Q.1) Given arr[n] and an integer k, check if there exists a pair (i, j) such that arr[i] + arr[j] = k and $i \neq j$.

eg :- arr → { 8 (0) 9 (1) 1 (2) -2 (3) 4 (4) 5 (5) 11 (6) -6 (7) 4 (8) }

k = 6 ⇒ i = 2, j = 5 True

k = 22 ⇒ does not exist False

k = 8 ⇒ i = 4, j = 8 ⇒ arr[4] + arr[8] = 4 + 4 = 8 = k True.

arr → { 3, 5, 1, 2, 1, 2 } k = 7

True

arr → { 3, 5, 1, 2, 1, 2 } k = 10.
(0) (1) (2) (3) (4) (5)

arr[1] + arr[1] = 10

✗ i = j ✗

Brute Force

For each i, explore $j \in [i+1, n-1]$

i = 0 ⇒ $j \in [1, n-1]$

i = 1 ⇒ $j \in [2, n-1]$

i = 2 ⇒ $j \in [3, n-1]$

⋮

i = n-2 ⇒ $j \in [n-1, n-1]$

```
boolean checkPair(arr[], k){
    n = len(arr)
    for(i → 0 to n-1){
        for(j → i+1 to n-1){
            if(arr[i] + arr[j] == k)
                return true
        }
    }
    return false
}
```

$O(n^2)$ T.C
$O(1)$ S.C

→ sort and two pointers approach

$O(n\log n + n) = O(n\log n)$ T.C

Hashing approach

arr → { 8 (0) 9 (1) 1 (2) -2 (3) 4 (4) 5 (5) 11 (6) -6 (7) 4 (8) }

k = 6

8 → we need a (-2) → HashSet

arr[i]

Wrong approach

→ Put all elements in the HashSet
→ Iterate through the array, for each i → check if (k − arr[i]) is present in HashSet or not.

k = 9

arr → { 8 (0) 9 (1) 2 (2) -2 (3) 4 (4) 5 (5) 11 (6) -6 (7) 4 (8) }

1 ✗ 0 ✗ 7 ✗ 11 ✓ True

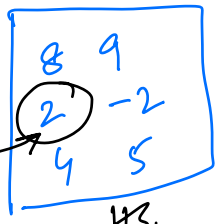
HS: { 8, 9, 2, -2, 4, 5, 11, -6, 4 }

Edge case

arr = {8, 9, 2, -2, 4, 5}

k = 4

k - 2 = 4 - 2 = 2



True ✓

HashSet with gradual insertion

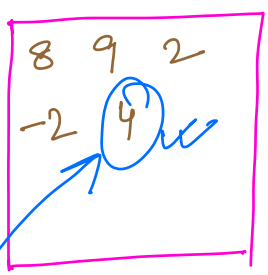
arr = {8(0), 9(1), 2(2), -2(3), 4(4), 5(5)}

k = 9

i = 0	1	2	3	4	5
Target	1 ✗	0 ✗	7 ✗	11 ✓	5 ✗	9 - arr[5] = 9 - 5 = 4

9 - arr[0] = 9 - 8 = 1

9 - (-2) = 9 + 2 = 11



HS → stores all elements before your current elem (i)

True.

```
boolean targetSum(arr[], k){
    n = len(arr)
    HashSet<int> hs
    for(i -> 0 to n-1){
        if(hs.contains(k - arr[i]))
            return true
        hs.add(arr[i])
    }
    return false
}
```

O(n) T.C.
O(n) S.C.

Q2) Given arr[n] and an integer k, count the no. of pairs (i,j) such that arr[i] + arr[j] = k, $i \neq j$, and $(i,j) \equiv (j,i)$.

$\{^{0}3, ^{1}5, ^{2}1, ^{3}2, ^{4}1, ^{5}2\}$

$k = 3$

$i=2, j=3$
$i=2, j=5$
$i=3, j=4$
$i=4, j=5$

} 4 pairs.

$\rightarrow$ $i=3, j=2$ don't count twice.

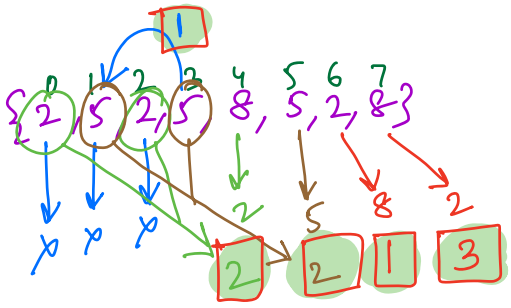
$\{^{0}2, ^{1}5, ^{2}2, ^{3}5, ^{4}8, ^{5}5, ^{6}2, ^{7}8\}$

$k = 10$.

