Hashing - 2

count no. of subarrays with sum = 0

pf[] = . . . 2 . . . . 2 . . . . 2 . . . .

$\frac{x(x-1)}{2}$

0 . . . 1 . . . 1 . 0 . . . . .

1 → 2 $\Rightarrow \frac{2(2-1)}{2} = 1$

for 0 $\Rightarrow \frac{x(x+1)}{2}$

5 → 3 $\Rightarrow \frac{3(3-1)}{2} = 3$

10 → 4 $\Rightarrow \frac{4(4-1)}{2} = 6$

Doubt

0 → 1 $\Rightarrow \cancel{\frac{1(1-1)}{2}} = 0$

[0, 0, 0, 0] = 10

$\frac{4 \times 5}{2} = 10$

[-1, 2, -1]

pf = [-1, 1, 0]

0 → 1 $\Rightarrow \frac{1(1+1)}{2} = 1$

Content

→ Pair sum = K

→ Distinct elements in every window of len=K

Question 1

Given N array elements, cheek if there exists a pair (i,j) s.t. $a[i] + a[j] = \boxed{K}$ & $i != j$

K is given

→ return true/false

eg

	0	1	2	3	4	5	6	7	8	9
a[] =	8	9	1	-2	4	5	11	-6	7	5

$K=11 \Rightarrow a[4] + a[8] = 4 + 7 = 11$ True

$K=6 \Rightarrow a[2] + a[5] = 1 + 5 = 6$

$a[0] + a[3] = 8 - 2 = 6$ True

$K=22 \Rightarrow \underbrace{a[6] + a[6]}_{i==j \times (i!=j)} = 11 + 11 = 22$ False

Idea1 : Check all pairs sum = K

```
for (i=0; i<n; ++i ) {
        x = a[i] , x+y=K , y = K-x = K-a[i]
    for (j=i+1; j<n; ++j) {
        if (a[i]+a[j] == K)  => if (a[j]==y)
            return true
    }
}
return false
```

TC : O(N^2)
SC : O(1)

Idea2 : Sorting + 2 pointer

[Ignore if don't understand]

TC : O(NlogN)   SC : O(N)

Idea3 : Use Hashset

	0	1	2	3	4	5	6	7	8	9
a[] =	8	9	1	-2	4	5	11	-6	7	5

K = 11

hs = { 8, 9, 1, -2, 4, 5, 11, -6, 7 }

x	y = K-x	check if y is present in hashset or not?
8	3	NO
9	2	NO

1	10	NO
-2	13	NO
4	7	YES {return true}

Say $K = 5$

x	y	present or not?
8	-3	NO
9	-4	NO
1	4	YES {return true}

$K = -4$

x	y	present or not?
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	YES {return true}

Basically, if $x = y$ then occurance of x should be more than 1.

NOTE: Freq. of elements is important to know.

Idea4 : Use Hashmap

	0	1	2	3	4	5	6	7	8	9
a[] =	8	9	1	-2	4	5	11	-6	7	5

hm = { <8,1> , <9,1> , <7,1> , <-2,1> , <4,1> ,
<5,2> , <11,1> , <-6,1> }

K=10

x	y	present or not ?
8	2	NO
-2	12	NO
4	6	NO
5	5	if (x==y && freq(x)>1) { return true }

Code

```
bool pairSum( a[] , K ) {
    Hashmap<int, int> hm
    insert  a[] → hm       // TODO    TC : O(N), O(N)
for (i=0; i<n; ++i) {
    x=a[i] , y= k-a[i]
    if ( hm.search (y) == true) {
        if (x != y)
            return true

        if (hm[y] > 1)
            return true
    }
}
return false
}
```

TC: O(N)

SC: O(N)

Idea 5: Use Hashset again

at i^{th} index, hs will only contain elements from [0, i-1] index.

	0	1	2	3	4	5	6	7	8	9
a[] =	8	9	1	-2	4	5	11	-6	7	5

K=22

x	y	hs	present or not ?
8	14	{}	NO

9	13	{8}	NO
5	17	{8,9}	NO
-2	24	{8,9,5}	NO
11	11	{8,9,5,-2}	NO

→ now code is working for x=y & freq(x)=1

K=10

x	y	hs	present or not?
8	2	{ }	NO
9	1	{8}	NO
5	5	{8,9}	NO
-2	12	{8,9,5}	NO
11	-1	{8,9,5,-2}	NO
5	5	{8,9,5,-2,11}	YES {return true}

Code

```
bool pairSum (a[], K) {
    Hashset<int> hs
    for (i=0; i<n; ++i) {
        x = a[i] , y = K - a[i]
        if ( hs.search(y) == true )
            return true
        hs.insert(x)
    }
    return false
}
```

TC : O(N)

SC : O(N)

Question 2

Given N elements, calculate no. of distinct elements in every subarray of size K.

eg a[10] =

	0	1	2	3	4	5	6	7	8	9
a[10] =	2	4	3	8	3	9	4	9	4	10

K=4

Subarrays	distinct elements
[0,3]	4
[1,4]	3
[2,5]	3
[3,6]	4
[4,7]	3
[5,8]	2
[6,9]	3 ⇒ Ans

Idea1 : for every subarray of len = k, insert into hashset & find size.

