HashMap | HashSet

↳ Quick access | search | insert

O(1) Avg case

- → insert(K,V) / insert(K)
- → get(K
- → contains(K)
- → delete(K)
- → size()

Q. Two Sum

Amazon Google FAS...

Given an Array of size N, check if there exists a pair (i,j) such that i != j

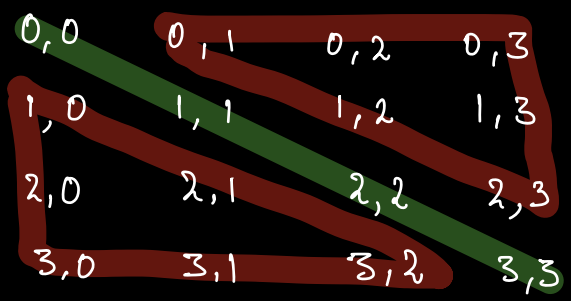
a[i] + a[j] = K

Brute force

Iterate over all possible pairs & check if their sum == K.

```
for(i=0; i<N; i++){
    for(j=0; j<N; j++){
        if(a[i]+a[j]==K && i!=j)
            return true;
    }
}
return false;
```

N = 4




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

TC : $O(N^2)$

SC : $O(1)$

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

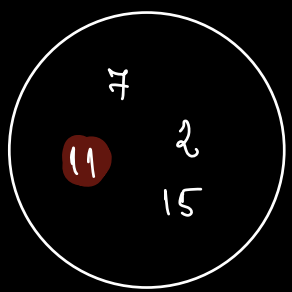
$A[j] = K - A[i]$ → finding

for every i index, search for $K - A[i]$ in the Array, if present return true.

⇒ Set.

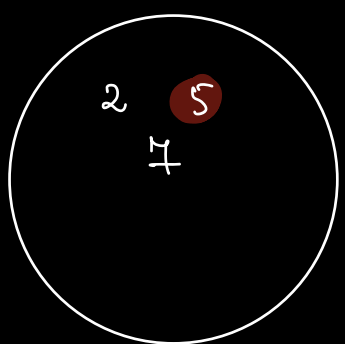
* Insert all Array elements in the set.
* Check for K - A[i] in Set.

A: 2, 7, 11, 15 K = 18



18 - 2 = 16 ? ✗
18 - 7 = 11 ? ✓
↳ True

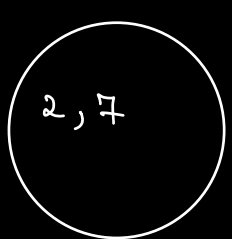
A: [2, 5, 7] K = 10
↳ false



10 - 2 = 8 ? ✗
10 - 5 = 5 ? ✓
→ True

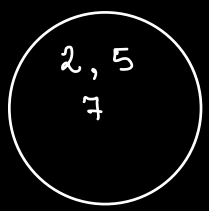
Set

A: 2, 7, 11, 15 K = 18



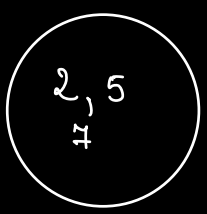
18 - 2 = 16 ?
18 - 7 = 11 ✗
18 - 11 = 7 ? ✓

A: [2, 5, 7] K = 10



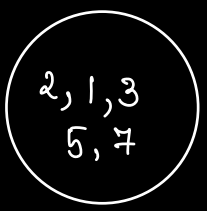
→ false

A: [2, 5, 7, 5] K = 10



10 - 2 = 8 ? ×
10 - 5 = 5 ? ×
10 - 7 = 3 ? ×
10 - 5 = 5 ? ✓ → true

2, 1, 3, 5, 2, 7 K = 11



Hashset

11 - 2 = 9 ? ×
11 - 1 = 10 ? ×
11 - 3 = 8 ? ×
11 - 5 = 6 ? ×
11 - 2 = 9 ? ×
11 - 7 = 4 ? ×

→ false

For every index i :

Check if K - A[i] is present in the SET,

if yes ⇒ Return true

else add a[i] in set.

