Target sum Pair

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

$$A[\,] = \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 8 & 9 & 1 & -2 & 4 & 5 & 11 & -6 & 4 \end{array}$$

K = 6 ans: true

K = 22 ans: false

K = 8 ans: true

idea1 : travel all pairs

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

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

(1,2) (1,3)

(2,3)

i	j
0	1 to 3
1	2 to 3
2	3 to 3
3	4 to 3 ×

A = 3 9 4 7
    0 1 2 3

K = 16

A[i]	x = K - A[i]	
3	13	
9	7	(true)

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 for x in i+1 to n-1
        for (int j= i+1 ; j<n; j++) {
            if ( A[j] == x) {
                return true;
            }
        }
    }

    return false;
}
```

K = 8

A = 3 9 4 6 4
    0 1 2 3 4

A[i]	x = K - A[i]	
3	5	(1 to 4)
9	-1	(2 to 4)
4	4	(3 to 4)

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

k = 12

A =	5	9	4	6	4
	0	1	2	3	4

A[i]	x = k - A[i]
5	7 (1 to 4)
9	3 (2 to 4)
4	8 (3 to 4)
6	6 (4 to 4)
4	8 (5 to 4)

return false

TC: $O(n^2)$

SC: $O(1)$

idea2: Hash Set

	0	1	2	3	4	5	6	7	8
A[] =	8	9	1	-2	4	5	11	-6	4
x = K - A[i]	10	9							

return true ✗

K = 18

8 9 1
-2 4 5
11 -6

hs

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

$x = K - A[i]$

i	left	
0	–	
1	0 →	(1, 0)
2	0, 1 →	(2, 0) (2, 1)
3	0, 1, 2 →	(3, 0) (3, 1) (3, 2)
4	0, 1, 2, 3 →	(4, 0) (4, 1) (4, 2) (4, 3)

$$A = \begin{array}{ccccccc} & & & & & \downarrow & \\ 4 & 9 & 4 & 6 & 7 & 6 & 8 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \qquad K = 12$$

$x = K - A[i]$

$x = 6$

return true

$$\boxed{\begin{array}{cc} 4 & 9 \\ 6 & 7 \end{array}}$$

hs

A = 2 -3 2 5 4   K = 10

0 1 2 3 4

$x = K - A[i]$

$x = 6$

2	-3	5
4		

Note: only keep the impact of left side in hashset.

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

    for (int i=0; i<n; i++) {
        int x = K - A[i];
        // find x in left side
        if( hs.contains(x) == true) {
            return true;
        }
        hs.add (A[i]);
    }
    return false;
}
```

TC: O(N)

SC: O(N)

Dry run

k = 14

$A = \begin{array}{ccccc} 5 & 10 & 7 & -2 & 8 \\ 0 & 1 & 2 & 3 & 4 \end{array}$

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

    for (int i = 0; i < n; i++) {
        int x = k - A[i];
        // find x in left side
        if( hs.contains(x) == true) {
            return true;

        }
        hs.add (A[i]);
    }
    return false;
}
```

x = 6

return false

5	10
7	-2
8	

hs

k = 14

$A = \begin{array}{ccccc} 5 & 10 & 7 & -2 & 7 \\ 0 & 1 & 2 & 3 & 4 \end{array}$

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

    for (int i = 0; i < n; i++) {
        int x = k - A[i];
        // find x in left side
        if( hs.contains(x) == true) {
            return true;

        }
        hs.add (A[i]);
    }
    return false;
}
```

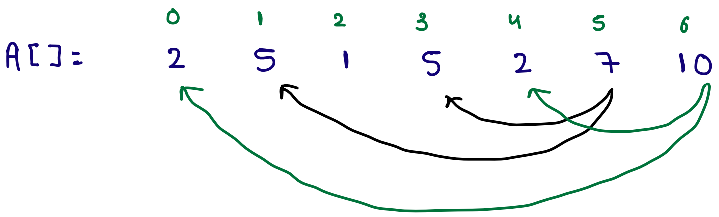
x = 7

return true

5	10
7	-2

hs

Q.2 Given an array, count no. of pairs such that $A[i] + A[j] = K$ and $i != j$.



