Lecture :- Hashing - 2

Agenda

- Pair sum = K
- Count no. of pairs with sum = K
- Subarray with sum = K
- Distinct elements in every window.

Qu.1 Given arr[n] and k. check if there exists a pair (i,j) such that arr[i] + arr[j] = k and i != j

8	9	1	-2	4	5	11	-6	4

k = 6 true [1+5]

k = 22 false

k = 8 true. [4+4]

3	5	1	2	1	2

k = 7 true [5+2]

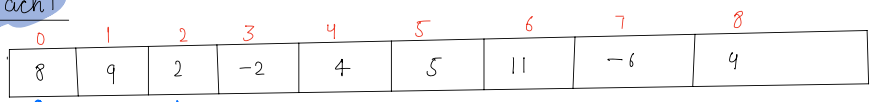
k = 10 false.

Brute force.

TC: $O(n^2)$

SC: $O(1)$

Approach!



k = 9

true.

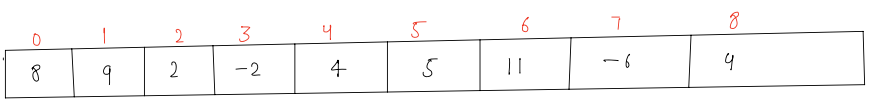
target 1 | 9-9 = 0 | 9-2 = 7 | 11

$-2 + x = 9$

$x = 11$ ✓

hashset

8	9	2	-2
~~4~~	5	11	-6
4			



k = 4

true.

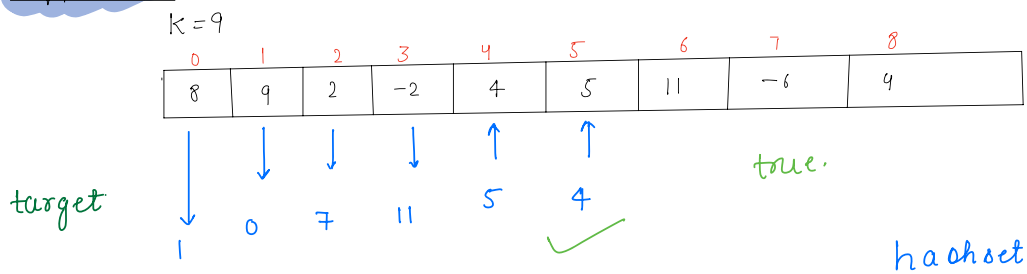
[wrong]

