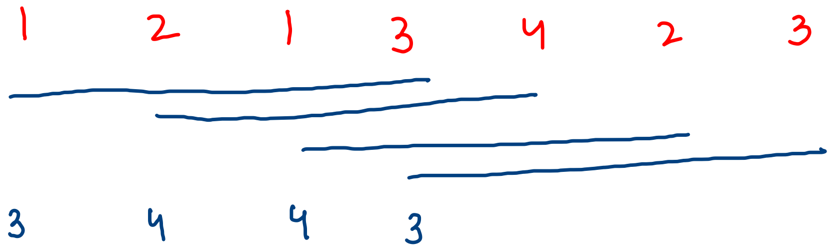
Count Distinct Elements In Every Window Of Size K

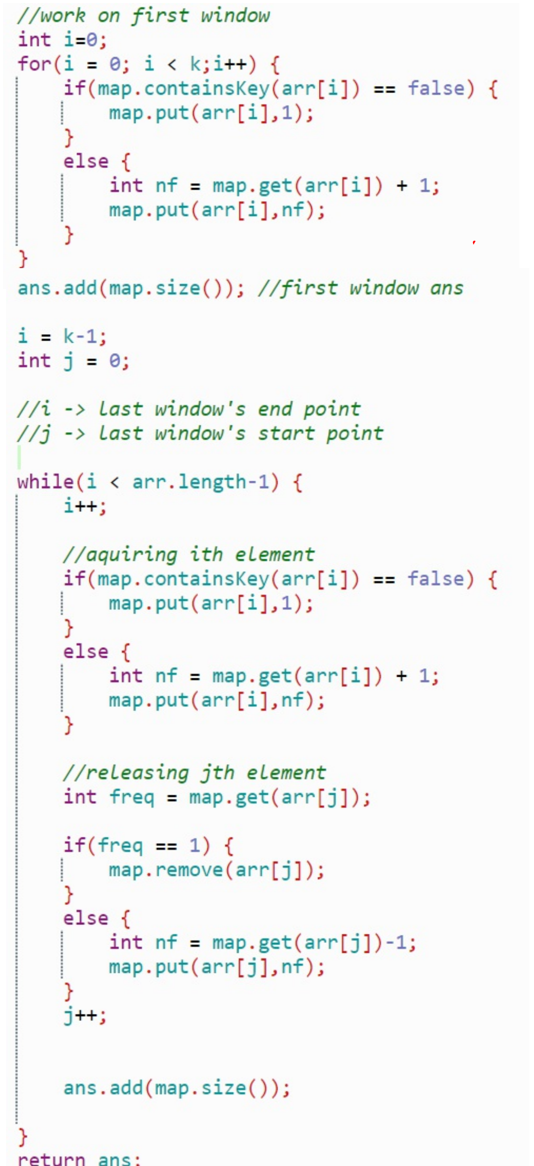
```
7
1 2 1 3 4 2 3
4
```



$k = 4$

$O(n)$

(i) aquire and release.



```java
//work on first window
int i=0;
for(i = 0; i < k;i++) {
    if(map.containsKey(arr[i]) == false) {
        map.put(arr[i],1);
    }
    else {
        int nf = map.get(arr[i]) + 1;
        map.put(arr[i],nf);
    }
}
ans.add(map.size()); //first window ans

i = k-1;
int j = 0;

//i -> last window's end point
//j -> last window's start point

while(i < arr.length-1) {
    i++;

    //aquiring ith element
    if(map.containsKey(arr[i]) == false) {
        map.put(arr[i],1);
    }
    else {
        int nf = map.get(arr[i]) + 1;
        map.put(arr[i],nf);
    }

    //releasing jth element
    int freq = map.get(arr[j]);

    if(freq == 1) {
        map.remove(arr[j]);
    }
    else {
        int nf = map.get(arr[j])-1;
        map.put(arr[j],nf);
    }
    j++;


    ans.add(map.size());

}
return ans;
```