Pairs (i,j)

(1,3)
(0,4)
(1,5)
(0,7)
(2,4)
(4,6)
(6,7)
(3,5)
(2,7)

9 pairs.

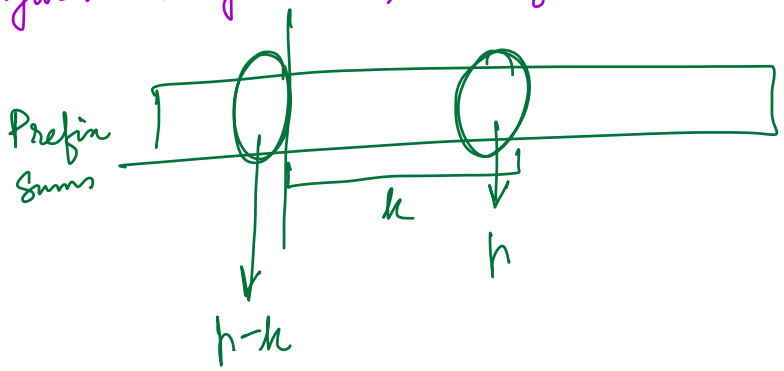


1+2+2+1+3 = 9.

```
int countTargetSum(arr[], k){
    n = len(arr)
    HashMap<int, int> hm
    int c = 0
    for(i -> 0 to n-1){
        if(hm.containsKey(k - arr[i])){
            c += hm.get(k-arr[i])
        }
        if(hm.containsKey(arr[i])){
            hm.put(arr[i], hm.get(arr[i])+1)
        }
        else{
            hm.put(arr[i], 1)
        }
    }
    return c
}
```

O(n) Te
O(n) Sc.

Q3) Given an array arr[n], check if there exists a subarray with sum = k.



sum(arr[i] ... arr[j]) = k

⇒ PfSum[j] − PfSum[i−1] = k.

⇒ PfSum[i−1] = (PfSum[j] − k) → search before j.

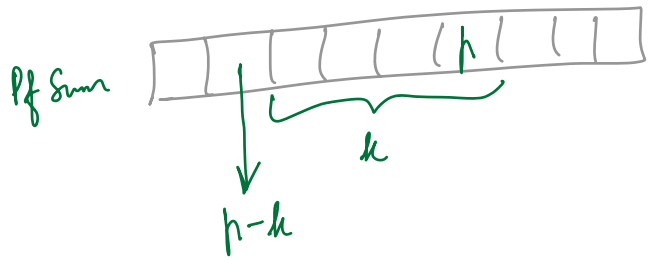
	0	1	2	3	4	5	6	7	8
	2	3	9	−4	1	5	6	2	5
PfSum →	2	5	14	10	11	16	22	24	29

PfSum[5] − PfSum[1] = 11

⇒ sum(arr[2] ... arr[5]) = 11.

k = 11 ⇒ {5,6} {2,3,9,−4,1} {9,−4,1,5}

k = 10 ⇒ {2,3,9,−4}



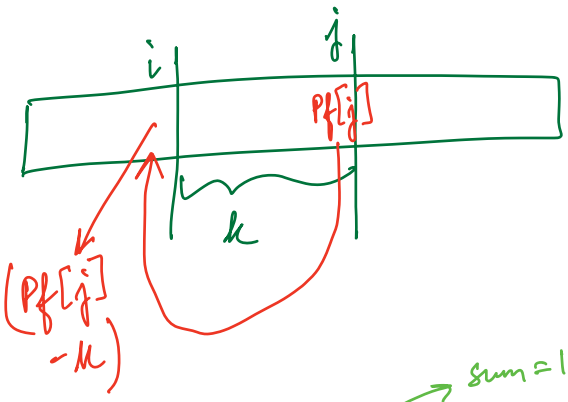
For a subarray ending at i to have a sum of k,