K = 12

X = 2

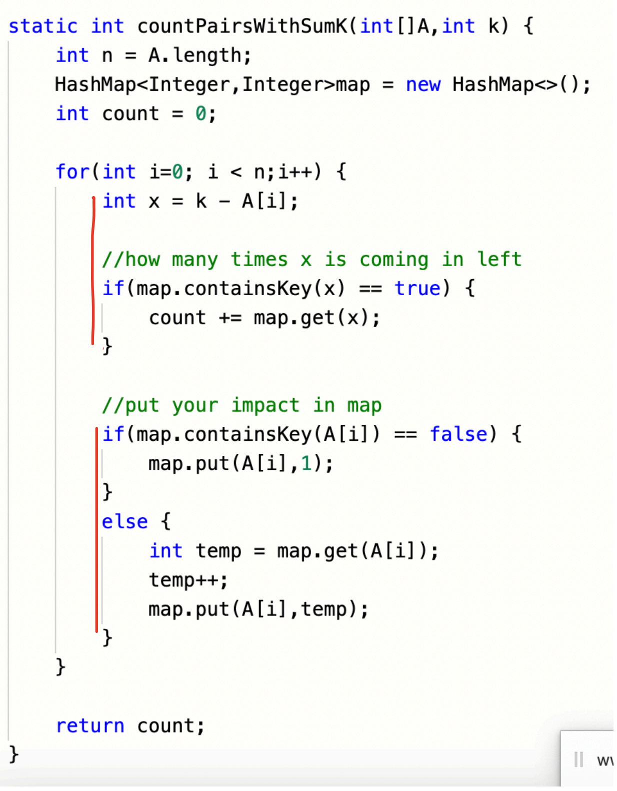
$X = K - A[i]$

count = 0 + 2 + 2

2 → 2
5 → 2
1 → 1
7 → 1
10 → 1

map
(ele vs freq)

left side



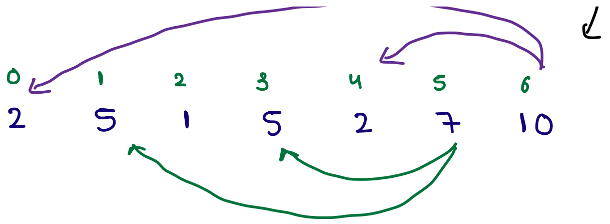
```java
static int countPairsWithSumK(int[]A,int k) {
    int n = A.length;
    HashMap<Integer,Integer>map = new HashMap<>();
    int count = 0;

    for(int i=0; i < n;i++) {
        int x = k - A[i];

        //how many times x is coming 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 temp = map.get(A[i]);
            temp++;
            map.put(A[i],temp);
        }
    }

    return count;
}
```



K = 12

X = K - A[i]

X = 2

Count=0+2+2

2 → 2
5 → 2
1 → 1
7 → 1
10 → 1

map

TC: O(n)

SC: O(n)

Q.3 Given an array, Check if there is a subarray with sum K

↳ continous part of an array.

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

Expected TC: O(n)

K = 11 ans = true

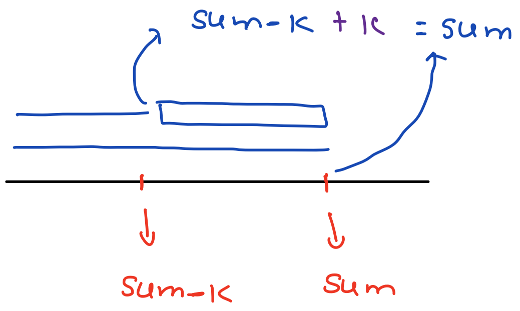
K = 10 ans = true

K = 13 ans = true

K = 25 ans = false

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A[\,] = & 3 & 9 & -4 & 1 & 5 & 6 & 2 \\ \text{sum} & 3 & 12 & 8 & 9 & 14 & 20 & 22 \end{array}$$