K = 4

3 3 3 3 3 2

arr :	3	1	2	1	3	2	2	4	2
	0	1	2	3	4	5	6	7	8
						j			i

2 - 3

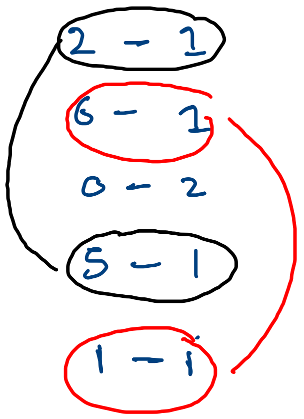
4 - 1

Check If An Array Can Be Divided Into Pairs Whose Sum Is Divisible By K

$k = 7$

	9	6	28	40	8	35
rem :	2	6	0	5	1	0

rem-freq-map



$n_1 \% k = x$ $\qquad$ $n_1 = k * n + \cancel{x}$

$n_2 \% k = k - x$ $\qquad$ $n_2 = k * m + k - \cancel{x}$

$$n_1 + n_2 = k(n + m + 1)$$

k = 8

	52	5	16	45	83	11	26	40	44	32	24	54
arr[i]%k $\downarrow$ rem	4	5	0	5	3	3	2	0	4	0	0	6

$\textcircled{4}$ - 2

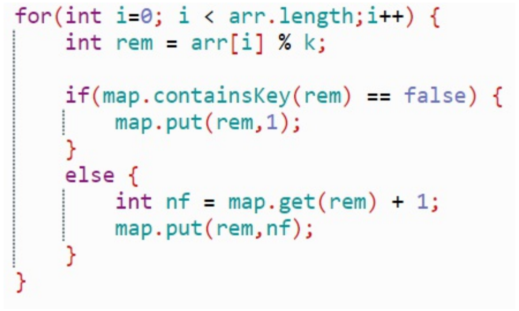
5 - 2

0 - 4

3 - 2

2 - 1

6 - 1

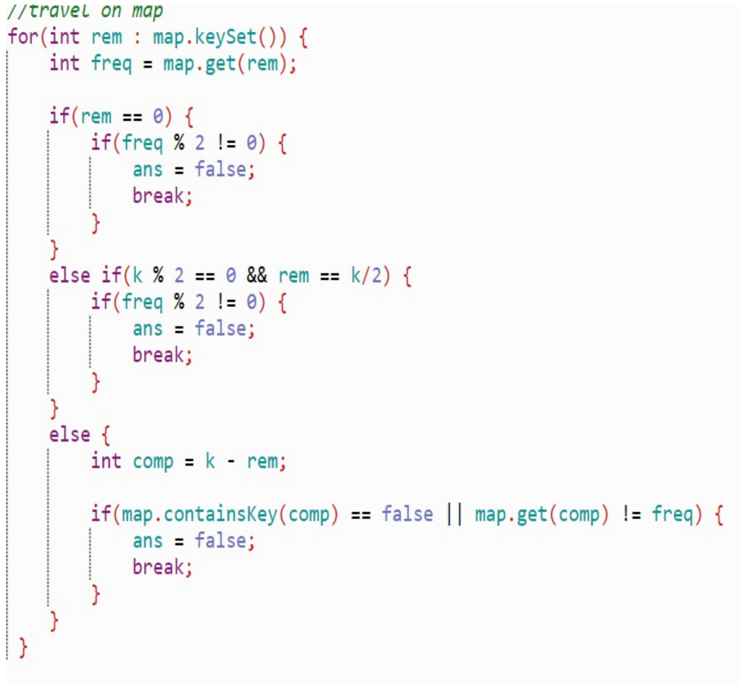


```
for(int i=0; i < arr.length;i++) {
    int rem = arr[i] % k;

    if(map.containsKey(rem) == false) {
        map.put(rem,1);
    }
    else {
        int nf = map.get(rem) + 1;
        map.put(rem,nf);
    }
}
```

