Agenda!

1. First Non Repeating Element
2. Pair Sum
3. Count of pair Sum
4. check if there exists a subarray with Sum = 0
5. check if there exisys a subarray with Sum = K.

1. First Non-Repeating Element O(N) O(N)

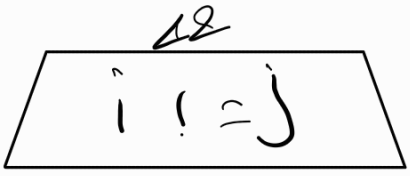
1. Cout the element frequencies in freq map
2. iterate over A again and check the element freqency if it is == 1

2.

Given int[] A and int k

check if there exist a pair (i, j)

such that $\boxed{arr[i] + arr[j] = k}$



Brute force : $O(N^2)$

1. lets create all possible pairs,
   using nested loop one for i one for
   j from i+1 to N. $O(N^2)$
2. and check if element exist or not

optimized :

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

N = 7

k = 6 ✓
k = 22 ×
k = 8 ✓

6 - 8 = -2

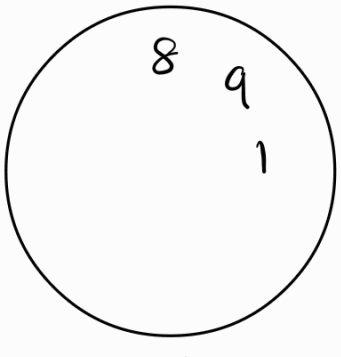
6 - 9 = -3

6 - 1 = 5

6 + 2 = 8 ↓

Return True

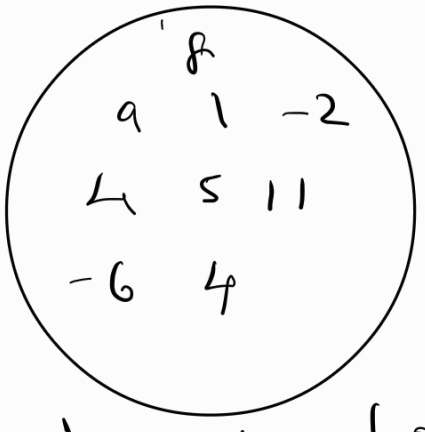
b = k - a



set()

& $b = k - a \Rightarrow -2$

$\Rightarrow$ do we have it in set?



Create set out of A

1. # create st from A
2. # iterate over A and check if b is found in set.

TC: O(N)

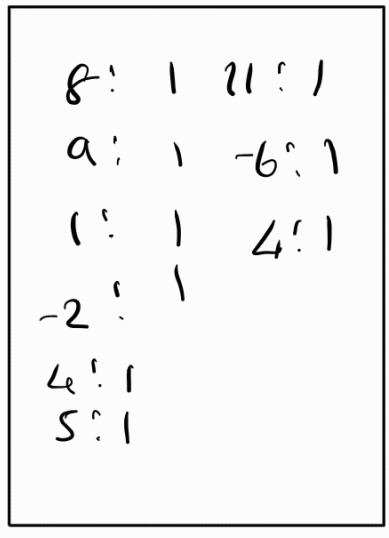
SC: O(N)

issue : above approach will not work $\boxed{i != j}$ condition

1. use dictonary to store element and frequency. and keep i != j

k = 18

18 - 9 = 9



we sould keep check

if freMap[9] == A[i]

if value == 1

↓ Pass

else:

↓ Consider

TC: O(N)

SC: O(N)

Count the number of pairs with sum = K

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

$K = 10$

8

2	5	2	5	8	5	2	8
8	5	8	5	2	5	8	2

$10 - 2 = 8$ ✓ 5

$a + b = k$

$b = k - b$

ans = 2 + 3 + 2 + 3 + 3 + 3 + 2 + 3

5 5 6 5

10 11

21

2	3
5	4
8	1

all the possible pairs = First create dictionary iterat A and all all frequency.

---

from left to right = iterate A and create dictionary. on the fly and add if it finds.

```
freqMap = { }, ans = 0, k = 6
for i in range(N)
    b = (k - A[i])
    value = freqMap.get(b, 0)
    ans += value

if A[i] in freqMap.
↓   ans += value
    increase freqency also.
else:
    freqMap[A[i]] = 1
```

Subarray with Sum Zero

1. Given int [] A
2. Find if subarray with sum zero. return True

Approach 1 : BF $O(N^3)$

1. going through all the subarray and $O(N^2)$
2. check if it's sum = 0 $O(N)$

$A = [2\ \ 2\ \ 1\ \ -3\ \ 4\ \ 3\ \ 1\ \ -2\ \ -3\ \ 2]$ $N = 11$

(indices: 0 1 2 3 4 5 6 7 8 9)

2
2

$\frac{N(N+1)}{2} = 66$

2 2
2 2 1
2 2 1 -3
:

2 1 -3

```
N = len(A)
for i in range(N)
    for j in range(i, N)
        temp = 0
        for k in range(i, j+1)
            temp += A[k]
        if temp == 0:
            return True
return False.
```

approch 2 Prefix Sum $O(N^2)$, $O(N)$ space

1. creating Prfix Sum
2. for every Sub array get sum from prefix Sum

PrefixSum = [ ], prefixsum.append(A[0])
for i in range(1, N): prefixSum.apend (prefixSum[i-1] + A[i])

```
for i in range(N)
    for j in range( i, N)
        temp = 0
        if i == 0: temp = prefixSum[j]
        else:
            temp = prefixsum[ j] - prfixSum[i-1]
        if temp == 0: return True
return False
```

approch 3: carry forword $O(N^2)$

```
for i in range(N)
    temp = 0
    for j in range( i, N)
        temp += A[j]
        if temp == 0
            return True
```

