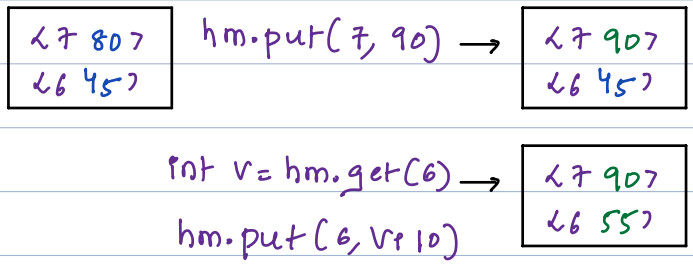
Todays Content

a) Target Sum Pair
b) Count no: of pairs with sum = k
c) Subarray with sum = k
d) No: of Subarrays with sum = k

Note: If you missed Hashing-1, plean watch recording
Others → Archived Section

a) HashMap functions

1. size() : Return no of keys
2. put(key, value): Insert key, value pair in HashMap
3. containsKey(key): Return True/False
4. get(key): If key present Return value of given key
   If key absent Return null
5. remove(key): If key present, Remove pair
   If key not present, does nothing
6. update( ): We don't have any predefined function.
   But we can update using put( )
   Ex: HashMap hm:




b) HashSet function

1. size() : Return no of keys
2. add(key): Add key in HashSet
3. contains(key): If key present return True
   If key not present return False
4. remove(key): If key present, Remove key
   If key not present, does nothing

c) HashMap & HashSet

1. Keys are not ordered in HashMap & HashSet
2. Keys are distinct in HashMap & HashSet
3. If key type is Number/Character operations take O(1)
   If key type is String operations take O(L)

# Target Sum Pair:

Given ar[N] & k, check if there exists a pair(i,j) such that

ar[i] + ar[j] == k && i != j

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Ex: ar[9] = { | 8 | 9 | 1 | -2 | 4 | 5 | 11 | -6 | 4 } |

k = 6 : ar[2] + ar[5] Yes

k = 22 : Doesn't Exist

k = 8 : ar[4] + ar[8] Yes

Ideal: Iterate on all pairs (i,j) check if their sum == k

TC: O(N²) SC: O(1)

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Ex: ar[5] = { | 3 | 2 | 6 | 8 | 4 } |

Obs:

Pairs:

| | j | j | j | j | j | i | j Loops from: [0...(i-1)] |
|---|---|---|---|---|---|---|---|
| i = 0 | (0 0) | (0 1) | (0 2) | (0 3) | (0 4) | 0 | * |
| i = 1 | (1 0) | (1 1) | (1 2) | (1 3) | (1 4) | 1 | 0..0] |
| i = 2 | (2 0) | (2 1) | (2 2) | (2 3) | (2 4) | 2 | 0..1] |
| i = 3 | (3 0) | (3 1) | (3 2) | (3 3) | (3 4) | 3 | 0..2] |
| i = 4 | (4 0) | (4 1) | (4 2) | (4 3) | (4 4) | 4 | 0..3] |

Trace:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Ex: ar[5] = { | 3 | 6 | 5 | 8 | 4 } |

k = 10, ar[i] + ar[j] = 10 ⇒ ar[j] = 10 - ar[i]

| i | ar[i] | Tar = k - ar[i] | j : [0 i-1] |
|---|---|---|---|
| 0 | 3 | 7 | j : [0, -1] * |
| 1 | 6 | 4 | j : [0, 0] * |
| 2 | 5 | 5 | j : [0, 1] * |
| 3 | 8 | 2 | j : [0 2] * |
| 4 | 4 | 6 | j : [0 3] ✓ |

```
boolean checkPair(int ar[], int k){
    int N = ar.length
    for(int i=0; i<N; i++) {
        // Target : k - ar[i]
        for(int j=0; j<i; j++) {
            if (ar[j] == k - ar[i]) {
                return true
            }
        }
    }
    return false
}
```

In Idea1 which part is taking time?

