Problem with given diff.

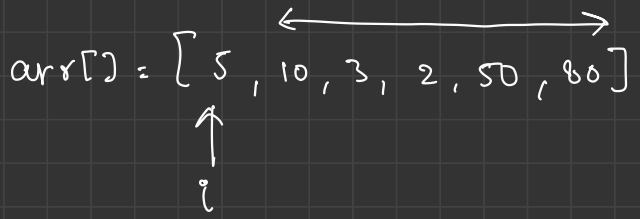
arr[] = [5, 10, 3, 2, 50, 80] B = 78

pair (10, 3)

↳ 10 − 3 = 7

↳ 3 − 10 = −7

Brute force



TC : O(N²)

SC : O(1)

```
for (int i = 0 → n)
{
    for (int j = i+1 → n)
    {
        if (arr[i] - arr[j] == B ||
            arr[j] - arr[i] == B)
            return 1;
    }
}
```

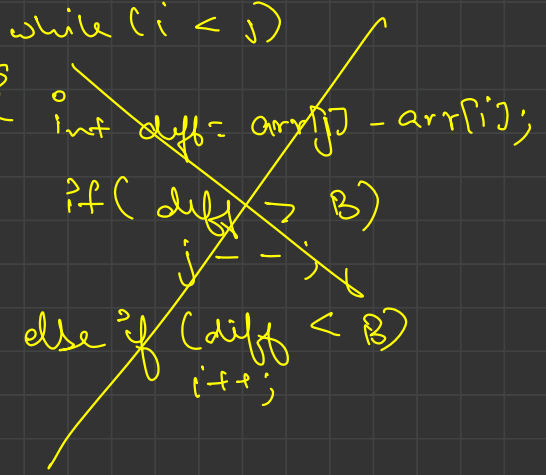
arr[] = [5, 10, 3, 2, 50, 80] B = 45

① Sort the array. → TC: O(N log N)

[2, 3, 5, 10, 50, 80]

i j

NOT WORKING



arr[] = [5, 10, 3, 2, 50, 80]

B = 78

80 - 78

$x - y = B$

$x = 80, y = 2$

(80, 2)

let $x = 5$

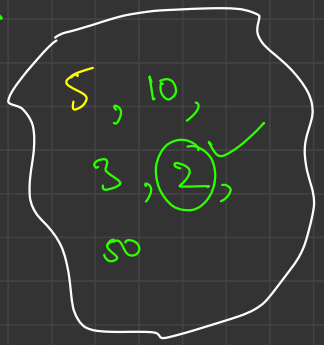
$5 - y = B$

$y = 5 - B$ — (1)

let $y = 5$

$x - 5 = B$

$x = B + 5$ — (2)



set

x = 50, y = -28

y = 50, x = 128

x = 2, y = -76

y = 2, x = 80

x = 3, y = -75

y = 3, x = 81

x = 5

y = 5 - 78 = -73 X

y = 5, x = 78 + 5 = 83

x = 10, y = -68

y = 10, x = 88

Array Pair divisible by K

arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} K = 5

n = 10 (5)

(9,1) , (2,8) , (3,7) , (4,6) , (5,10)

4 1 2 3 3 2 4 1 0 0

if rem1 = 1

rem2 = 5 - 1 = 4

rem1 = 2

rem2 = 5 - 2 = 3

arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} k = 5

1 2 3 4 0 1 2 3 4 0

rem	freq
1	2
2	2
3	2
4	2
0	2

freq-Map

O(N)

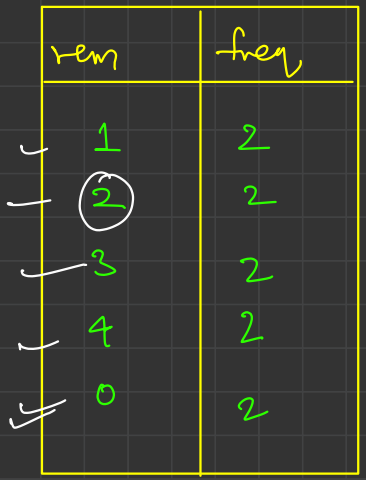
```java
public boolean arrayPairs(int[] arr, int k) {
    // write code here
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < arr.length; i++) {
        int rem = arr[i] % k;
        map.put(rem, map.getOrDefault(rem, defaultValue: 0) + 1);
    }
}
```

