f(i+1, j))$ is like that.



This a valid subarray.

and suppose in this there are many subarray which also valid so we can't figure out those directly so we you this approach.

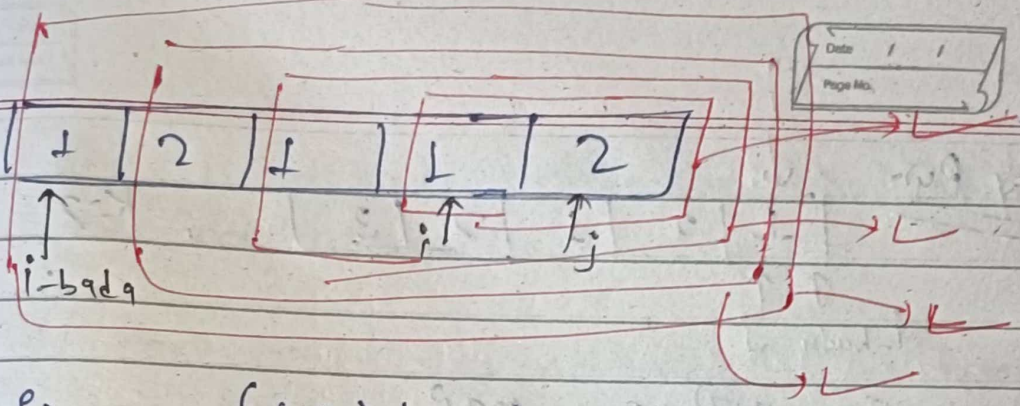
New approach Intution



This is your smallest valid subarray ending at j

Now what other subarrays are also possible like.

Teacher's Signature



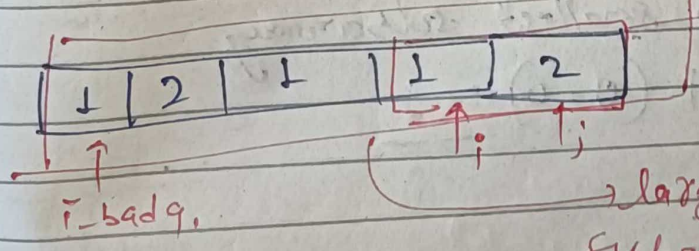
i.p. (1,2) ↳
(1,1,2) ↳
(2,1,1 2) ↳
(1,2,1,1,2) ↳

Or you can say that I am shifting i to left till that I can get the largest valid subarray.

res. += 1 + (i - i_badq)

for smallest valid subarray

for remaining valid subarray.



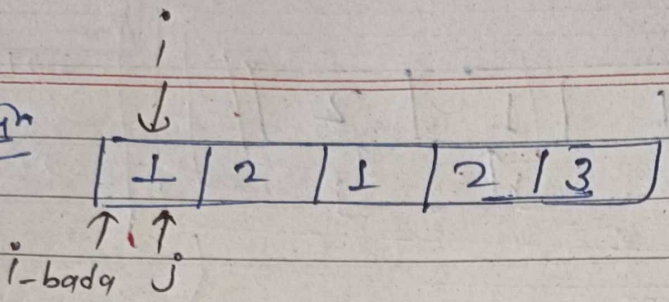
→ smallest valid subarray ending at j

→ largest valid subarray ending at j

So from (i - i_badq) you can find the no. of valid subarrays

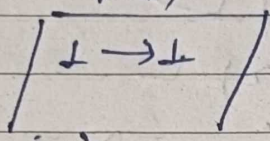
Teacher's Signature

dry Run



, K=2

mp[nums[j]]++

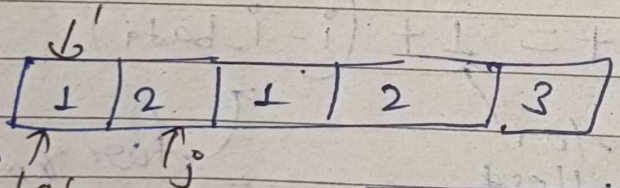


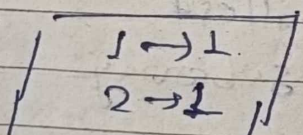

```
while(mp.size() > k)
{
    shrink the window
    i.e. i++. & also i_badq = i;
}
```

// find the smallest valid subarray

```
res += 1 + (j - i_badq)
j++;
```

j = 1



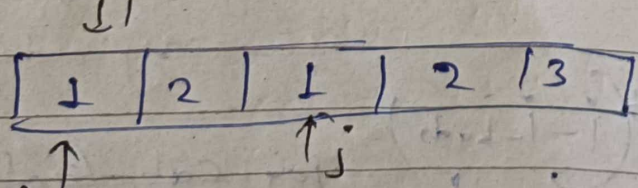


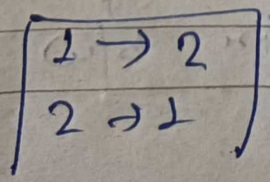
valid & also smallest subarray

res += 1 + (0 - 0) = 1

j++

j = 2





j = 3

j = 4

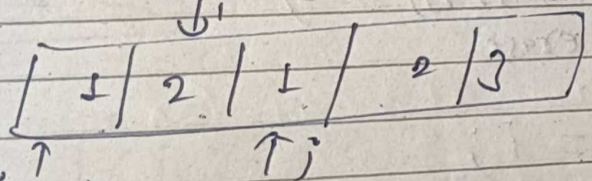
Teacher's Signature

This is a valid subarray but not the smallest valid subarray.

// how could you make the subarray the smallest valid subarray

```
while (mp[nums[i]] > 1)
{
    mp[nums[i]]--;
    i++;
}
```

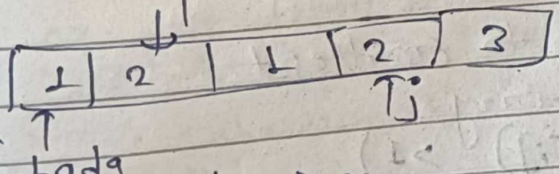
so mp[nums[0]]--; i++;



res += 1 + (j - 0) = 2 + 1 = 3

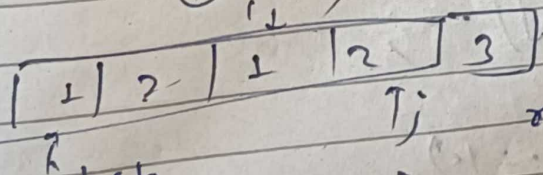
j++

j=3



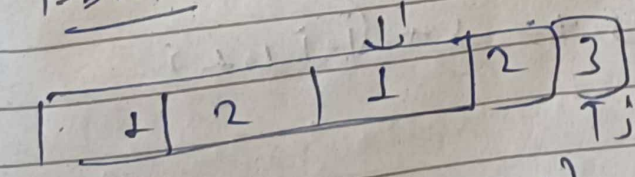
1 → 1
2 → 2

i++;



res += 1 + (j - i-bada)

j=4



1 → 1
2 → 2
3 → 1

while (mp.size() > k)

{

// Psude code

```
while (j<n)
{
    mp[nums[j]]++;

// step-1: if mp.size() > k
           shrink window

while(mp.size()>k)
  {     mp[nums[i]]--;
        if(mp[nums[i]]==0)
              erase
        i++;
        i_bada = i;
  }

// make the window to smallest
   valid sub array
while (mp[nums[i]] >1)
  {  mp[nums[i]]--;
       i++;
  }
  if(mp.size()==k)
       res += i-i_bada+1;

j++;
}
}
```