Ques

Given arr[N] and K, check if there exists a pair(i, j) such that,

```
arr[i] + arr[j] == K && i != j
```

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

k = 6, → arr[2] + arr[5] → True

k = 22, → fa'lse

k = 8, arr[4] + arr[8] → True.

Brute force :- check all pairs, arr[4]

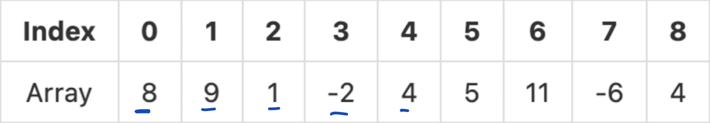
$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & \cancel{(0,0)} & (0,1) & (0,2) & (0,3) \\ 1 & (1,0) & \cancel{(1,1)} & (1,2) & (1,3) \\ 2 & (2,0) & (2,1) & \cancel{(2,2)} & (2,3) \\ 3 & (3,0) & (3,1) & (3,2) & \cancel{(3,3)} \end{array}$$

```
for i → 0 to n - 2
    for j → i+1 to n-1
        . . .
```

T.C → O(n²)

S.C → O(1).

idea 2 :- using hashset. ✗



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

k = 8

arr[i] ,	arr[j] = k - arr[i]
8	0
9	-1
1	7
-2	10
4	4

→ True.

8 9 1 -2
4 5 11 -6

```
for i → 0 to n-1
    other = k - arr[i];
    if (hset.contains(other)) {
        return True;
    }
}
return false
```

Edge Case :-

arr[] → 5, 7, 9, 2, 3

k = 4

k = 4

arr[i] ,	arr[j] = k - arr[i]
5	-1
7	-3
9	-5
2	2 → True .

5, 7, 9, 2, 3

idea 3 :- Optimization with Hashset :-

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

k = 9 .

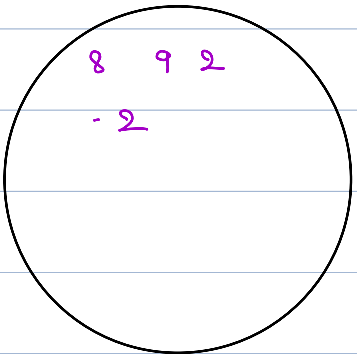
arr[i],	k - arr[i] .
8	1
9	0
2	7
-2	11
4	5
5	4 → True .

8 9 2
-2 4

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

k = 4.

arr[i],	k - arr[i].
8	-4
9	-5
2	2
-2	6



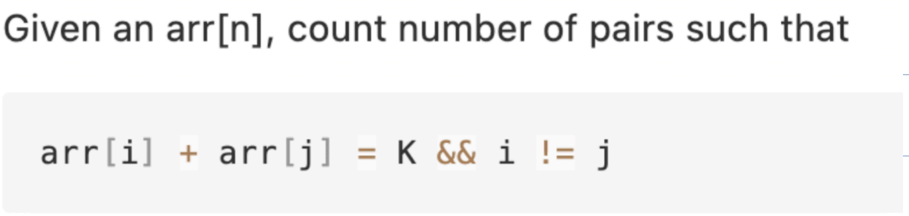
```
HashSet < int > hs;
for (i=0; i<n; i++) {
        other = k - arr[i];
        if (hs.check (other) ) {
            True;
        }
        hs.add(arr[i]);
}
return false;
```

T.C → O(n)
S.C → O(n).

idea 4 :- with hashmap. → Todo.

Ques   Counting all Pairs.

$$arr \rightarrow \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 3, & 5, & 1, & 2, & 1, & 2 \end{array}$$, k = 3 , Ans → 4 .



Given an arr[n], count number of pairs such that

```
arr[i] + arr[j] = K && i != j
```

k = 10 .

$$arr[8] = \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 5 & 2 & 5 & 8 & 5 & 2 & 8 \end{array}$$

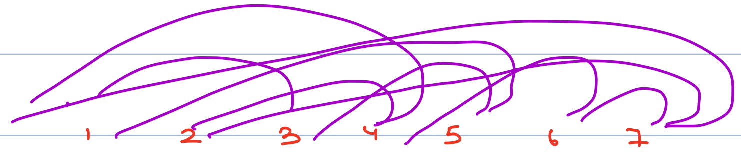
Pairs   Ans → 9

(0,4)   (1,3)   (2,4)   (3,5)   (6,7)

(0,7)   (1,5)   (2,7)   (4,6)

k = 10

arr[i]	comp
2	8 → 2



	0	1	2	3	4	5	6	7
arr[8] =	2	5	2	5	8	5	2	8
k = 10	+0	+0	+0	+1	+2	+2	+1	+3

arr[i]	comp
2	8
5	5
2	8
5	5
8	2
5	5
2	8
8	2

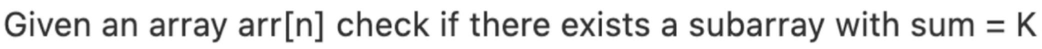
HashMap

2 → 1 ~~2~~ 3
5 → 1 ~~2~~ 3
8 → 1

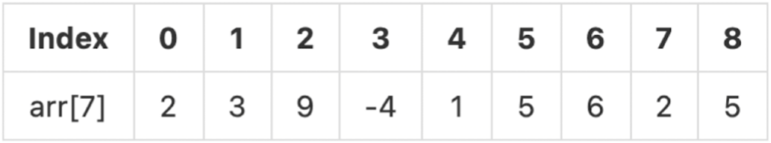
```
ans = 0;
HashMap <int, int> hm;
for (i = 0; i < n; i++) {
        other = k - arr[i];
        if ( hm. check (other)) {
                ans += hm. get(other);
        }
        if ( hm. check (arr[i])) {
                hm [arr[i]] += 1;
        }
        else {   hm [arr[i]] = 1;
        }
}
return ans;
```