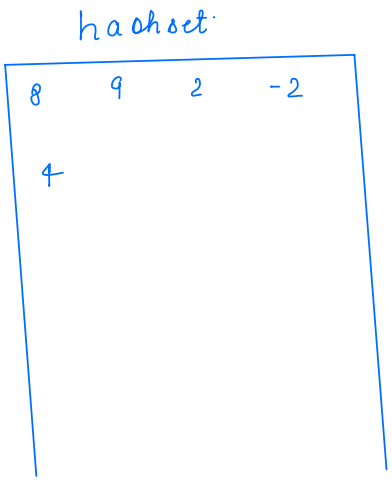
target -4 | -5 | 4-2 = 2 ✓

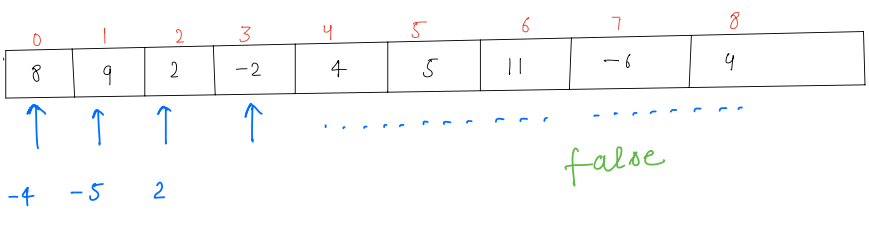
hashset

8	9	2	-2
~~4~~	5	11	-6
4			

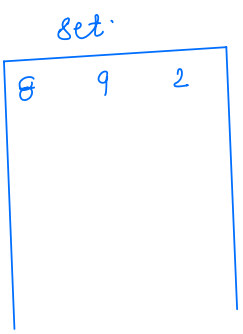
Approach2







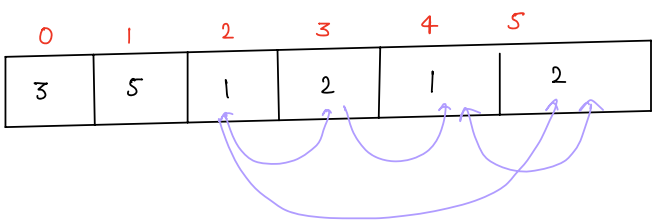
target -4 -5 2



Code

```
boolean targetSum(int[] arr, int k) {
    int n = arr.length;
    Set<Integer> set;
    for(i=0; i<n; i++) {
        int target = k - arr[i];
        if( set.contains(target)) {
            return true;
        }
        set.add(arr[i]);
    }
    return false;
}
```

TC: O(n)
SC: O(n)

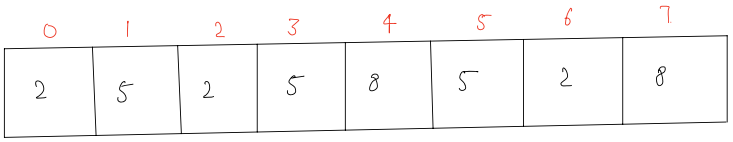


k = 3 ans = 4

Qu Count no. of pairs with sum = K.

Given arr[n], count no. of pairs such that

arr[i] + arr[j] = k and i != j

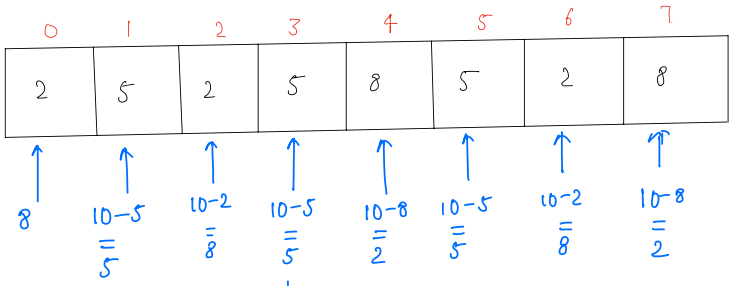


, K=10

pairs :- [0, 4]
[0, 7]
[1, 5]
[1, 3]
[2, 4]
[2, 7]
[3, 5]
[4, 6]
[6, 7]

ans = 9.

Approach



target:

8	10-5 = 5	10-2 = 8	10-5 = 5	10-8 = 2	10-5 = 5	10-2 = 8	10-8 = 2

count:

		+1	+2	+2	+1	+3

map

2 — ~~1~~ ~~2~~ 3

5 — ~~1~~ ~~2~~ 3

8 — ~~1~~ 2

ans: 9

Code

```
int countTargetSum(int[] arr, int k) {
    n = arr.length;
    Map<Integer, Integer> map;
        array elements    frequency.
    int cnt = 0;
    for(i=0; i<n; i++) {
        int target = k-arr[i];
        if(map.containsKey(target)) {
            cnt = cnt + map.get(target);
        }
        if(map.containsKey(arr[i])){
            int freq = map.get(arr[i]);
            freq += 1;
            map.put(arr[i], freq);
        }
        else{
            map.put(arr[i], 1);
        }
    }
    return cnt;
}
```

TC: O(n)
SC: O(n)

Qu3 Given arr[n], Check if there exists a subarray with sum = k

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

k = 11 { 2, 3, 9, -4, 1 } { 5, 6 } true

k = 10 { 2, 3, 9, -4 } true.

k = 15 { -4, 1, 5, 6, 2, 5 } true

0	1	2	3	4
5	10	20	100	105

k = 110 false.

