Q → Find the length of longest subarray with sum = 0.

No subarray → Ans = -1

0 1 2 3 4 5 6 7 8

A = [3 3 4 -5 -2 2 1 -3 3]

P = [3 6 10 5 3 5 6 3 6]

subarrays

[1 4] → 4

[1 7] → 7

[5 7] → 3

[2 6] → 5

[2 8] → 7

[7 8] → 2

[4 5] → 2

Ans = 7

→ (P[i] == P[j]) ⇒ subarray (i+1) to j has sum = 0

[L R] → R - L + 1

i

0 1 2 3 4 5 6 7 8

A = [3 3 4 -5 -2 2 1 -3 3]

P = [3 6 10 5 3 5 6 3 6]

i

3 ← _ _ _ _ _ _ _ 3 _

ignore all 3's in between.

0 1 2 3 4 5 6 7 8

A = [3 3 4 -5 -2 2 1 -3 3]

P = [3 6 10 5 3 5 6 3 6]

L → R

P[i] == P[j]

i	j
0	4
3	5
1	6
0	7
1	8

∀P[i] → store first index where P[i] is present.

HashMap

< P[i] , first index >

0 1 2 3 4 5 6 7 8

A = [3 3 4 -5 -2 2 1 -3 3]

P = [3 6 10 5 3 5 6 3 6]

```
ans = -1
for i → 0 to (N-1)
    if (hm.containsKey(P[i]))
        l = hm.get(P[i]) + 1
        r = i
        ans = max(ans, r - l + 1)
    else
```

i	Map	length
0	3 → 0	
1	6 → 1 ✓	
2	10 → 2	
3	5 → 3	

```
        hm.put(P[i], i)
```

$$\begin{array}{cccccc} & 0 & 1 & 2 & 3 & \overset{i}{4} \\ A = [& 4 & -3 & -1 & 2 & -2] \\ P = [& 4 & 1 & 0 & 2 & \overset{}{0}] \end{array}$$

↑ Ans = $\underline{5}$ ✓

4
5
6
7
8

[1 4] → $\underline{4}$
[4 5] → $\underline{2}$
[2 6] → $\underline{5}$
[1 7] → $\underline{7}$
[2 8] → $\underline{7}$

```
ans = -1
P[0] = A[0]
if (P[0] == 0)    ans = 1
for i → 1 to (N-1)
    P[i] = P[i-1] + A[i]
    if (P[i] == 0)
        ans = i+1
for i → 0 to (N-1)
    if (hm.containsKey(P[i]))  ← ✓
        l = hm.get(P[i]) + 1 = 1+1 = 2
        r = i                  = 2
        ans = max(ans, r - l + 1)
                   2 - 2 + 1 = 1 ✓
    else
        hm.put(P[i], i)
return ans
```

$$\begin{array}{ccccc} & 0 & 1 & \overset{i}{2} & 3 \\ A = [& 3 & 8 & 0 & 7] \\ P = [& 3 & 11 & 11 & 18] \end{array}$$

↑

map
3 → 0
11 → 1
18 → 3

A = [0 3 8]
P = [0 3 11]

TC = O(N)
SC = $\underline{O(N)}$

Q → Given an array containing only 0's & 1's.
Find max length of a subarray which contains equal no. of 0's & 1's.

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

Ans = $\underline{8}$

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

Ans = 8

max length subarray with sum = 0

A = [0 1 0 0 0 1 0 0 1 1 0 0]
[-1 1 -1 -1 -1 1 -1 -1 1 1 -1 -1]

replace 0 with -1.

equal 0's & 1's ⇒ equal -1's & 1's → $\underline{sum = 0}$ ✓

H.W → Count the # subarrays with sum = 0 for given integer A[].

$\boxed{10:25\ PM}$

Q→ Given an integer array A & an integer K,

Hyland

check if there exist a pair (i,j) s.t $\boxed{\begin{array}{l} i \mathrel{!}= j \ \&\& \\ A[i]+A[j] = K \end{array}}$

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

K = 20 A[1] + A[6] = 9 + 11 = 20 Ans = $\underline{true}$

K = 9 $\left.\begin{array}{l} A[0] + A[2] = 8 + 1 = 9 \\ A[4] + A[5] = 4 + 5 = 9 \\ A[3] + A[6] = -2 + 11 = 9 \end{array}\right\}$ Ans = $\underline{true}$

K = 0 ⟶ Ans = $\underline{false}$

$$A = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [8 & -6 & 0 & 1 & 0 & 3] \end{array} \longrightarrow K = \underline{16} \quad i=0, j=0 \ ✗$$