T.C → O(n)
S.C → O(n)

Ques)



Given an array arr[n] check if there exists a subarray with sum = K



Index	0	1	2	3	4	5	6	7	8
arr[7]	2	3	9	-4	1	5	6	2	5

K=11, (5-6) or (2,3,9,-4,1)

K= 10, (2,3,9,-4)

K= 15, (-4,1,5,6,2,5)

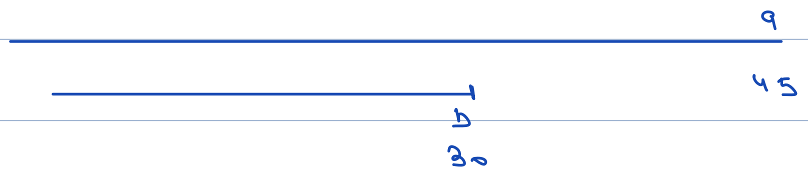
Brute force :- check all subarray sums.

idea 2 :- Optimized .

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

pf[i] → 0 to i .

k= 15



a-b = k . => a-k = b

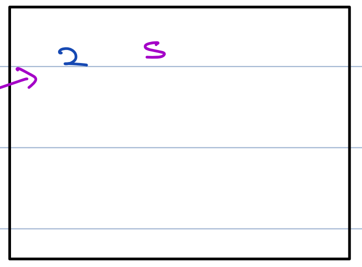
	0	1	2	3	4	5	6	7	8
arr[] →	2	3	9	-4	1	5	6	2	5
pf[] →	2	5	14	10	11	16	22	24	29

k = 12 .

a	a-k
2	-10
5	-7
14	2

→ True .

Hashset .



pf[R] - pf[L] = k

L+1 to R = k

```
pf = 0;
Hashset <int> hs;
hs. add(0)    // edge case (handles
                             subarray
                             starting
                             with 0)
for (i=0; i<n; i++) {
        pf = pf + arr[i];
        other = pf - k;
        if (hset. contains(other)) {
                return True;
        }
        hs.add (pf);
}
return false;
```

T.C → O(n)
S.C → O(n).

Edge Case :-

arr[] = 2, 3, 9, -4, 1
pf[] = 2 5 14 10 11

} add 0 to handle subarrays starting with 0.

k=11

a	a-k
2	-9
5	-6
14	3
10	-1
11	0

Hset

2 5 14
10 11,
0

Ques Distinct element in every window of size k.

$arr[] \rightarrow \{1, 2, 1, \overline{3, 4, 2, 3}\}$ $k = 4$

ans → 3, 4, 4, 3

start	end	len
x	$n-1$	k

$\Rightarrow n-1-x+1 = k$

$\Rightarrow x = n-k$.

idea ①

$arr[] \rightarrow \{\overset{0}{1}, \overset{1}{2}, \overset{2}{1}, \overline{\overset{3}{3}, \overset{4}{4}, \overset{5}{2}, \overset{6}{3}}\}$ $\boxed{k = 4}$

as → 3, 4, 4, 3

Hset

3, 4, 2

```
for(i=0 ; i<=n-k ; i++){
    Hashset<int> hs;
    for(j=0; j<k; j++){
        hs.add(arr[i+j])
    }
    print(hset.size());
}
```

$T.C \rightarrow (n-k+1) * k$

when, $k = \frac{n}{2}$, $\left(n - \frac{n}{2} + 1\right) * \frac{n}{2} \Rightarrow \left(\frac{n}{2} + 1\right)\frac{n}{2}$

$\Rightarrow \frac{n^2}{4} + \frac{n}{2} \Rightarrow O(n^2)$.

worst case.

$S.C \rightarrow O(k)$

idea 2

```
             0  1  2  3  4  5  6
arr[] →    { 1, 2, 1, 3 ,4, 2, 3 }      k = 4
```

3 4 4 3

Hashmap

1 → ~~2~~ ~~1~~ 0
2 → ~~1~~ ~~0~~ 1
3 → ~~1~~ 2
4 → 1

```
HashMap< int, int> hm;
    //[0  k-1]
    for ( i = 0; i < k; i++ ){
        if ( hm.contains (arr[i])) {
            hm[arr[i]] += 1;
        }
        else {
            hm[arr[i]] = 1
        }
    }                                   [0 → k - 1]

    Print (hm.size());

    s = 1
    e = k
    while ( e < n ) {
        hm[arr[s-1]] -= 1;
        if (hm[arr[s-1]] == 0){
            hm.remove (arr[s-1]);
        }
    if (hm.contains(arr[e])) {
        hm[arr[e]] += 1;
    }
    else {
        hm[arr[e] = 1
    }
    Print (hm.size());
    s++;
    e++;
}
```

T.C → K+n-k ⇒ O(n)

S.C → O(k)