1 +1

0 - (k-1)

x % k

0, 1, 2, ..., (k-1)



O(N)

```
for (int rem : map.keySet()) {
    int compRem = k - rem;

    if (map.containsKey(compRem) == false) {
        // compRem is not present
        return false;
    } else if (map.get(compRem) != map.get(rem)) {
        // compRem freq != rem freq
        return false;
    }
}

return true;
```

5 - 0 = 5

rem = 1 , compRem = 4

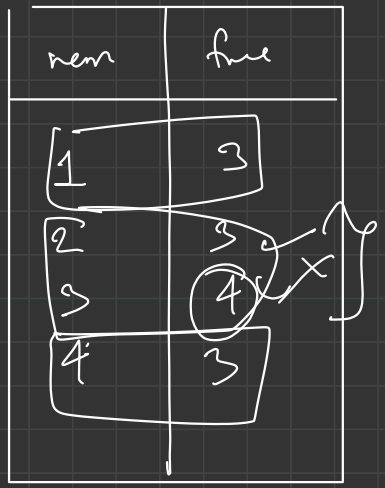
rem = 2 , compRem = 3

rem = 3 , = 2

TC: O(N)

SC: O(K)

K=5



rem = 1 comp = 4

rem = 2 comp = 3

Largest Subarray with zero Sum

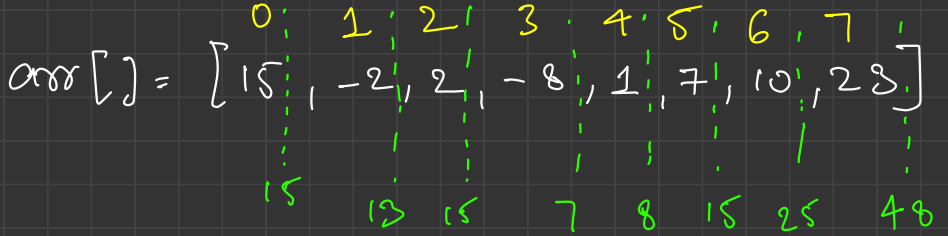
arr[] = [15, -2, 2, -8, 1, 7, 10, 23]

Bruteforce

compute all subarray sum, and store Maxlen.

TC : $O(N^2)$

SC : $O(1)$



15 → 0

13 → 1

7 → 3

8 → 4

len of subarray with sum zero = 2 − 0 = 2

Sum occured again

Sum encountered first time

5 − 0 = 5

arr = [1, -1, 2, -2]

0 1 2 3

sum = ~~1~~ ~~0~~ ~~2~~ 0

1 - (-1) = 2

3 - (-1) = 4

sum	idx
0	-1
1	0
2	2

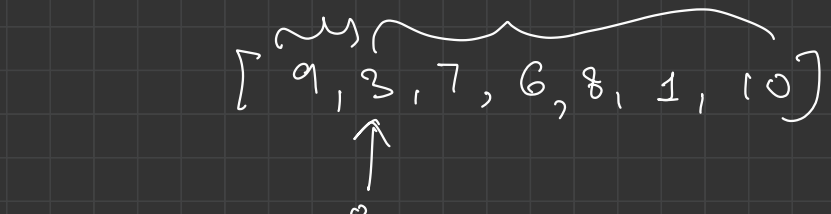
```java
// TC: O(N), SC: O(N)
public int maxLen(int arr[]) {
    // Write your code here
    HashMap<Integer, Integer> map = new HashMap<>();

    int sum = 0;
    int mlen = 0;
    map.put(key: 0, -1);

    for (int i = 0; i < arr.length; i++) {
        sum += arr[i];

        if (map.contai
            int len = i - map.get(sum);
            mlen = Math.max(mlen, len);
        } else {
            map.put(sum, i);
        }
    }

    return mlen;
}
```

int i = Solution.maxLen(int[])

