Q → Given an integer array A[N], check if there exist a pair (i, j) s.t. (A[i] + A[j] == K) & (i != j)

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

K = 6 Ans = <u>true</u> 1 + 5 = 6

K = 22 Ans = <u>false</u>

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [\,3 & \underline{5} & 1 & \underline{2} & 1 & 2\,] \end{array} \quad K = 7 \quad \text{Ans} = \underline{\text{true}}$$

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [\,3 & 5 & 1 & 2 & 1 & 2\,] \end{array} \quad K = 10 \quad \text{Ans} = \underline{\text{false}}$$

<u>Bruteforce</u> → ∀ i, j (i < j) check sum = K.

TC = <u>O(N²)</u> SC = <u>O(1)</u>

<u>Sol</u> → A[i] + A[j] == K

1) Only iterate on 1 index (let say 'i')

2) Check if <u>A[j]</u> is present in the array.

A[j] = K − A[i]

→ ∀i, <u>check</u> if (K − A[i]) is present in the array.

<u>Hashset</u>

Steps → 1) Insert ∀i, A[i] in hashset.

2) ∀i, check if K − A[i] is present in hashset.

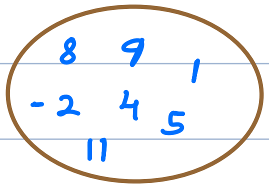
$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ [\,8 & 9 & 1 & 2 & 4 & 5 & 11\,] \\ \cancel{i} & \cancel{i} & i & & & & \end{array}$$

K = 2

check → 2-8 = -6 ✗

2-9 = -7 ✗

2-1 = 1 ✓ present in hashset ⇒ Ans = true ✗



K - A[i] = A[i]

check on values with index

↳ (j < i)

⇒ Hashset contains index from index 0 to (i-1).

~~i~~ ~~i~~ ~~i~~ i

0 1 2 3 4 5 6

A = [8 9 1 2 4 5 11]

K = 2

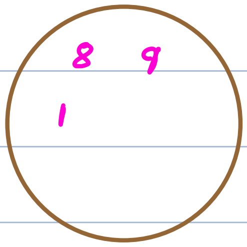
check → 2-8 = -6

2-9 = -7

2-1 = 1

2-2 = 0 . . .

Hashset



```
for i → 0 to (N-1) {
    if (hs.contains(K - A[i]))
        return true
    hs.add(A[i])
}
return false
```

TC = O(N) SC = O(N)

Q → Count the # pairs with sum = K (j < i).

0 1 2 3 4 5

A = [3 5 1 2 1 2] K = 3

Ans = 4 (2,3) (3,4)

(2,5) (4,5)

Bruteforce → TC = O(N²)   SC = O(1)

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

K = 10

10 - 8 = 2

store freq of A[i]

⇒ Hashmap

```
cnt = 0
for i → 0 to (N-1) {
    if (hm.containsKey(K - A[i])) {
        freq = hm.get(K - A[i])
        cnt += freq
    }
    if (hm.containsKey(A[i]))
        hm.put(A[i], hm.get(A[i]) + 1)
    else    hm.put(A[i], 1)
}
                                // hm.getORDefault(A[i], 0)
return cnt
```

TC = O(N)   SC = O(N)

$$\begin{array}{cccccccc} & & \cancel{i} & \cancel{i} & \cancel{i} & \cancel{i} & \cancel{i} & i \\ & & 0 & 1 & 2 & 3 & 4 & 5 \\ A & = [& 3 & 5 & 1 & 2 & 1 & 2] \end{array} \qquad K = 3$$

check → 0 -2 2 1 2 1

cnt = ~~0~~ + ~~2~~ 4

(3, 1)

(5, 1)   (1, 2) ✓

(2, 2)

HashMap

Q → Given an integer array, check if there exists a subarray with sum K.

$$A = [\overset{0}{\boxed{2}}\;\overset{1}{3}\;\overset{2}{9}\;\overset{3}{-4}\;\overset{4}{1}\;\;\overset{5}{5}\;\overset{6}{6}]$$

$$A = [\boxed{2 \quad 3 \quad 9 \quad -4 \quad 1} \quad \boxed{5 \quad 6}]$$

K = 11   Ans = <u>true</u>

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A = [& 5 & 10 & 20 & 100 & 105\,] \end{array}$$