TC: O(N)

SC: O(N)

↳ HashSet.

Q. Count the no. of pairs with sum = K.

Q. Count the no. pairs s.t i != j &

A[i] - A[j] = K.

Q: Given an Array of size N & no. K, Calculate the no. of distinct elements in every window of size = K.

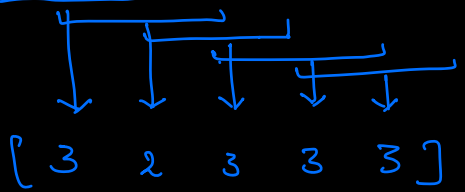
Amazon, FB, Google MNS GS...

A: 1, 1, 2, 2 K = 2

[1, 2, 1]

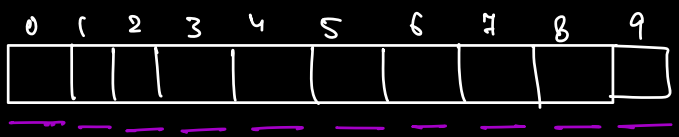
⇒ A: 6, 3, 7, 3, 8, 6, 9 ⇒ 5

A: 6, 3, 7, 3, 8, 6, 9. K=3

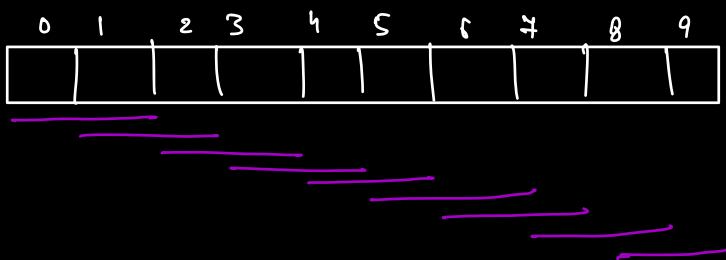


[3 2 3 3 3]

N = 10, K = 1 ⇒ 10



N = 10, K = 2



9

N, K

Start point of 1st window = 0.

$[x, N-1] \Rightarrow K$

↳ $N-1-x+1 = K$

$N-x = K$

$x = N-K$

Start point of last window = N-K

$S \in [0, N-K]$

No. of windows of size = K

↳ $N-K-0+1 \Rightarrow N-K+1$

* Brute force

for every window/subarray of size K, count the # of distinct elements.

* for every window of size K,

i) insert all K elements to empty set

ii) add set.size() to ans. array.

# of windows = $N-K+1$

# of iterations = $(N-K+1)K$

$K=1 \Rightarrow N$

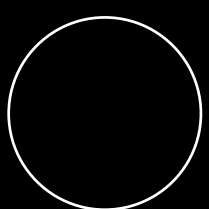
$K=N \Rightarrow N$

$K=\frac{N}{2} \Rightarrow \left(N-\frac{N}{2}+1\right).\frac{N}{2} \approx O(N^2)$

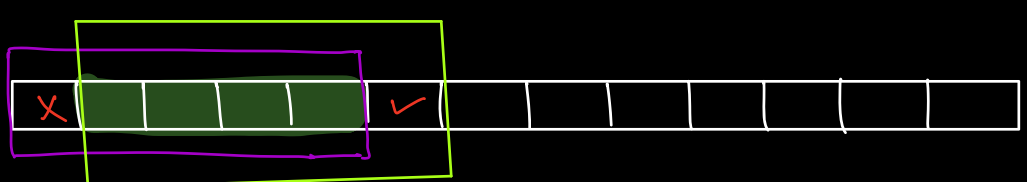
TC: $O(N^2)$

SC: $O(N)$

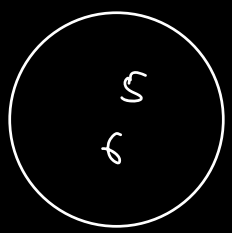
Ex A: 5, 3, 2, 5, 1, 5, 2 $\quad k=3$



ans: [3, 3, 3, 2, 3]