Brute force Go to all subarrays — $O(n^2)$

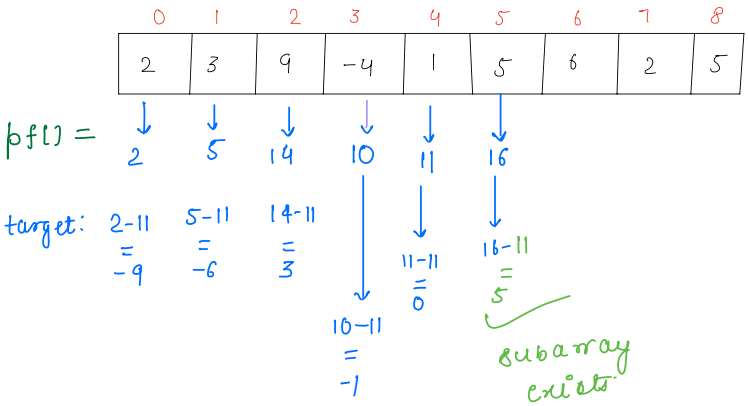
find sum — $O(n)$

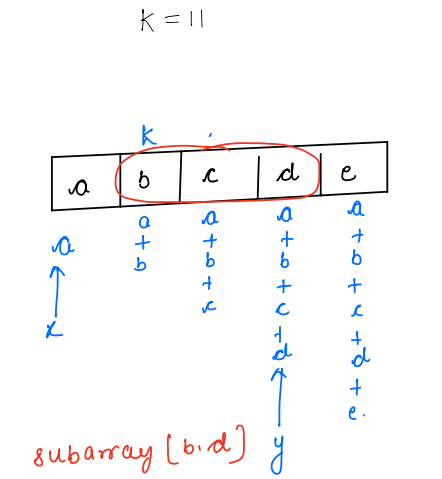
compare.

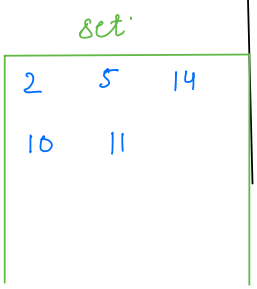
TC: $O(n^3)$

SC: $O(1)$

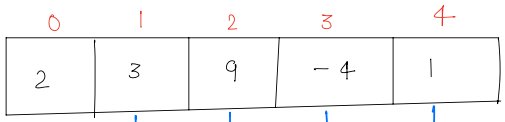
Approach


0 1 2 3 4 5 6 7 8
2 3 9 -4 1 5 6 2 5
pf[] =
2 5 14 10 11 16
target: 2-11 = -9
5-11 = -6
14-11 = 3
10-11 = -1
11-11 = 0
16-11 = 5
subarray exists.


k = 11
k
a b c d e
a
x
a + b
a + b + c
a + b + c + d
y
a + b + c + d + e.
subarray [b, d]
y - x = k.
y - k = x
search in set.

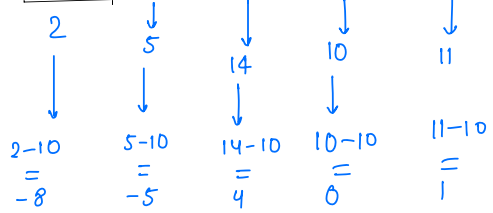

set:
2 5 14
10 11

Edge case



k = 10

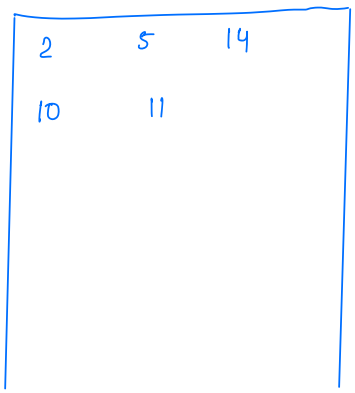
pf[]