K = 110   Ans = <u>false</u>

subarray sum → prefix sum

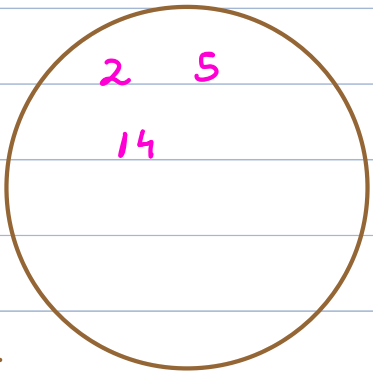
sum i — j $\boxed{i <= j}$ $P[j] - P[i-1] = K$ OR $P[j] = K$

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [& 2 & 3 & 9 & -4 & 1 & 5 & 6\,] \\ P = [& 2 & 5 & 14 & 10 & 11 & 16 & 22\,] \\ & \cancel{i} & \cancel{i} & \cancel{i} & i & & & \end{array}$$

K = 10

$P[j] - P[i-1] = K$ left   right

$\Rightarrow P[j] = K + P[i-1] \Rightarrow \boxed{P[i-1] = P[j] - K}$



<u>Hashset</u>

check → 2 - 10 = -8

5 - 10 = -5

14 - 10 = 4   (P[3] == K) ⇒ Ans = <u>true</u>

```
p = 0
for i → 0 to (N-1) {
    p += A[i]
    if ( p == K || hs.contains(p - K))
        return true
    hs.add(p)
}
return false
```

TC = $O(N)$ SC = $O(N)$

Q → Given an integer array, find the count of distinct elements in every window of size K.

$$\begin{array}{cccccccc} & & & & (K-1) & & & (N-1) \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [& 1 & 2 & 1 & 3 & 4 & 2 & 3] \end{array}$$

K = 4

3 4 4 3 (Ans)

Bruteforce → ∀ subarray of length K count the # distinct elements.

TC = $O(K)$

SC = $O(K)$ — Hashset

subarrays of length K = $N-K+1$

$[(K-1) \quad (N-1)]$

$\Rightarrow (N-1)-(K-1)+1$

TC = $O((N-K+1)*K)$ → worst case

$(K = \frac{N}{2}) \rightarrow O(N^2)$

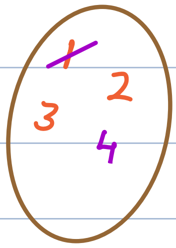
Fixed length subarray → sliding window

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [& \cancel{1} & 2 & 1 & 3 & 4 & 2 & 3] \end{array}$$

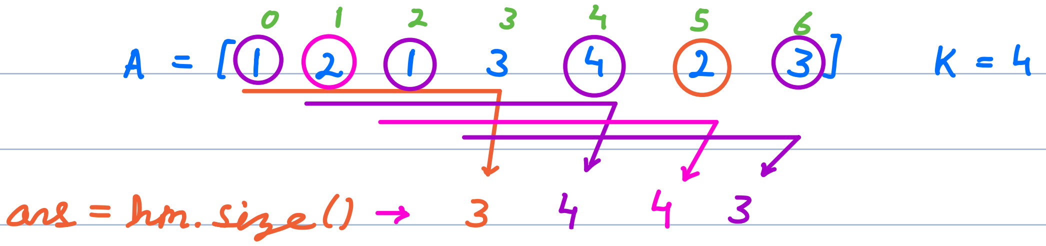
K = 4

ans = hs.size() → 3 ~~3~~ 4

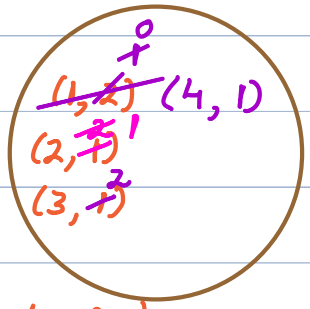
Hashset



Hashmap ← { keep track of # times any element is inserted.



remove element
with frequency 0



Hashmap

(A[i], freq of A[i])

```
for i → 0 to (K-1) {
        f = hm.getOrDefault(A[i], 0)
        hm.put(A[i], f+1)
}

ans.add(hm.size())
for i → K to (N-1) {
        f = hm.getOrDefault(A[i], 0)
        hm.put(A[i], f+1)
        // Remove → A[i-K]
        if ( hm.get(A[i-K]) == 1)
                hm.remove(A[i-K])
        else
                hm.put(A[i-K], hm.get(A[i-K]) - 1)
        ans.add(hm.size())
}
return ans
```

TC = O(N)   SC = O(K)