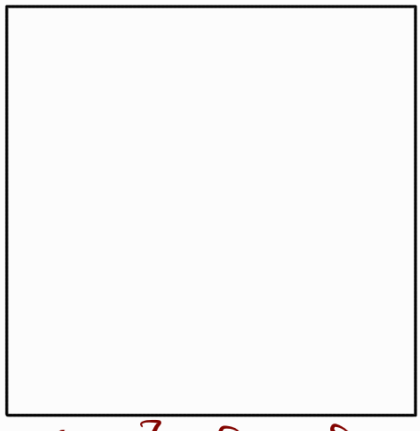
approch 4:

Sum(i, j) = 0

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

i

j



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

i j

= Prefix[j] - Prefix[i-1]

= 6 - 2

=> 4

idea:

sum(i,j) = 0

Prefix [j] - Prfix[i-1] = 0

Prfix[j] = Prfix[i-1]

So, if Prfix[j] and Prfix[i-1] are same. they will mak "Zero"

get frequency of all element in prefixsum

2 : 2 = True

4 : 1

5 : 2

6 : 1

9 : 1

10 : 1

8 : 1

7 : 1

A = [-2 -1 3 5]

PF = [-2 -3 0 5]

```
if value = 0 :
        return True
```

-2 : 1

-3 : 1

0 : 1

5 : 1

Subarray with sum → k

⇒ Given int [] A, check if there is a subarray with sum `k'

$1 \le N \le 10^8$

arr = [2 , 3, 9 , -4 , 1, 5, 6 , 2, 5]

k = 11

k = 10

Sum (i, j) = k

PrfixSum[j] − prfixsum[i-1] = k

PF = [2 5 14 10 11 16 22 24 29]

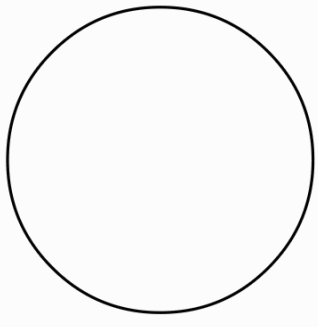
arr = [5 10 20 100 105]

PF = [5 15 35 135 240]

k = 10

PF [j] - PF[i-1] = k

15 - 5 = 10



Approch 1 : BF $O(N^3)$

1. going through all subarray and check if
2. sum is == k.

Approch 2 :

$\Rightarrow a - b = k$

add b both sides

$\Rightarrow a - b + b = k + b$

simplify it.

$\Rightarrow a = k + b$

subtract k from both sides

$\Rightarrow a - k = k + b - k$

Simplify

$a - k = b$

$b = a - k$

$Sum(i, j) = k$

$PrfixSum[j] - prfixsum[i-1] = k$

(a) - (b) = k

as per derived equation:

$Prefix[i-1] = Prefix[j] - k$

$b = a - k$

$Prefix[i-1] = Prefix[j] - k$

Dry run!

```
                                  j
arr = [ 2   3   9   -4   1   5   6   2   5 ]
```

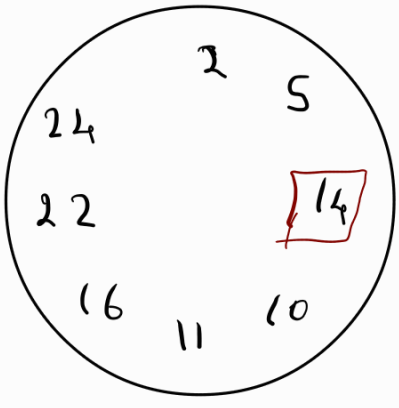
Subarray = 15

k = 15

```
PF = [ 2   5   14   10   11   16   22   24   29 ]
       ↓   ↓    ↓    ↓    ↓    ↓    ↓    ↓    ↓
      -13 -10   -1   -5   -4    1    7    9   14
```

29 - 15 = 14

2 - 15 = -13




```
arr = { 2   3   9   -4   1 }

PF  = [ 2   5   14   10   1 ]        k = 10
        8   5    4    0   1
```

if PF[j] == k

return True

one way

2nd way

add 0 beforehand

Code :

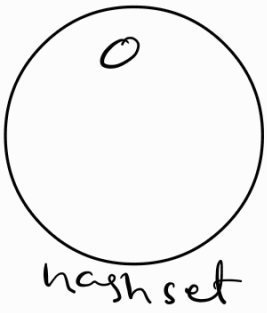
```
# creatR prefixsum O(N)
Pf = [ ]
Pf.append( A[0])
for i in range( 1, N)
    ↓ Pf.append( Pf[i-1] + A[i])
# iterate through the Pf array and O(N)
find if sum = 0 exists
hashset = 0,
for j in rang(N)
    # b = a - K formula
    b = Pf[j] - K
    if b in hashset :
        ↓ return True
    else:
        ↓ hashset.add( PF[j])
```

TC : O(N)

SC : O(N)

without using PF array

```
sum = 0
hashset = {}
for j in range(N)
    sum += A[j]
    b = sum - K
    if b in hashset:
        return True
    else:
        hashset.add(sum)
return False
```



TC: O(N)

SC: O(N)