Q1) Given N array elements, Check if there exists a pair (i,j) such that $ar[i]+ar[j] = K$ && i!=j

index values

$a+b=K$

Eg:

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

$K=11$ $i=4$ $j=8$

$K=6$ $i=2$ $j=5$

$K=22$ $i=6$ $j=6$ ✗

Brute force: Check each pair

TC: $O(n^2)$ SC: $O(1)$

$a+b=K \implies \boxed{b = K-a}$

$K=11$ $a=5$ $b=6$

a	b (k-a)
8	3
9	2
1	10
-2	13
4	7

$K=11$

found it!!!

Hashset

Allows me to check if b exists or not. in $O(1)$

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

K=22

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

a	b(k-a)
8	14
9	13
1	21
-2	24
4	18
5	17
11	11

K = 22

Yes

Learning: We need to maintain freq.

Freq Hashmap

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

K=22

HM:

8 :	1
9 :	1
1 :	1
-2 :	1
4 :	1
5 :	2
11 :	1
-6 :	1
7	1

a	b
8	14
9	13
1	21
-2	24
4	18
5	17
11	11
-6	28
7	15
5	17

freq (11) ≥ 2

return false

Pseudo Code

1) Create the frequency hm.

```
for(i=0; i<n; i++) {
    a = ar[i]        b = k-a
    if( a == b ) {
        if( hm.get(a) >= 2 )
            return true.
    }
    else {
        if( hm.containsKey(b))
            return true.
    }
}

return false
```

TC: O(n)   SC: O(n)

Q Count no of pairs a[i]+ a[j] = K   i!=j

Eg   2 5 2 5 8 5 2 8   K=10

b = 8 5 8 5 2 5 8 2   HM   2: ~~1~~ ~~2~~ 3

+1 +2 +2 +1 +3   5: ~~1~~ ~~2~~ 3

8: ~~1~~ 2

```
hashmap<int,int> hm
ans = 0
for( i=0 ; i<n ; i++) {
    b = K-a
    if (hm. contains(b))
        ans + = hm[b]
    hm[arr[i]] ++
}
return ans
```

TC, SC } O(n)

Q Calc the number of distinct elements in each subarray of size K.

Eg: ar[10] =

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

K=4

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

Idea: Calc using hashset for every window

TC: $O(n^2)$

Idea: Optimize using hashmap

0:3 <2,1> <4,1> <3,1> <8,1>

1:4 remove a_0
add a_4

add ⇒ freq++

sub ⇒ freq-- but if freq=0 remove

$\langle 1,4 \rangle$ $\langle 4,1 \rangle$ $\langle 3,2 \rangle$ $\langle 8,1 \rangle$

ans = 3

$\langle 2,5 \rangle$ $\langle 9,1 \rangle$ $\langle 3,2 \rangle$ $\langle 8,1 \rangle$

ans = 3

{done}

0	$k-1$
1	k

Code

```
hashmap <int, int> hm
for (i=0 ; i<k ; i++) {
    hm[a[i]] ++
}
print (hm.size())
s=1          e=k
while (e<n) {
    hm[ar[s-1]] --
    if ( hm[ar(s-1]] ==0)
        hm.remove ( ar[s-1] )

    hm[ar(e]] ++
    print (hm.size)
    s++          e++
}
```

TC O(n)
SC O(k)

Q Check if subarray with sum = K exist

Eg - 2 3 9 -4 1 5 6 2 5   K = 11

K = 10

Idea: Similar to prev class where sum = 0

$$\text{Sum}[s:e] \Rightarrow \begin{cases} pf[e] - pf[s-1] & s \neq 0 \\ pf[e] & s = 0 \end{cases}$$

Create pf array.

	0	1	2	3	4	5	6	7	8
arr	2	3	9	-4	1	5	6	2	5
pf	2	5	14	10	11	16	22	24	29

Now, $pf[e] - pf[s-1] = K$

$$\underbrace{pf[s-1]}_{\text{target}} = pf[e] - K$$

1) Create pf sum

2) hashset <int> hs

```
for ( i=0 ;i<n ;i++ ){
        target = pf[i] -k
        if( hs. contains ( target ))
                return true
      else {
            hs. add (pf[i])
      }
}
 if ( hs. contains(K))
        return true
else
      return false
```

TC $\Big\}$ O(n)
SC

```
sum += a(i)
if(( sum==0) || set.contains(sum))
        ret 1
set.add sum
```

1 3 -1 -2 4

sum 1 4 3 1 5