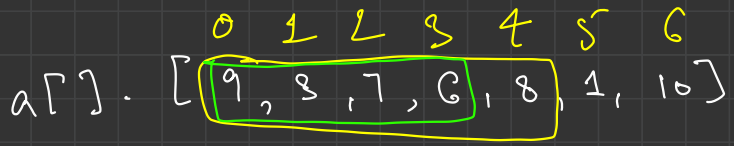
Equilibrium Index

a[] . [9, 3, 7, 6, 8, 1, 10]

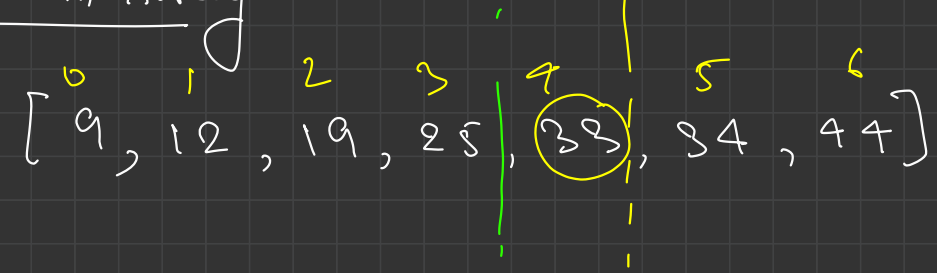
Bruteforce Approch



TC: O(N²)



Prefix Sum Array

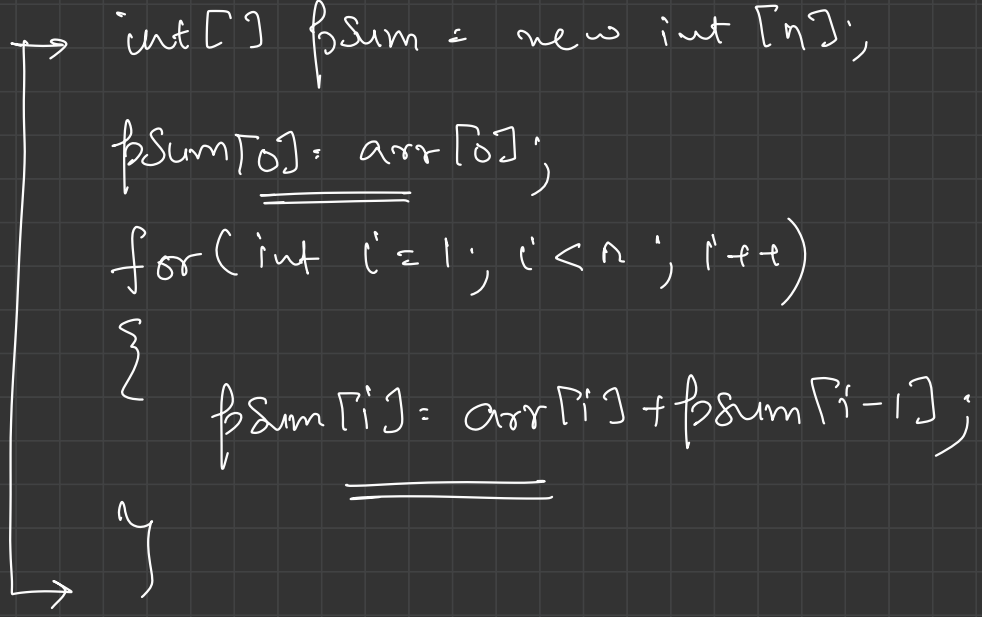


pSum[4] : arr[0] + arr[1] + arr[2] + arr[3] + arr[4]

pSum[3]

pSum[4] = pSum[3] + arr[4]

pSum[n] = arr[n] + pSum[n-1]


int[] pSum = new int[n];
pSum[0] = arr[0];
for(int i=1; i<n; i++)
{
pSum[i] = arr[i] + pSum[i-1];
}

arr[] : [1, 2, 3, 4, 5, 6]

psum[0] = arr[0]

psum[1] = arr[0] + arr[1] = psum[0] + arr[1]

psum[2] = arr[0] + arr[1] + arr[2] = psum[1] + arr[2]