K = 0 A[2] + A[4] = 0 + 0 = 0 Ans = $\underline{true}$ A[0] + A[0] = 16 ✓

$\underline{Bruteforce}$ → ∀ i,j check if A[i] + A[j] = K.

```
for i → 1 to (N-1)
    for j → 0 to (i-1)
        if (A[i] + A[j] == K)
            return true
return false
```

$\boxed{i \mathrel{!}= j}$ ∵ A[i] + A[j] = A[j] + A[i]

∴ only consider

i < j or $\boxed{i > j}$

TC = $\underline{O(N^2)}$ SC = $\underline{O(1)}$

∀i find A[j] s.t A[i] + A[j] = K ⇒ $\boxed{A[j] = K - A[i]}$

∀i, $\underline{\text{check if } K - A[i] \text{ is present in input.}}$ → $\underline{Hashset}$

Solution Steps → 1)

```
for i → 0 to (N-1)
    hs.add(A[i])
```

} Insert all elements in hashset.

2)

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

} check if K-A[i] is present in input ∀i.

$$A = \left[\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 8 & -6 & 0 & 1 & 0 & 3 \\ i & & & & & \end{array}\right]$$

K = 16

(16-8) present in input → Hashset ✓ (i==j)

Ans = true ✗

∀i, check if K-A[i] is present in input from index 0 to (i-1).

```
for i → 1 to (N-1)
    for j → 0 to (i-1)
        if (A[i] + A[j] == K)
            return true
return false
```

∀ index i → only elements from index 0 to i-1 should be part of Hashset.

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

$$A = \left[\begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 9 & 7 & 420 & -2 & 0 & 9 \end{array}\right]$$

K = 7

i	K-A[i]	Hashset
0	7-9=-2	{ }
1	7-7=0	{9}
2	7-420=-413	{9, 7}
3	7-(-2)=9	{9, 7, 420} ✓ return true

TC = O(N)

SC = O(N)

Q → Given an integer array A & an integer K. Calculate the number of distinct elements in every subarray of size K.

$$A = [\overset{0}{2}\ \ \overset{1}{4}\ \ \overset{2}{3}\ \ \overset{3}{8}\ \ \overset{4}{3}\ \ \overset{5}{9}\ \ \overset{6}{4}\ \ \overset{7}{9}\ \ \overset{8}{4}]$$

$K = 4$
$N = 9$ } # subarray = 6

o/p → 4 3 3 4 3 2

Bruteforce → ∀ subarray of size K, count # distinct elements.

#subarrays of size $K = N-K+1$ | $O(K)$ { Add all elements in Hashset,
Ans = Hashset size.

$TC = O((N-K+1)*K)$
$= O(N*K)$ → $K = \frac{N}{2}$ $O((N-K+1)*K) = O(N^2)$

$(N - \frac{N}{2} + 1) * \frac{N}{2} \approx \frac{N}{2} * \frac{N}{2} = \frac{N^2}{4}$

$SC = O(K)$

Fixed size subarray → sliding window + Hashset

$$A = [\overset{0}{2}\ \ \overset{1}{4}\ \ \overset{2}{3}\ \ \overset{3}{8}\ \ \overset{\overset{i}{4}}{3}\ \ \overset{5}{9}\ \ \overset{6}{4}\ \ \overset{7}{9}\ \ \overset{8}{4}]$$ $K = 4$

Remove ↗ i   Add ↗ i-K

o/p → 4 3 3 4 3 2
✓ ✓ ✓

i	Add	Remove	Hashset	Size
0—3			{~~2~~,4,3,8}	4
4	A[4] = 3	A[0]	{~~4~~,3,8}	3
5	A[5] = 9	A[1]	{~~3~~,8, 9}	3
6	A[6] = 4	A[2]=3	{8,9, 4} →	3 ✗

HashSet → Remove an element completely irrespective of how many times it was added. ✓

HashMap
A[i] → freq of A[i]

$$A = [\overset{0}{2}\ \ \overset{1}{4}\ \ \overset{2}{3}\ \ \overset{3}{8}\ \ \overset{4}{3}\ \ \overset{5}{9}\ \ \overset{6}{4}\ \ \overset{7}{9}\ \ \overset{8}{4}]$$ $K = 4$

i	Add	Remove	HashMap → delete	Size
0—3			{2:~~1~~0, 4:1, 3:1, 8:1}	4
4	A[4]	A[0]	{4:~~1~~0, 3:2, 8:1}	3
5	A[5]	A[1]	{3:~~2~~1, 8:1, 9:1}	3

6	A[6]	A[2]	$\{3:1, 8:\cancel{1}^{0}, 9:1, 4:1\}$	4
7	A[7]	A[3]	$\{3:\cancel{1}^{0}, 9:2, 4:1\}$	3
8	A[8]	A[4]	$\{9:2, 4:2\}$	<u>2</u>

Sliding Window + HashMap

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

H.W. → **Pseudocode**
