Target sum Pair

Q.1 Given an array, find if there is a pair such that $A[i] + A[j] = K$ and $i \,!= j$.

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

K = 6 true

K = 22 false

	0	1	2	3	4	5
A [ ] =	8	9	11	5	2	11

K = 22 true

K = 14 true

K = 25 false

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ A[\,] = & 8 & 9 & 11 & 5 & 2 & 11 \end{array}$$

x = 6 5

k = 14

x = k - A[i]

```
boolean solve (int [] A, int k) {
    int n = A.length;
    for (int i = 0; i < n; i++) {
        int x = k - A[i];
        //search x in i+1 to n-1
        for (int j = i+1; j < n; j++) {
            if (A[j] == x) {
                return true;
            }
        }
    }
    return false;
}
```

TC: $O(N^2)$

SC: $O(1)$

K = 12

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A = & 9 & 5 & 1 & 7 & 6 \end{array}$$

x = 7

K = 4

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ A = & 9 & 2 & 1 & 2 \end{array}$$

x = 2

(0,1) (0,2), (0,3)

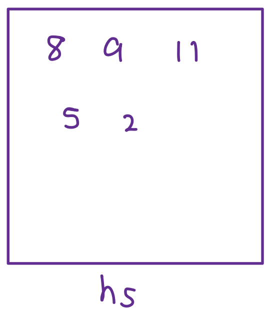
(1,2) (1,3)

(2,3)

idea2 : using Hashset

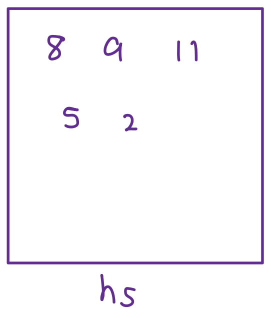
K = 14

	0	1	2	3	4	5
A [ ] =	8	9	11	5	2	11
x :	6	5				



K = 4

	0	1	2	3	4	5
A [ ] =	8	9	11	5	2	11
x :	-4	-5	-7	-1	2	



Don't fill hashset with all array values at once and then do the work.

hashset will contain the impact of left always (0, i-1)

	0	1	2	3
A [ ] =	8	9	11	5

idx	left		
0	–		
1	0	→	(1, 0)
2	0, 1	→	(2, 0) , (2, 1)
3	0, 1, 2	→	(3, 0) (3, 1) (3, 2)

```
boolean solve (int[ ]A, int k) {
    int n= A.length;
    Hashset<Integer> hs= new Hashset<>();

    for (int i=0; i< A.length; i++) {
    |       int x= k- A[i];
    |       if ( hs.contains(x) == true) {
    |               true;
    |       }
    |       hs.add (A[i]);
    }

    return false;
}
```

TC : O(N)

SC : O(N)

```
boolean  solve (int[] A, int k) {
    int  n= A.length;
    HashSet<Integer> hs= new HashSet<>();

    for (int i=0; i< A.length; i++) {
            int  x= k- A[i];
            if (hs.contains(x) == true) {
                    true;
            }
            hs.add (A[i]);
    }
    return false;
}
```

$K = 12$

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 9 & 4 & 6 & 1 & 2 & \overset{\swarrow}{6} \end{array}$$

$X = 6$

$$\boxed{\begin{array}{ccc} 9 & 4 & 6 \\ 1 & 2 & \end{array}}$$

hs

return true

```
boolean  solve (int[] A, int k) {
    int  n= A.length;
    HashSet<Integer> hs= new HashSet<>();

    for (int i=0; i< A.length; i++) {
            int  x= k- A[i];
            if (hs.contains(x) == true) {
                    true;
            }
            hs.add (A[i]);
    }
    return false;
}
```

$K = 4$

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 9 & 5 & 6 & 1 & 2 & \underset{\uparrow}{6} \end{array}$$

$X = -2$

$$\boxed{\begin{array}{cc} 9 & 5 \\ 6 & 1 \\ 2 & \end{array}}$$

hs

return false

Q.2 Given an array, count no. of pairs such that

$A[i] + A[j] = K$ and $i \,!= j$.

↙

	0	1	2	3	4	5	6
A[] =	2	5	1	5	2	* 7	10
x :	10	7	11	7	10	5	2

$K = 12$

count = 2 + 2

$2 \rightarrow 2$
$5 \rightarrow 2$
$1 \rightarrow 1$
$7 \rightarrow 1$
$10 \rightarrow 1$

map

Previous approach with hashmap (x is coming how many times on left)

```
int  countPairs (int [] A, int k) {

    int n= A.length;
    HashMap< Integer, Integer> map = new  HashMap<>();
     int count = 0;

    for (int  i=0; i<n; i++) {

         int  x= k - A[i];
         // how many times x is present in left
         if (map.containsKey(x) == true) {
                count += map.get(x);
         }

         // put your impact in map
         if(map.containsKey(A[i]) == false) {
                map.put(A[i], 1);
         }
         else {
                int t= map.get(A[i]);
                t++;
                map.put(A[i], t);
         }
    }

    return count;
}
```

TC: O(n)

SC: O(n)

```
for (int  i=0; i<n; i++) {
    int  x = k - A[i];
    // how many times  x  is present in left
    if (map.containsKey(x)  == true) {
            count += map.get(x);
    }
    // put your impact in map
    if (map.containsKey(A[i]) == false) {
            map.put(A[i], 1);
    }
    else {
            int  t = map.get(A[i]);
            t++;
            map.put(A[i], t);
    }
}
```

k = 12

A = 5 2 5 7 10

X : 7 10 7 5 2

$5 \to 2$

$2 \to 1$

$7 \to 1$

$10 \to 1$

map

Count = 2 + 1

Q.3 Given an array, check if there is a subarray with sum k

↳ continous part of an array.