⋮

psum[n] = psum[n-1] + arr[n]

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

LSum[] : [9, 12, 19, 25, 33, 34, 44] ⟶ O(N)
(0 1 2 3 4 5 6)

RSum[] = [44, 35, 32, 25, 19, 11, 10] ⟶ O(N)

```
for(int i=0 → n)
if(LSum[i] == RSum[i])
    i is equi position.
```

```
RSum[n-1] = a[n-1];
for(int i=n-2; i>=0; i--)
{
    RSum[i] = a[i] + RSum[i+1];
}
```

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

$a[] = [9, 3, 7, 6, 8, 1, 10]$ (indices 0 1 2 3 4 5 6)

totalSum = 44

$LSum[] = [9, 12, 19, 25, 33, 34, 44]$ (indices 0 1 2 3 4 5 6)

$RSum[] = [44, 35, 32$

{ lSum = 25

rSum = totalSum − lSum + a[i]

44 − 12 + 3 =

44 − 19 + 7

=

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

totalSum = 44

lSum = 0 9 12 19 25

rSum = 44 - 25 + 6 = 25

```java
static int findEquilibriumIndex(int[] a) {
    // Write code here
    int n = a.length;

    int totalSum = 0;
    for (int i = 0; i < n; i++) {
        totalSum += a[i];
    }

    int lSum = 0;
    for (int i = 0; i < n; i++) {
        lSum += a[i];
        int rSum = totalSum - lSum + a[i];

        if (rSum == lSum) {
            return i;
        }
    }

    return -1;
}
```

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

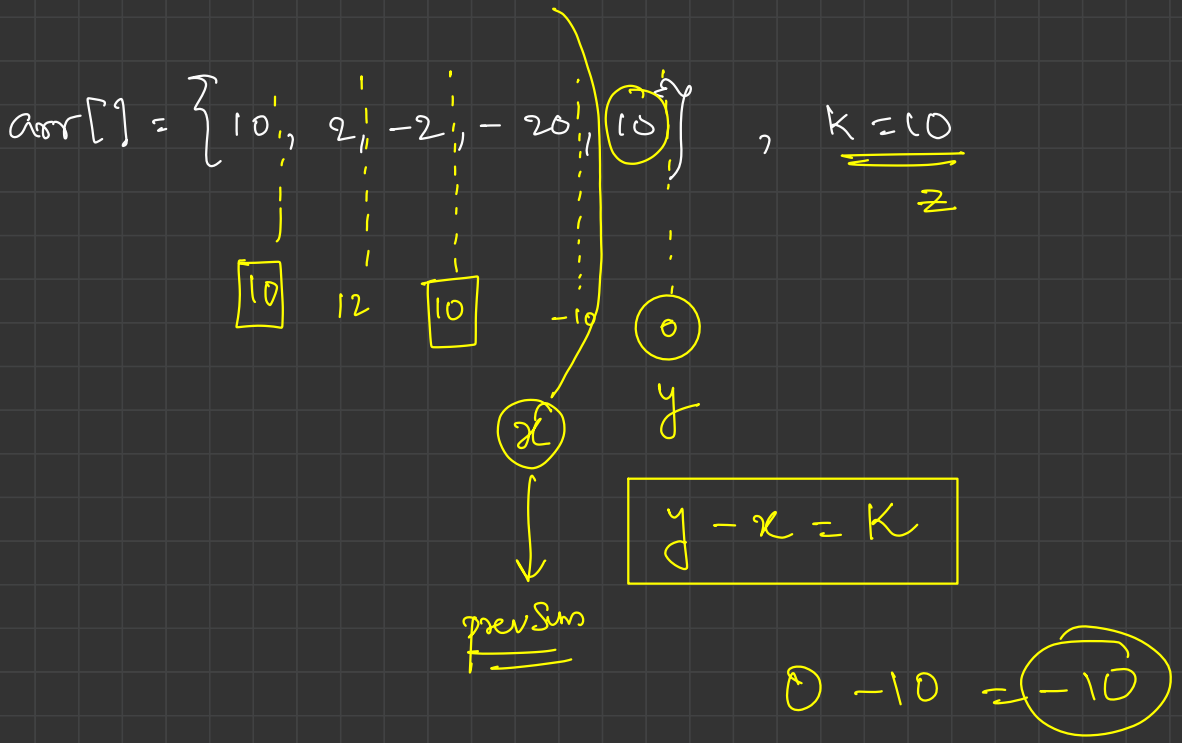
Subarray Sum equal k

arr[] = {10, 2, -2, -20, 10} , k = 10

3

Brute force: get all subarrays, and check sum == k

TC: O(N^2)


