Pair sum = k

Distinct elements in every window of len = k

bool

Q) Given N array elements, check if there exists pair (i, j) such that A[i] + A[j] == k && (i != j) k is given sum

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

	i	j	A[i]	A[j]	A[i] + A[j] == k
k = 11	4	8	9	7	11 == 11 T
k = 10	5	9	5	5	10 == 10 T
		×			
k = 22	6	6			F

TC: $O(N^2)$

SC: $O(1)$

```
for(i=0; i<N; i++) {
    for(j=i+1; j<N; j++){
        if( A[j] == k - A[i])
            return true
    }
}
return false
```

$A[i] + A[j] = k$

$A[j] = k - A[i]$

Approach 2: Using hashset

	0	1	2	3	4	5	6	7
A[] = {	8	9	-2	4	5	11	-6	7 } → create HS

k = 9

i	A[i]	b = k - A[i]	b in hs
0	8	1	✗
1	9	0	✗
2	-2	11	✓ return true

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

k = 10

Expected ans: False

i	A[i]	b = k - A[i]	b in hs
0	8	2	∝
1	9	1	∝
2	-2	12	∝
3	4	6	∝
4	5	5	✓ should not return true

(i != j) ∝

Key → A[i] value → fr

HM

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

A[] = { 8 9 -2 4 5 11 -6 7 }

8 : 1

9 : 1

-2 : 1

4 : 1

5 : 1

11 : 1

-6 : 1

7 : 1

k = 10

i	A[i]	b = k - A[i]
0	8	2
1	9	1
2	-2	12
3	4	6
4	5	5
5	11	-1

freq(2) > 0 ✗

freq(1) > 0 ✗

freq(12) > 0 ✗

freq(6) > 0 ✗

freq(5) > 1 ✓

why?

Because A[i] and b - A[i] are both same

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

i	A[i]	b = k - A[i]

1) Create hashmap

TC: O(N)

SC: O(N)

2)

```
for (i=0; i<N; i++) {
    if (A[i] != k-A[i]) {
        if (freq(k-A[i]) > 0) {
            return true
        }
    }
    else if (A[i] == k-A[i]) {
        if (freq(k-A[i]) > 1) {
            return true
        }
    }
}
return false
```

i is the first index

↑

A[i] pairs with k-A[i]

A[i] pairs with k-A[i]

Q2) If we want to know the indexes

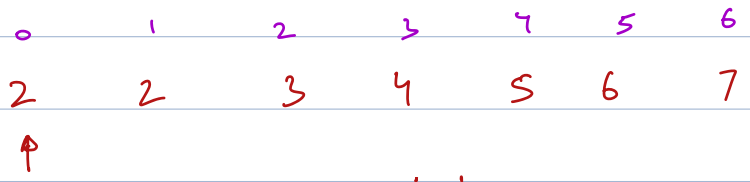
Approach 1):

iterate and find index of k-A[i]

Approach 2): element and its index

<key value>

ie list of indexes



2: [0, 1]

3: [2]

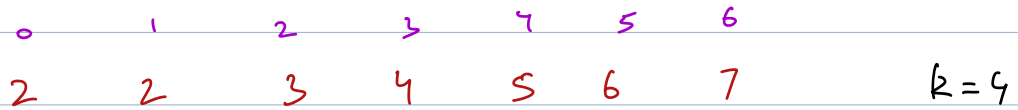
4: [3]

5: [4]

6: [5]

7: [6]

k = 4



k = 4

i	A[i]	k - A[i]	
0	2	2	>1 times occur

Break (10:00 – 10:12)

Q2) Given N elements, calculate no. of distinct elements in every subarray of size k.

N = 10

	0	1	2	3	4	5	6	7	8	9
A = {	2	4	3	8	3	9	4	9	4	10 }

k = 4

Total subarrays: N-k+1

	s	e	unique ele
0	0	3	4
	1	4	3
	2	5	3
	3	6	4
	4	7	3
	5	8	2
N-k	6	9	3

[0 N-k]

N-k+1

TC: $O((N-k+1) \times k)$

$k = \frac{N}{2}$

$(N - \frac{N}{2} + 1) \times \frac{N}{2}$

Brute force

s = 0 e = k-1

```
while( e < N ){
    # b/w s and e
    # I have a subarray
    hs = HashSet()
    for(i=s; i<=e; i++){
        hs.insert(A[i])
    }
    print(hs.size())
    s++, e++
}
```