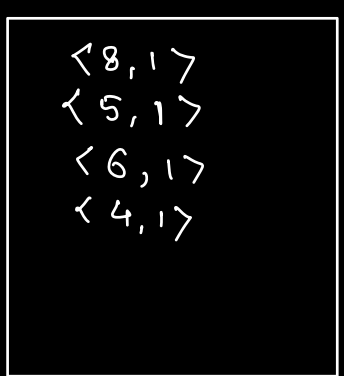
A : 3, 3, 5, 6, 3, 5, 4, 6, 8, 3 (indices 0 1 2 3 4 5 6 7 8 9) k = 4



[3, 3, 2 ✗

⇒ Map<int, int>
A[i] freq.

A : 3, 3, 5, 6, 3, 5, 4, 6, 8, 3 (indices 0 1 2 3 4 5 6 7 8 9) k = 4
x x ✓



Map

ans: [3, 3, 3, 4, 4, 4,

A : [3 3 5 6 5 4 6 8 3] k = 4

5,2
6,1
4,1

[3, 3, 3

1) Build a freq map for 1st window. ⇒ O(K)

2) Iterate over all remaining windows : → N-K

- Remove the first element of previous window
  - ↳ decrement the freq
  - → if (freq == 0) remove ele from map
- Add the new element.

TC: O(N)

SC: O(N)

Q. Given an Array of size N, find the length of largest sequence that can be rearranged to a sequence of consecutive numbers.

Google
Amazon
MS | Flipkart
- - -

A: 100, ④, 200, ①, ③, ②.

4, 1, 3, 2 ⇒ 1, 2, 3, 4

⇒ 4

Quiz

A: -1, 8, 5, 2, 3, 7, 1, 4, 9

[5, 2, 3, 1, 4] → 1, 2, 3, 4, 5

⇒ 5

Sorting

A: -1, 8, 5, 2, 3, 7, 4, 1, 4, 9

↓ sort

-1, 1, 2, 3, 4, 4, 5, 7, 8, 9

d=1 d=1 1 1 1 5 d=1 1 3

TC: O(N log N)

SC: Depends on sort algo.

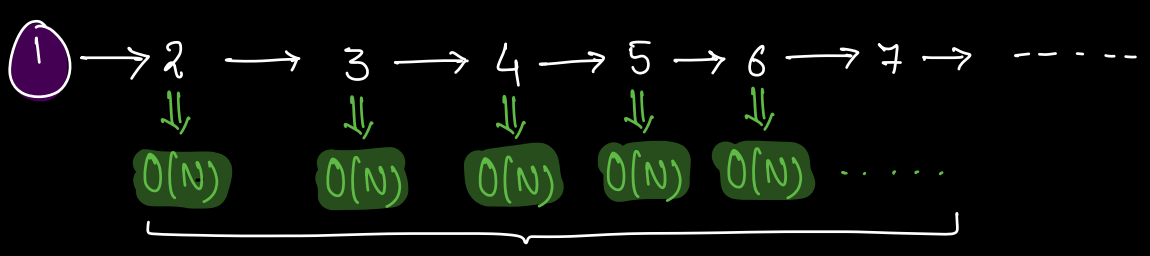
Brute Force

Pick every element & try to see the length of longest consecutive seq. we can form with this starting element.