k = 8

52 5 16 45 83 11 26 40 44 32 24 54



```
//travel on map
for(int rem : map.keySet()) {
    int freq = map.get(rem);

    if(rem == 0) {
        if(freq % 2 != 0) {
            ans = false;
            break;
        }
    }
    else if(k % 2 == 0 && rem == k/2) {
        if(freq % 2 != 0) {
            ans = false;
            break;
        }
    }
    else {
        int comp = k - rem;

        if(map.containsKey(comp) == false || map.get(comp) != freq) {
            ans = false;
            break;
        }
    }
}
```

4 — 2 ✓

5 - 2 ✓

0 - 4 ✓

3 — 2 ✓

2 — 1 ✓

6 — 1 ✓

ans = true

$k = 6$

38   18   7   16   -5   24   -13   -11

if (pos no.)

$n \to +ve$

$k * n + x$

if (neg no.)

$n \to -ve$

$k * n - y$

$\to k * n - y + k - k$

$\to k(n-1) + (k - y)$

```
if (rem -> -ve) {
    rem = rem + k;
}
```

$k = 6$

0	~~-6~~
1	-5
2	-4
3	-3
4	-2
5	-1

$k = 6$

$$\begin{tikzcd}
 & 38 & 18 \arrow[rrrr, bend left, dash] & 7 \arrow[rrrr, bend left, dash] & 16 & -5 \arrow[rrr, bend right] & 24 & -13 & -25 \\
\text{rem} & 2 \arrow[rrr, bend right, dash] & 0 & 2 & 4 & \cancel{-5} & 0 & \cancel{-1} & \cancel{-1} \\
 & & & & & +1 & & +5 & +5
\end{tikzcd}$$

$-25 = -24 - 1$

$-25 = -30 + 5$ ✓

$\underline{-25} \,\%\, 6 \to -1$

$\underline{-24} \underline{- 1 + 6} \underline{- 6}$

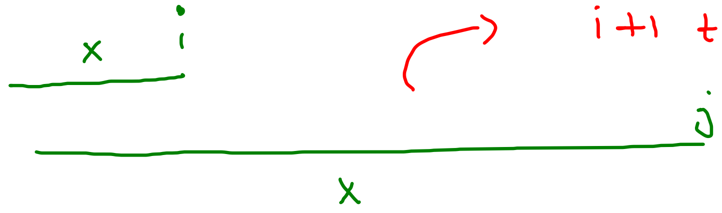
$-30 + 5$

Largest Subarray With Zero Sum

```
8
15 -2 2 -8 1 7 10 23
```

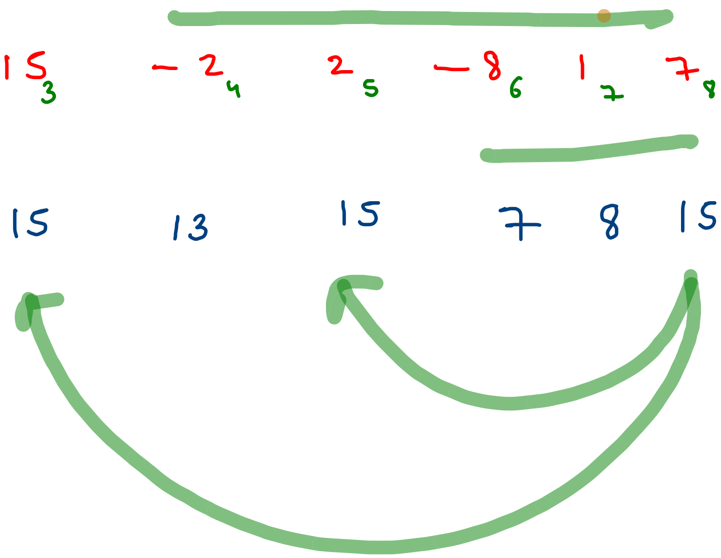
15	-2	2	-8	1	7	10	23
15	13	15	7	8	15	25	48

prefix_sum[i] == prefix_sum[j]



i+1 to j subarray sum = 0

	9_0	-1_1	-8_2	15_3	-2_4	2_5	-8_6	1_7	7_8	10_9	23_{10}
prefix_sum	9	8	0	15	13	15	7	8	15	25	48