false. (true),

target

set



Resolve:- Add 0 at the start in set

Code

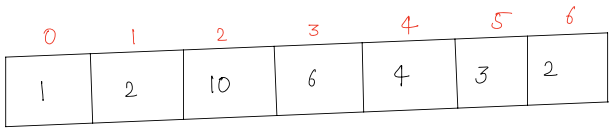
```
boolean targetSubArray (arr, k) {
    n = arr.length;
    Set<Integer> set;
    set.add(0);
    int sum = 0;
    for(int i=0; i<n; i++) {
        sum += arr[i];
        int target = sum-k;
        if(set.contains(sum-k)) {
            return true;
        }
        set.add(sum);
    }
    return false;
}
```

TC: O(n)
SC: O(n)

Break: 8:45 – 8:55

Sliding window – Map

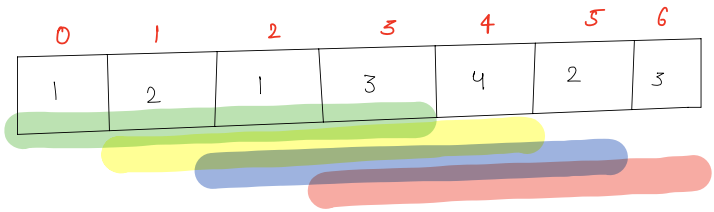
Sliding window suggests to insert all elements of first window in the set. Now slide the window, throw the previous element out and insert new adjacent element in the set.



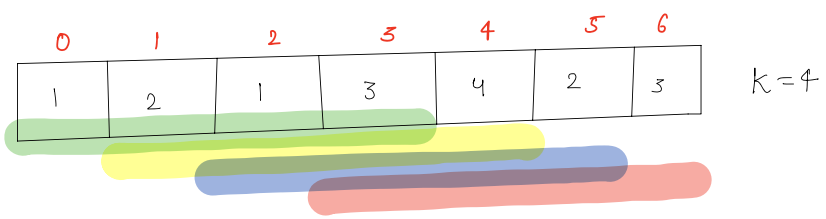
size of window = 4

Window	Indices	All elements of window

Qu Given arr[n] and number K. find count of distinct elements in every window of size K in array.



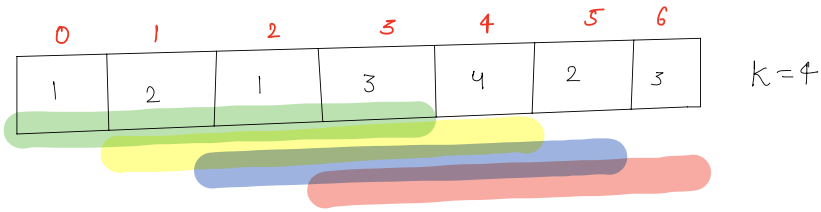
Window	Indices	All elements of window	distinct
first	[0-3]	1, 2, 1, 3	3
2nd	[1-4]	2, 1, 3, 4	4
3rd	[2-5]	1, 3, 4, 2	4
4th	[3,6]	3, 4, 2, 3	3



1st window: < 1, 2, ~~1~~, 3> ans = 3

2nd window: < 2, 3, 4> ans = 3 ✗ wrong.

Approach



Window	Indices	Map	Distinct elements
1st	[0-3] {1, 2, 1, 3}	1 : 2 2 : 1 3 : 1	3 [map.size()]
2nd	[1-4] {2, 1, 3, 4}	1 : ~~2~~ 1 2 : 1 3 : 1 4 : 1	4 map.size()]
3rd	{2-5} {1, 3, 4, 2}	1 : 1 ~~2 : 1~~ 3 : 1 4 : 1 2 : 1	4 [map.size()]
4th	[3, 6] [3, 4, 2, 3] remove = 1 add = 3	~~1 : 1 0~~ ~~2 : 1~~ 3 : ~~1~~ 2 4 : 1 2 : 1	3 [map.size()]

Code: h/w

Thankyou

8th Dec: