

"If I have seen further than others, it is by standing upon the shoulders of giants."
— Isaac Newton

⇒ For mock interview practice: pramp.com

Q. Given N elements, check if there is a pair (i, j) , such that $A[i] + A[j] = K$ and $i \neq j$

ex. $A = [8, 9, 1, -2, 4, 5, 11, -6, 7, 5]$

$K = 11$, ans = true (4, 8)

$K = 6$, ans = true (0, 3), (2, 5) (2, 9)

$K = 22$, ans = false (~~6, 6~~) ?

⊛ Brute force:
① for every num in the array
② check if $K - \text{num}$ exists.

bool twoSum (A, K) {

```

int N = A.size();
for (i=0; i<N; i++) {
    a = A[i];
    for (j=i+1; j<N; j++) {
        b = A[j];
        if (a+b == k) {
            return true;
        }
    }
}
return false;

```

T.C. = $O(N^2)$
S.C. = $O(1)$

2nd approach: ① could we use hashset?
(wouldn't work)

idea: ① insert every element in hashset
② for every num, check if K-num is in the set $O(1)$ query in the set

A = ⁰8 ¹9 ²1 ³-2 ⁴4 ⁵5 ⁶11 ⁷-6 ⁸7 ⁹5

Ⓘ $K = 11$, $ans = \underline{true}$ (4,8)
 $\underline{a+b} == K$ (11)

a	b	is b Present
8	3	X
9	2	X
1	10	X
-2	13	X
4	7	✓

return true.

Ⓜ $K = 6$, $ans = \underline{true}$
 $a+b = 6$

a	b	is b Present
8	-2	✓

return true

Ⓝ $K = 22$, $ans = \text{false}$
 $a+b = 22$

a	b	is b Present
---	---	--------------

8	14
9	13
1	21
-2	24
4	18
5	17
11	11
-6	

X
X
X
X
X
X

(11, -6, 7
-2, 4, 5
8, 9, 1
hs.)

✓ return true

O(1)
hs.contains(num)?
✓

IV

K = 10, ans = true

Q
8
...

b

is present

5

5

✓ return true

3rd approach : using hashmap. (key, value)
complicated

① populate hashmap

② check for every A[i]

if (a == b) check

index != hm[a]

A[i] ↑
index
a

①

$A = [8^0, 9^1, 1^2, -2^3, 4^4, \underline{5^5}, (11^6), -6^7, 7^8, \underline{5^9}]$

$hm = ((8,0), (9,1), (1,2), (-2,3), (4,4), (5,8), (11,6), (-6,7), (7,8), \cancel{(5,9)})$
 ✓, ✗.

$K = 10$, $ans = \underline{true}$

(II)

a	b	is b present
8	.	✗
⋮		
5	5	✓

return true

(II) $K = 22$, $ans = \underline{false}$

a	b	is b present
8	14	✗
9	13	✗
1	21	✗
-2	24	✗
4	18	✗

5	17
11	11
-6	28
7	15
5	17

\times
 $(a == b) \Rightarrow \times \text{ hm}[11] \neq 6$
 \times
 \times
 \times
 return false

4th approach: ✓

- ① use hash map (key, value)
- \uparrow \uparrow
 $A[i]$ freq.
- ② for every a check if $b = k - a$ is present
 if $(a == b)$? $\text{freq.} > 1$ return true;

$\text{hm} = ((8, 1), (9, 1), (1, 1), (-2, 1), (4, 1), (5, 2), (11, 1), (-6, 1), (7, 1))$

I $K = 11$, $\text{ans} = \text{true}$ (4, 8)
 $a + b == K$ (11)

a	b	is b Present
8	3	\times

9
1
-2
4

2
10
13
7

X
X
X
✓

return true,

II

K = 10, ans = true

a + b = 10

a
8
⋮

b
3
⋮

is b present
X
⋮

5

5

(a == b)?, freq > 1, return true

III

K = 22, ans = false

a + b = 22

a
8
⋮
11
⋮

b
3
⋮
11
⋮

is b present
X
⋮
(a == b)?
X
X

X
freq > 1 X

return false

```
bool twoSum (A, k)
{
    int N = A.size();
    hashmap <int, int> hm; // (A[i], freq)
    // populate hm. // O(N), O(N)
                        TC: SC
    for (i=0; i<N; i++) { // O(N) Time
        a = A[i];
        b = k - a;
        if (a != b and hm.contains(b) == true) {
            return true;
        }
        if (a == b and hm[a] > 1)
            return true;
    }
    return false;
}
```

$O(2N)$

$T.C. = O(N)$
 $S.C. = O(N)$

return true,

} H.W. return (i, j)

5th approach: Hashset $O(N)$

A = [⁰8 ¹9 ²1 ³-2 ⁴4 ⁵5 ⁶11 ⁷-6 ⁸7 ⁹5]

idea: ① populate hashset while finding the pair
② for every num check hashset

k = 11

a + b = 11

a	b	HS.	is b present in HS.
8	3	(<u> </u>) ↳ (8)	×
9	2	(<u>8</u>) ↳ (9)	×
1	10	(8, 9) ↳ (1)	×

-2	13	(8, 9, 1) ↳ (-2)	X
4	7	(8, 9, 1, -2) ↳ (4)	X
5	5	(8, 9, 1, -2, 4) ↳ (5)	X
11	0	(8, 9, 1, -2, 4, 5) ↳ (11)	X
-6	17	(8, 9, 1, -2, 4, 5, 11) ↳ (-6)	X
7	4	(8, 9, 1, -2, 4, 5, 11, -6)	✓

return true

K = 10

A = [8⁰ 9¹ 1² -2³ 4⁴ 5⁵ 11⁶ -6⁷ 7⁸ 5⁹]