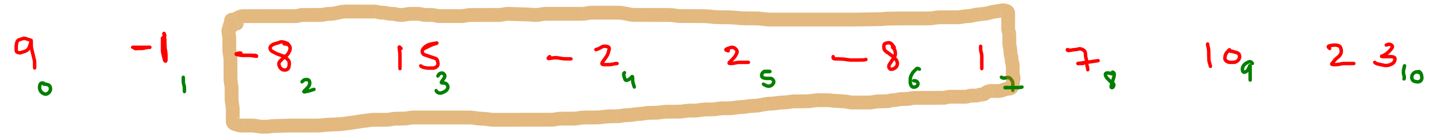
ans = 6

0	(-1)
9	0
8	1
15	3
13	4
7	6
25	9
48	10

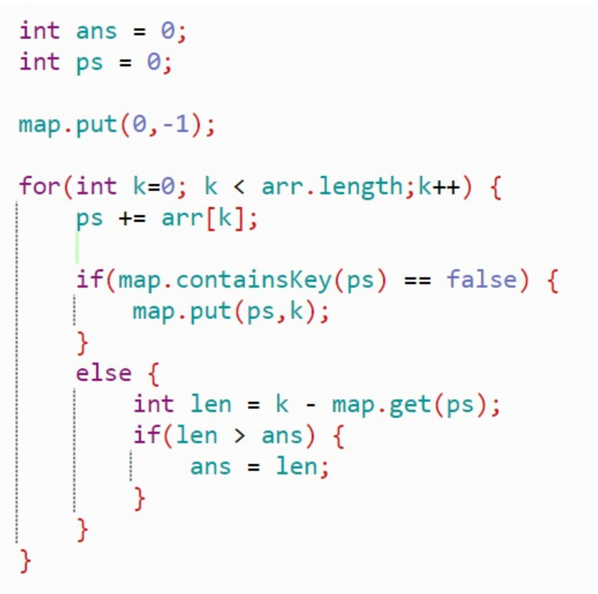
Hashmap

prefix_sum vs first index

ans = ~~0~~ ~~3~~ 6



	0	1	2	3	4	5	6	7	8	9	10
	9	-1	-8	15	-2	2	-8	1	7	10	23
ps	9	8	0	15	13	15	7	8	15	25	48



```
int ans = 0;
int ps = 0;

map.put(0,-1);

for(int k=0; k < arr.length;k++) {
    ps += arr[k];

    if(map.containsKey(ps) == false) {
        map.put(ps,k);
    }
    else {
        int len = k - map.get(ps);
        if(len > ans) {
            ans = len;
        }
    }
}
```

0 - (-1)
9 - 0
8 - 1
15 - 3
13 - 2
7 - 6
25 - 9
48 - 10

Count Of All Subarrays With Zero Sum

	9	-1	-8	15	-2	2	-8	1	7	10	-10	6
ps	9	8	0	15	13	15	7	8	15	25	15	21

0 - 2
9 - 1
8 - 2
15 - 4
13 - 1
7 - 1
25 - 1
21 - 1

Hash map

prefix_sum vs freq

count = 0 + 1 + 1 + 1 + 2 + 3

Longest Subarray With Equal Number Of Zeroes And Ones

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

replace 0 with -1 $\longrightarrow$ longest subarray with 0 sum.

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

0 - (-1)

-1 → 0

-2 → 3

1 → 8

2 → 9

len = ~~2~~ ~~4~~ ~~6~~ 8

ans = ~~0~~ ~~2~~ ~~4~~ ~~6~~ 8