A: -1, 8, 5, 2, 3, 7, 1, 4, 9

	length
Consecutive seq. starting at -1 ⇒ -1, ~~8~~	1
Consecutive seq. starting at 8 ⇒ 8, 9, ~~2~~	2
Consecutive seq. starting at 5 ⇒ 5, ~~3~~,	1
Consecutive seq. starting at 2 ⇒ 2, 3, 4, 5, ~~8~~	4
⋮	

* A: [1, 4, 2, 3, 7, 6, 5, 8, 9]



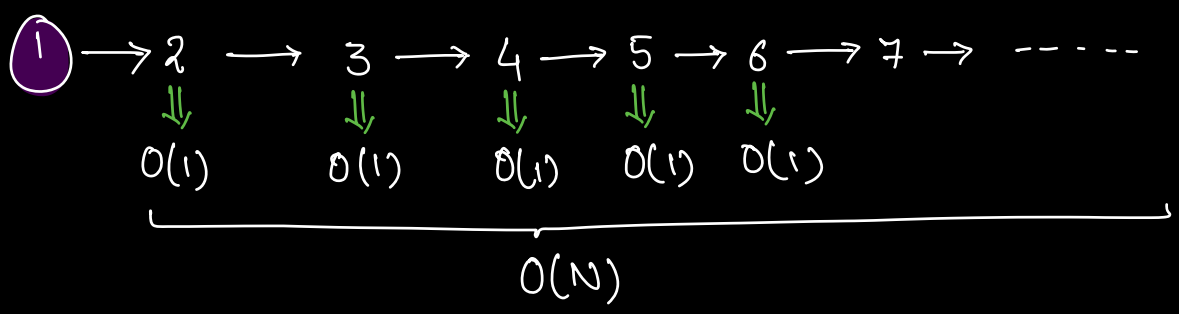
$O(N^2)$

Overall TC: $O(N \cdot N^2)$

$O(N^3)$

⇒ Instead of searching linearly, we can use SET.

```
for(i= 0; i< N; i++) {
    Set.insert(A[i])
}
// Count the length of seq. starting at A[i]
for(i= 0; i< N; i++) {
    l=0 , n= A[i]
    while(Set.contains(n)) {
        l++
        n++
    }
    ans = max(ans, l);
}
```



TC: $O(N \cdot N)$

SC: $O(N)$

1 ⇒ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ✱

2 ⇒ 2, 3, 4, 5, 6, 7, 8, 9, 10, ✱

3 ⇒ 3, 4, 5, 6, 7, 8, 9, 10, ✱

4 ⇒ 4, 5, 6, 7, 8, 9, 10, ✱

⋮ ⋮

⇒ for any A[i]:

if A[i]-1 is present in the set then there's NO need to check the seq. for A[i].

```
for(i= 0; i< N; i++) {
    set.insert(A[i])
}
// Count the length of seq. starting at A[i]
for(i= 0; i< N; i++) {
    if(!set.contains(A[i]-1)) {
        l=0, n= A[i]
        while(set.contains(n)) {
            l++
            n++
        }
        ans = max(ans, l);
    }
}
```

A : [1 , 4 , 2 , 3 , 7 , 6 , 5 , 8 , 9]

1 → 1, 2, 3, - - - - 9 J = 4

4 → X

2 → X

3 → X

7 → X

6 → X

5 → X

8 → X

9 → X

TC : O(N)

SC : O(N)

A : -1 , 8 , 5 , 2 , 3 , 7 , 1 , 4 , 9

-1 → -1, ~~∅~~

8 → X

5 → X

2 → X

3 → X

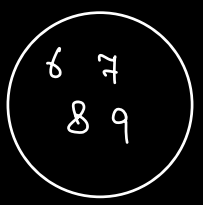
7 → ~~7~~, 8, 9, ~~6~~

1 → 1, 2, 3, 4, 5, 6

4 → X

9 → /

A: [6, 6, 7, 6, 6, 6, 6, 8, 6, 9]



6 → 6, 7, 8, 9
6 → 6, 7, 8, 9
6 → 6, 7, 8, 9
6 → 6, 7, 8, 9
6 → 6, 7, 8, 9
6 → 6, 7, 8, 9
6 → 6, 7, 8, 9

⇒ Instead of iterating over Array, iterate over set.

↳ WC TC ⇒ O(N)

— * —