Q	b	HS	is b present
8	2	{ ↳ 8	X

9	1	(8)	x
		↘ 9	
1	9	(8, <u>9</u>)	✓

A = [8, 5, 5]

K = 10

a	b	HS	is b present
---	---	----	--------------

8	2	(-)	x
		↘ 8	

5	5	(8)	x
		↘ 5	

5	5	(8, <u>5</u>)	↘ return true
---	---	----------------	---------------

```

bool twoSum (A, K) {
    int N = A.size();

    hashset<int> hS;

```

```

    for (i=0; i<N; i++) {
        a = A[i];
        b = K-a;
        if (hs.contains(b)) {
            return true;
        }
        hs.insert(a);
    }
    return false;
}

```

T.C. = $O(N)$
S.C. = $O(N)$

10:39

Q. Given N elements, calculate no. of distinct elements in every subarray of size K .

ex. $A = [2 \ 4 \ 3 \ 8 \ 3 \ 9 \ 4 \ 9 \ 4 \ 10]$

0 1 2 3 4 5 6 7 8 9

$K = 4$.

Subarrays : print

[0-3]	:	4
[1-4]	:	3
[2-5]	:	3
[3-6]	:	4
[4-7]	:	3
[5-8]	:	2
[6-9]	:	3

array size

Quiz:

$A = [3, 2, 1, 5, 3, 6, 2, 6, 2, 1]$, $K=5$, $N \swarrow$

subarrays.

[0-4]

[1-5]

[2-6]

[3-7]

[4-8]

[5-9]

4
5
5
4
3
3

1st idea

① for all the subarrays.

$\Rightarrow \underline{(N-K+1)}$

② find unique elements (using set)

T.C. = $(N-K+1) * K$

$$\begin{aligned}
 & \text{K} = \text{N}/2 \quad \text{K} = \text{L} \quad \text{T.C.} = O(N) \\
 & \quad \quad \quad \text{K} = \text{N} \quad \text{T.C.} = O(N) \\
 & \rightarrow \left(\frac{\text{N}}{2} + 1\right)^* \left(\frac{\text{N}}{2}\right) \Rightarrow \underline{O(N^2)}
 \end{aligned}$$

$$\text{S.C.} = \underline{O(K)}$$

- 2nd idea:
- ① sliding window over hashset
 - ② delete prev. element and add next element

ex. $A = [2, 4, 3, 8, 3, 9, 4, 9, 4, 10], K = 4$

0 1 2 3 4 5 6 7 8 9

(0-3) \Rightarrow fill the hash set (2, 4, 3, 8) size 4 ✓

[1-4] delete Add (4, 3, 8) 3

A[0] A[4]

[2-5] A[1] A[5] (3, 8, 9) 3

✓ ✓

[3-6] A[2] A[6] (8,9,4) ~~3~~ 4

obsⁿ: in hashset, deleting, delete all the occurrences

3rd idea: ① use hashmap with frequency.

ex. A = [2 4 3 8 3 9 4 9 4 10], K=4
 0 1 2 3 4 5 6 7 8 9

[0-3] Hashmap [(2,1), (4,1), (3,1), (8,1)] size
 4 OK

[1-4] delete A[0] Add A[4] [~~(2,1)~~⁰, (4,1), (~~3,1~~)², (8,1)] 3 OK

[2-5] A[1] A[5] [(9,1), ~~(4,1)~~⁰, (3,2), (8,1)] 3 OK

[3-6] A[2] A[6] [(9,1), (4,1), (~~3,2~~)¹, (8,1)] 4 OK

[4-7] A[3] A[7] [~~(9,1)~~², (4,1), (3,1), ~~(8,1)~~] 3
 2 - -

[5-8] A[4] A[8] [(9,2), (4,~~1~~), (~~3~~,1)] 2
 (6-9) A[5] A[9] [(~~9~~,~~2~~), (4,2), (10,1)] 3

obsⁿ: ① max. k elements would be present in hashmap.

$$\begin{aligned}
 T.C &\Rightarrow O(N - K + 1 + K) = \underline{O(N)} \\
 S.C &= \underline{O(K)} \Rightarrow \underline{O(2K)}
 \end{aligned}$$

idea:

- ① for first subarray populate hm, ^(num, freq)
- ② then sliding window, add, delete elements
- ③ print hashmap size

void

distinctElement(A, K) {

int N = A.size()

hashmap<int, int> hm; // (num, freq).

// populate the hm for first subarray

for (int i = 0; i < K; i++) {


```
if (hm.containsKey(A[i])) {  
    hm[A[i]]++;  
}  
hm[A[i]] = 1;
```

```
}  
print (hm.size());
```

// sliding window

s = 1, e = k

```
while (e < N) {  
    // we have a subarray from [s-e]  
    // remove A[s-1] and add A[e]  
  
    hm[A[s-1]]--;  
    if (hm[A[s-1]] == 0) {  
        hm.remove(A[s-1]);  
    }  
}
```

```
if (hm.contains(A[e]) == true) {  
    hm[A[e]]++;  
}  
else {  
    hm[A[e]] = 1;  
}  
print(hm.size());  
// update the window  
s++;  
e++;
```

}

}