K = 11



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

K = 11

i	sum	sum - K
0	3	-8
1	12	1
2	8	-3
3	9	-2
4	14	3

return true

3 12 8
9

hs
(past journey
of prefix sum's)

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

K = 10

i	sum	sum - K
0	3	-7
1	12	2
2	8	-2
3	9	-1
4	14	4
5	20	10
6	22	12

→ return true

3 12 8
9 14 20

past journey
of prefix sum's

		0	1	2	3	4
A[] =		2	5	-1	10	3
sum :	0	2	7	6	16	19

K = 6

```
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;
}
```

TC: O(n)

SC: O(n)

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= 10

	0	1	2	3	4	5
A =	2	-3	-1	6	5	4
sum: 0	2	-1	-2	4	9	

sum-k

= -1

return true

hs

0 2 -1

-2 4

```
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 = 7

	0	1	2	3	4
A =	5	3	-1	4	10
sum: 0	5	8	7		

sum-k

= 0

return true

0 5

8

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

↙

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

K = 6

Sum : 0 3 14 10 11 9 14 20 22

sum - k = 16

Count = 1 + 2

TODO : code

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

map
(past jouney of prefix sum's with freq)

Doubts

		0	1	2	3	4
A[] =		8	-8	6	2	8
sum:	0	8	0	6	8	16

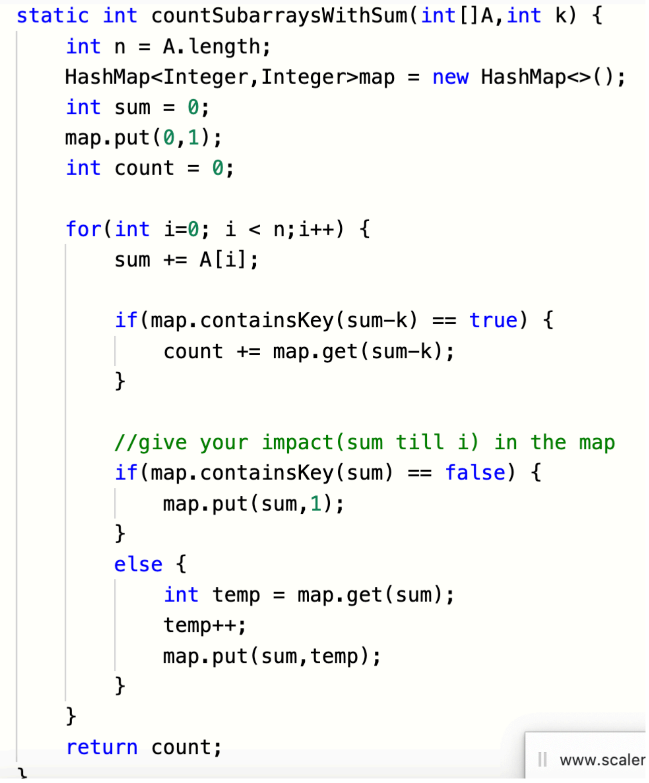
K = 8

0 → 2
8 → 2
6 → 1
16 → 1

map

sum-k = 8

count = 1 + 2 + 2



```
static int countSubarraysWithSum(int[]A,int k) {
    int n = A.length;
    HashMap<Integer,Integer>map = new HashMap<>();
    int sum = 0;
    map.put(0,1);
    int count = 0;

    for(int i=0; i < n;i++) {
        sum += A[i];

        if(map.containsKey(sum-k) == true) {
            count += map.get(sum-k);
        }

        //give your impact(sum till i) in the map
        if(map.containsKey(sum) == false) {
            map.put(sum,1);
        }
        else {
            int temp = map.get(sum);
            temp++;
            map.put(sum,temp);
        }
    }
    return count;
```

Doubts

Common element

A = 5 2 1 2 1 6

B = 1 3 1 2 1 5

1 1 2 5

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

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

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

$6 \rightarrow 1$

i) Create freq map using 1st Array.

ii) travel 2nd array, find common ele ~~and~~ do needful changes in map.