Code

```
for (i=0; i< n-k+1; ++i ){  → (n-k+1) iterations
    Hashset<int> hs
    for ( j=i; j<i+k; ++j ){  →  k iterations
        hs.insert(a[j])
    }
    print( hs.size)
}
```

TC: $O(K(n-K+1)) \Rightarrow O(N^2)$ SC: $O(K)$

1. $K = n \rightarrow$ TC: $O(n \times (\cancel{n}-\cancel{n}+1)) = O(N)$

2. $K = 1 \Rightarrow$ TC: $O(1 \times (n-1+1)) = O(N)$

3. $K = n/2 \Rightarrow$ TC: $O(n/_2 \times (n - n/_2 + 1)) = O\left(\frac{n^2}{4}\right) = O(N^2)$

Idea2 : Sliding window using Hashset

$$a[10] = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 4 & 3 & 8 & 3 & 9 & 4 & 9 & 4 & 10 \end{array}$$

		Hashset	hs.size
[0,3]	→	{2,4,3,8}	4
[1,4]	remove a[0] insert a[4]	{4,3,8}	3
[2,5]	remove a[1] insert a[5]	{3,8,9}	3
[3,6]	remove a[2] insert a[6]	{8,9,4}	3 ✗

In hashset we don't know the frequency of inserted element. If we insert 2 times & remove 1 time, there should be 1 element in the hashset but we can't do it.

Idea 3 : Sliding window using Hashmap

a[10] =

0	1	2	3	4	5	6	7	8	9
2	4	3	8	3	9	4	9	4	10

		Hashmap	Size
[0,3]	$\longrightarrow$	{<2,1>, <4,1>, <3,1> <8,1>}	4
[1,4]	$\xrightarrow[\text{insert a[4]}]{\text{remove a[0]}}$	{<4,1>, <3,2>, <8,1>}	3
[2,5]	$\xrightarrow[\text{insert a[5]}]{\text{remove a[1]}}$	{<3,2> <8,1>, <9,1>}	3
[3,6]	$\xrightarrow[\text{insert a[6]}]{\text{remove a[2]}}$	{<3,1>, <8,1>, <9,1>, <4,1>}	4

Code

```
def  distinctCount(a[], K) {
    n = a.length
    Hashmap<int, int> hm

    for (i=0; i<K; ++i) {
        if (hm.search(a[i]) == true)
            hm[a[i]]++
        else
            hm.insert({a[i], 1})
    }
    print(hm.size)          → for first subarray [0, k-1]

    s=1, e=K

    while (e<n) {
        // subarray :  [s, e]     → remove a[s-1] , add a[e]
        hm[a[s-1]]--

        if (hm[a[s-1]] == 0)      → remove logic
            hm.delete(a[s-1])
```

```
if (hm.search(a[e]) == true)
    hm[a[e]]++
else
    hm.insert({a[e], 1})
```

→ insert

```
print(hs.size)

s++, e++

  }
}
```

TC : O(N)

SC : O(K)

Doubt

count no. of subarrays with sum = 0

a[] = 1 1 -1 2 2 -3 0 -2 1

pf[] = [1 2 1 3 5 2 2 0 1]

freq =>

1 → 3

2 → 3

3 → 1

0 → 1

5 → 1

no. of subarrays ⇒ $\frac{x(x-1)}{2}$ for freq = x of all unique elent

elements	freq = x	$x(x-1)/2$
1	3	$3(3-1)/2 = 3$
2	3	$3(3-1)/2 = 3$
3	1	$1(1-1)/2 = 0$
5	1	0
0	1	$1(1-1)\cancel{/2} = 0$ $1(1+1)/2 = 1$

for 0 formula is $\frac{x(x+1)}{2}$

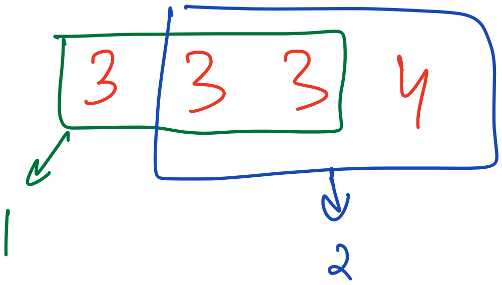
ans = 3 + 3 + 0 + 0 + 1 = 7

a = $\boxed{\boxed{1 \quad -1} \; \boxed{-2 \quad 2}}$

pf[] = 1 0 -2 0

	freq	
1	1	⇒ $1(1-1)/2 = 0$
-2	1	$1(1-1)/2 = 0$
0	2	$2(2+1)/2 = 3$

eg



K = 3

hs = {3}   size = 1

hs = {4}   Size ~~= 1~~