arr[] = {10, 2, -2, -20, 10}
, K = 10
2
10
12
10
-10
0
y
x
y - x = K
prevSum
0 - 10 = -10

int prevSum = Sum − k

arr[] = [1, 1, 1, 1, 1] (0 1 2 3 4)

K = 2

if (currSum == K)
    ans++;

else
{
    find (prevSum)
    ans += preSum
}

prev = ~~0~~ 2 = 3

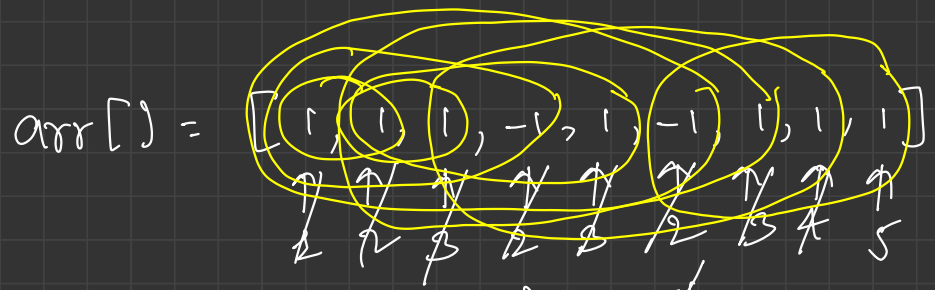
ans = ~~1~~ 2 ~~3~~ 4

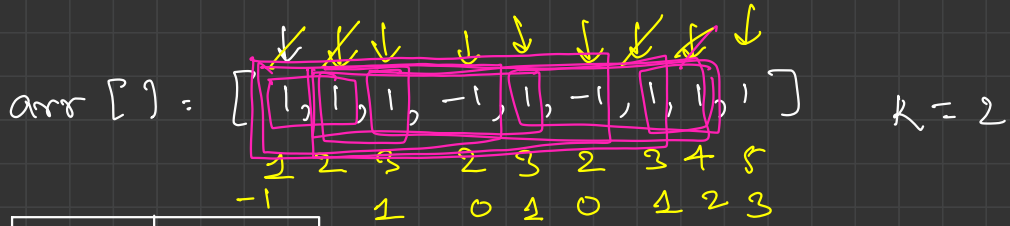
currSum = 1

Set: 1, 2, 3, 4,

currSum − prevSum = K

prevSum = currSum − K


arr[] = [1, 1, 1, -1, 1, -1, 1, 1, 1]
K = 2
1 2 3 2 3 2 3 4 5
pw = -1 1 2 3
ans = 1 2 3 4 5 6 7 8
1, 2, 3
4
Set



Sum	freq
0	1
1	1
2	3
3	3
4	1

ans = ~~1~~ ~~4~~ ~~3~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ (12)

arr[] = [1, 1, 1, -1, 1, -1, 1, 1, 1] K = 2

Sum	fr
0	1
1	1
2	3
3	3
4	1
5	2

```java
public class Main {
    static int solve(int n, int[] arr, int k) {
        // Write your code here
        HashMap<Integer, Integer> mp = new HashMap<>();
        mp.put(key: 0, value: 1);

        int ans = 0;
        int cSum = 0;
        for (int i = 0; i < n; i++) {
            cSum += arr[i];

            int prevSum = cSum - k;
            if (mp.containsKey(prevSum)) {
                ans += mp.get(prevSum);
            }

            mp.put(cSum, mp.getOrDefault(cSum, defaultValue: 0) + 1);
        }

        return ans;
    }
```

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