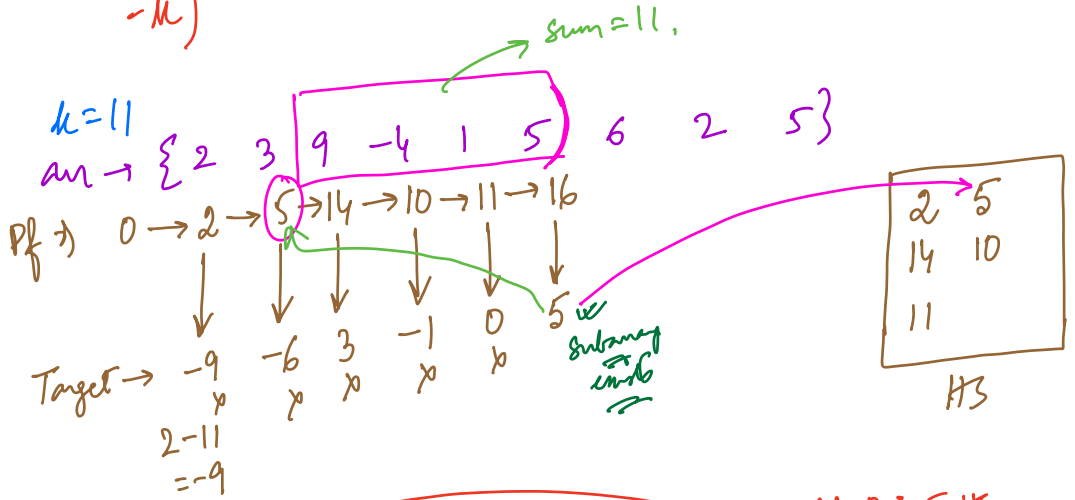
Pf[j] = Pf[i] − k should occur for some j < i.

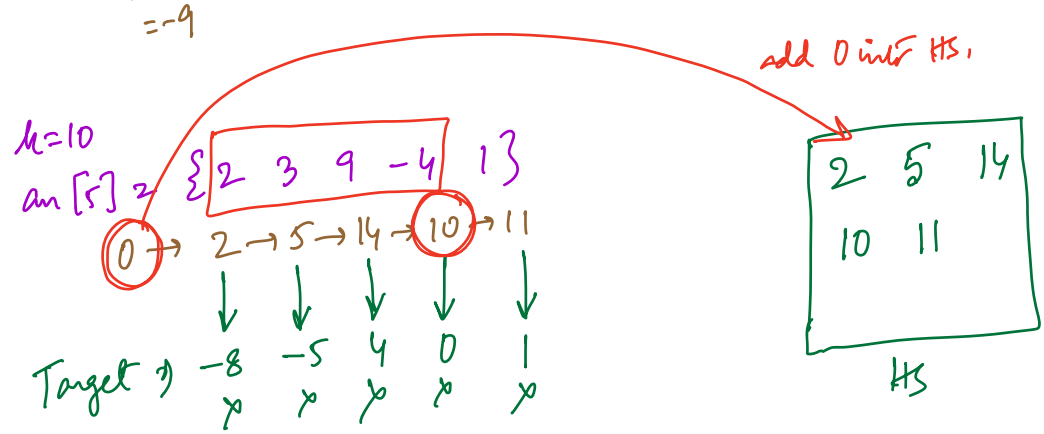
[Break till 10:39 PM]

Any subarray sum → $Pf[j] - Pf[i]$ $(j > i)$

(k)



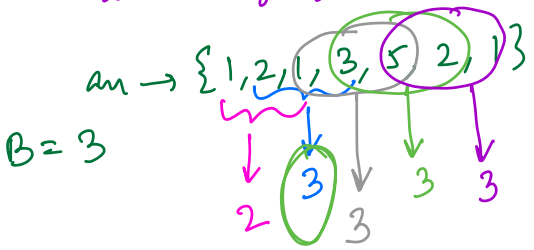


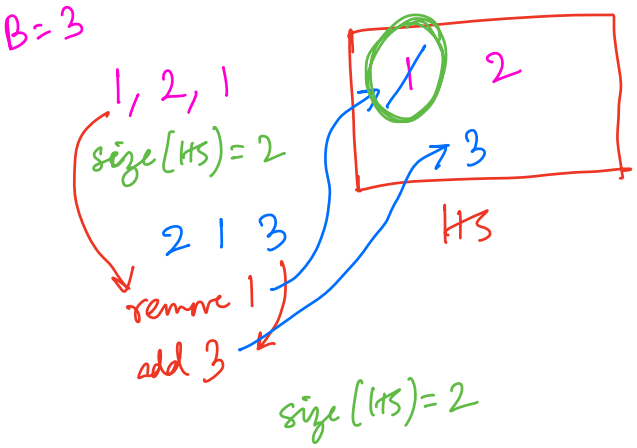


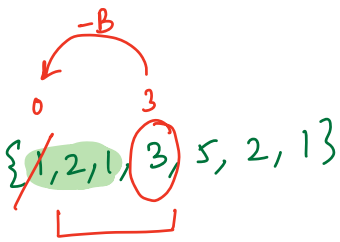
```
boolean targetSubarray(arr[], k) {
    n = len(arr)
    sum = 0
    HashSet<int> hs
    hs.add(0)
    for(i -> 0 to n-1){
        sum += arr[i]
        if(hs.contains(sum-k))
            return true
        hs.add(sum)
    }
    return false
}
```

O(n) T.C
O(n) S.C

Q4) Given arr[n], denoting the sequence of topics a learner interacted with for a module, and a nr. B denoting window size (duration). Determine the count of different topics the learner engaged with within each window of size B.