TC: $\sim O(N^2)$

Sliding window HS will fail

	0	1	2	3	4	5	6	7	8	9
A = {	2	4	3	8	3	9	4	9	4	10 }

s=0 , e=3 HS: {2 4 3 8} len = 4

Index A[0] needs to be removend and A[4] should be added

s=1 e=4 HS: {4 3 8} len = 3

Index A[1] needs to be removend and A[5] should be added

s=2 e=5 HS: {3, 8, 9} len=3

Index A[2] needs to be removend and A[6] should be added

HS: {8, 9, 4} len=3

Freq also

2 3 2 1

	0	1	2	3	4	5	6	7	8	9
A = {	2	4	3	8	3	9	4	9	4	10 }

HM

4:2

9:2

$s = 0$ $e = 3$ $len = 4$

Index A[0] needs to be removend and A[4] should be added

$s = 1$ $e = 4$ $len = 3$

Index A[1] needs to be removend and A[5] should be added

$s = 2$ $e = 5$ $len = 3$

Index A[2] needs to be removend and A[6] should be added

$s = 3$ $e = 6$ $len = 4$

Index A[3] needs to be removend and A[7] should be added

$s = 4$ $e = 7$ $len = 3$

Index A[4] needs to be removend and A[8] should be added

len = 2

```
hm = new HashMap()
for(i=0; i<k; i++){
    if (hm.contains(A[i])){
        hm[A[i]]++    // hm.update(A[i], hm.get(A[i])+1)
    }else{
        hm[A[i]] = 1
    }
}
print(hm.size())
```

iteration : k

s = 1 e = k

[k, N-1]

= N-k

while (e < N) {

inden to remove = s-1

hm[A[s-1]]--

```
hm.update(A[s-1], hm.get(A[s-1])-1)
if (hm.get(A[s-1]) == 0) {
    hm.delete(A[s-1])
}
```

Removal part

inden getting added = e

```
if (hm.contain(A[e]) {
    hm[A[e]]++
}else{
    hm[A[e]] = 1
}
```

Adding part

Iterations : N-k

print(len(hm))

```
|
}
```

s++ , e++

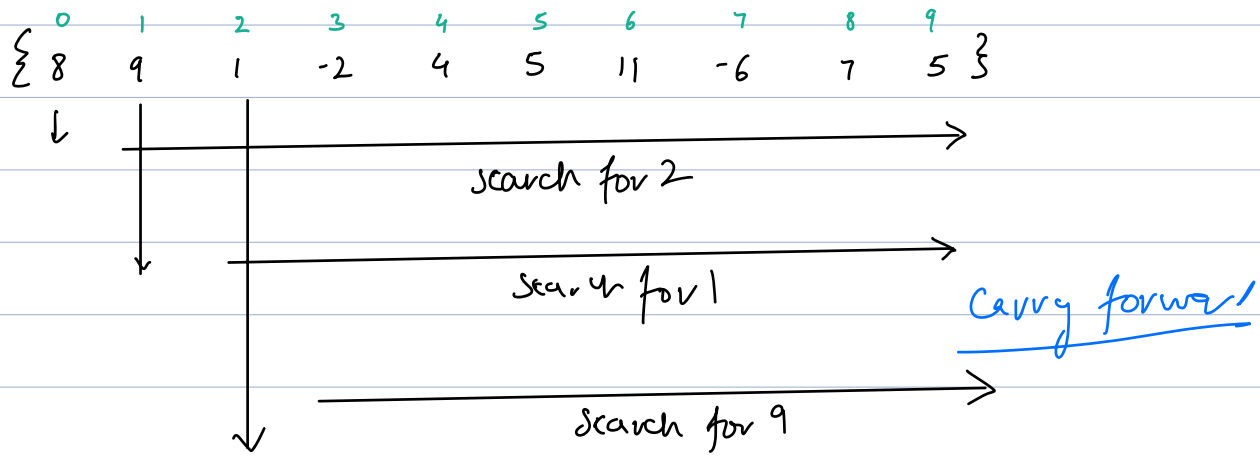
TC: O(N)

N-k+1

SC: O(k)
worst case
O(N)

Done!

k = 10



a q