A[] =	0	1	2	3	4	5	6
	3	9	-4	1	5	6	2

K = 11 ans: true

K = 10 ans: true

idea1 : brute force

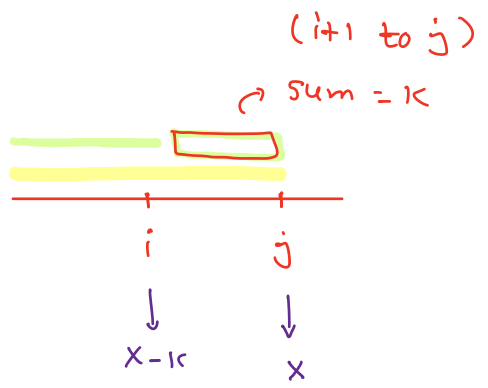
→ go on every subarray s to e and find sum (prefix sum / cf)

TC: O(n²)

Expected TC: O(n)

A[] =	0	1	2	3	4	5	6
	3	9	-4	1	5	6	2
PS :	3	12	8	9	14	20	22

K = 11



	0	1	2	3	4	5	6
A[] =	3	9	-4	1	5	6	2
sum =	3	12	8	9	14		
sum-K =	-8	1	-3	-2	3		

K = 11

	0	1	2	3	4	5	6
A[] =	3	9	-4	1	5	6	2
sum	3	12	8	9	14	20	22
sum-K	-7	2	-2	-1	4	10	12

K = 10

	-1	0	1	2	3	4
A[] =	↓	3	9	-5	4	6
sum :	0	3	12	7		
sum-K :		-4	5	0		

K = 7

Maintain a hashset to store sum value (left side impact)

```
boolean solve (int[] A, int K) {
   int n= A.length;
   Hashset <Integer> hs= new HashSet<>();
   int sum = 0;
   hs.add(0);

   for(int i= 0; i<n; i++) {
       sum+= A[i];
       if (hs.contains(sum-k) == true) {
           return true;
       }
       hs.add(sum);
   }

   return false;
}
```

K = 8

	0	1	2	3	4
A =	2	1	5	-3	7

Sum: 0 2 3 8

sum-K = 0
(return true)

$\boxed{0 \quad 2 \quad 3}$

hs

Dry run

```
boolean solve (int[] A, int k) {
   int n= A.length;
   HashSet<Integer> hs= new HashSet<>();
   int sum=0;
   hs.add(0);

   for(int i=0; i<n; i++) {
      sum+=A[i];
      if(hs.contains(sum-k)==true) {
            return true;
      }
      hs.add(sum);
   }

   return false;
}
```

$k = 6$

	0	1	2	3	4	5
A =	2	5	3	-1	4	10

Sum: 0 2 7 10 9 13

sum-k = 7

return true

$$\boxed{\begin{matrix} 0 & 2 & 7 \\ 10 & 9 & \end{matrix}}$$

hs

Q.4 Given an array, count total no. of subarray with sum K

$$A[\,] = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 9 & -4 & 1 & 5 & 6 & 2 \end{array}$$

K = 6 ans = 3

$$\left.\begin{array}{l} (1,3) \\ (3,4) \\ (5,5) \end{array}\right\} (s,e)$$

$$\begin{array}{lccccccccc} & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A[\,] & = & 3 & 11 & -4 & 1 & -2 & 5 & {}^{*}6 & 2 \\ \text{Sum} & : & 3 & 14 & 10 & 11 & 9 & 14 & 20 & 22 \end{array} \qquad K = 6$$

$$20 \downarrow \quad \text{sum} - K = 14$$

map → sum vs freq

↙

$$\begin{array}{lccccccccc} & & \times & & & & \times & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\ A[\,] = & 3 & 11 & -4 & 1 & -2 & 5 & 6 & 2 & \end{array}$$

K = 6

Sum : 0 3 14 10 11 9 14 20 22

sum - k = 16

count = 1 + 2

0 → 1
3 → 1
14 → 2
10 → 1
11 → 1
9 → 1
20 → 1

map

(sum vs freq)

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A = & 6 & -6 & 4 & 2 & 6 \end{array}$$

K = 6

sum : 0 6 0 4 6 12

sum - K = 6

count = 1 + 2 + 2

0 → 2
6 → 2
4 → 1

map

Common elements

A = 3 1 2 3 5

B = 5 1 3 3 3

↑

ce : 5 1 3 3

$3 \rightarrow \cancel{2}\ \cancel{1}\ 0$

$1 \rightarrow \cancel{1}\ 0$

$2 \rightarrow 1$

$5 \rightarrow \cancel{1}\ 0$

map

i) create a freq map using 1st Array

ii) travel the 2nd array and find common ele and also do req. changes in map.

for- each

1)

```
int [ ] A = {16, 15, 19, 20};

for( int ele : A ) {
        SOPln (ele);
  }
```

2) hs :

```
┌────────────────┐
│  10      19    │
│     15         │
└────────────────┘
```

```
for( int ele : hs ) {
        SOPln (ele);
  }
```

map.keySet ⇒ ["India", "Aus" "England"]

3)

```
"India" → 270
"Aus" → 320
"England" → 189
```

```
for (String s : map.keySet()) {
        int val = map.get(s);
        SOPln ( s + " → " + val);
}
```