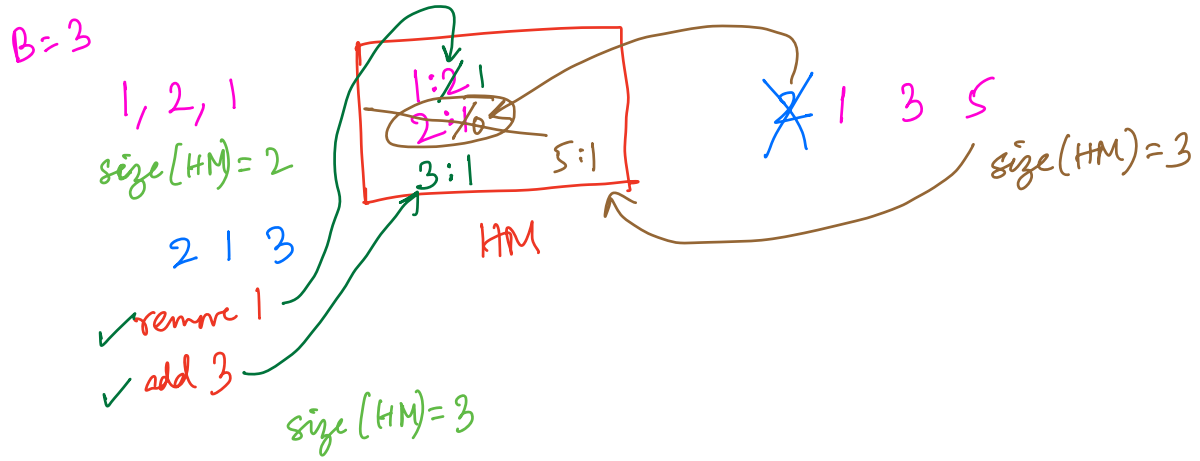
One way → new set for each window.

```
fn countDistinctElements(arr, B){
    res = {}    // empty list
    fn(i → 0 to n-B){
        set = new HashSet<>();
        fn(j → i to i+B-1){
            set.add(arr[j]);
        }
        res.add(set.size());
    }
    return res
}
```

TC → O((n-B+1)*B)
= O(n*B)

SC → O(B)

Another way → store freq with element

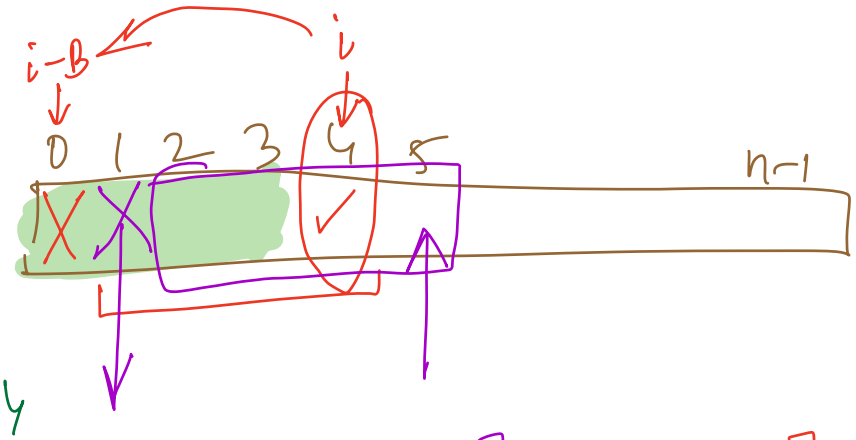



```
fn countDistinct(arr[], B){
    n = len(a)
    HashMap<int,int> hm
    for(i → 0 to B-1)
        if(hm.containsKey(arr[i]))
            hm.put(arr[i], hm.get(arr[i])+1)
        else
            hm.put(arr[i], 1)
    }
    print(hm.size())
    for(i → B to n-1){
        hm.put(arr[i-B], hm.get(arr[i-B])-1)
        if(hm.get(arr[i-B]==0))
            hm.remove(arr[i-B])
        if(hm.containsKey(arr[i]))
            hm.put(arr[i], hm.get(arr[i])+1)
        else
            hm.put(arr[i], 1)
        print(hm.size())
    }
}
```

first window O(B) T.C

next windows O(n-B) T.C

O(n) T.C.
O(B) S.C.



B=4

$i-1 \Rightarrow [i-B, i-1]$

$i \Rightarrow [i-B+1, i]$

$-a_n[i-B]$

$+a_n[i]$