For every ar[i], we iterate from [0..i-1] & search for k

above is taking time

Idea2: Optimization with HashSet

k = 9

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Ex: ar[9] = { | 8 | 9 | 2 | -2 | 4 | 5 | 11 | -6 | 4 } |
| i = | ✓ | ✓ | ✓ | ✓ | | | | | |
| Target = | 1 | 0 | 7 | 11 | : return True | | | | |
| | * | * | * | ✓ | | | | | |

HashSet:

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

k = 4

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Ex: ar[9] = { | 8 | 9 | 2 | -2 | 4 | 5 | 11 | -6 | 4 } |
| i = | ✓ | ✓ | ✓ | | | | | | |
| Target = | -4 | -5 | 2 | : return true 2+2=4 | | | | | |
| | * | * | ✓ | : above wrong * | | | | | |

HashSet:

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

Is above logic working? Not working

Issue: Is above idea doing exact brute force approach:

In Brute force: At $i^{th}$ index j loops from: [0 i-1]

In Optimisation: At $i^{th}$ index HashSet should contain all elements from [0 i-1]

But in above logic HashSet is containing all arr[] elements

**Ideal:** Optimization using HashSet:

At $i^{th}$ index: HashSet should contain all el: {0..i-1}

k = 9

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Ex: ar[9] = { | 8 | 9 | 2 | -2 | 4 | 5 | 11 | -6 | 4 } |
| i = | 0 | 1 | 2 | 3 | 4 | 5 | | | |
| Target = | 1 | 0 | 7 | 11 | 5 | 4 | | | |
| | * | * | * | * | * | ✓ | | | |

: return True

HashSet:

| 8 9 2 -2 |
|---|
| 4 |

k = 4

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Ex: ar[9] = { | 8 | 9 | 2 | -2 | 4 | 5 | 11 | -6 | 4 } |
| i = | ✓ | ✓ | ✓ | . | . | . | . | . | . |
| Target = | -4 | -5 | 2 | | | | | | |
| | * | * | * | | | | | | |

HashSet:

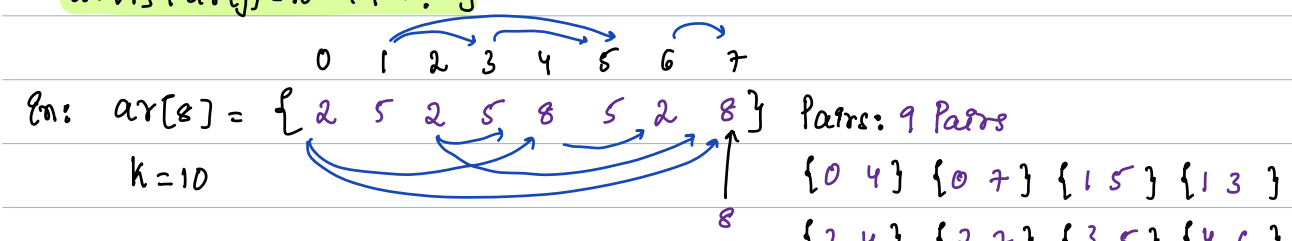
| 8 9 . . . |
|---|

```
boolean targetSum(int ar[], int k){ TC: O(N) SC: O(N)
    int N = ar.length;
    HashSet<Integer> hs = new HashSet<Integer> hs
    for(int i=0; i<N; i++){
        // Target = k-ar[i]
        if(hs.contains(k-ar[i]){
            return true
        }
        else{
            hs.add(ar[i])
        }
    }
    return false
}
```

Count Pair Sum:

Given an ar[N] count no: of pairs such that

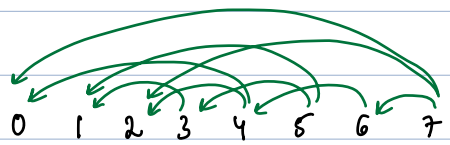
ar[i] + ar[j] = k && i != j




Pairs: 9 Pairs

{0 4} {0 7} {1 5} {1 3}
{2 4} {2 7} {3 5} {4 6}
{6 7}

Optimisation:




Eg: ar[8] = { 2 5 2 5 8 5 2 8 }

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| k = 10 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Target = | 8 | 5 | 8 | 5 | 2 | 5 | 8 | 2 |
| Count = | 0 | 0 | 0 | +1 | +2 | +2 | +1 | +3 |

HashMap

<2, 3>
<5, 3>
<8, 2>

```
int countTargetSum(int ar[], int k){ TC: O(N) SC: O(N)
    int N = ar.length;
    HashMap<Integer, Integer> hm = new HashMap<>();
    int c = 0;
    for(int i = 0; i < N; i++){
        // target = k - ar[i]
        if(hm.containsKey(k - ar[i])){
            c = c + hm.get(k - ar[i]) // freq of target => Pairs
        }
        // insert ar[i]
        if(hm.containsKey(ar[i])){ // Inc freq of ar[i] by 1
            int val = hm.get(ar[i])
            hm.put(ar[i], val+1)
        } else { hm.put(ar[i], 1) }
    }
    return c;
}
```

Q3: Given an ar[N] check if there exists a subarray with sum = k

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Eg: ar[7] = { | 2 | 3 | 9 | -4 | 1 | 5 | 6 | 2 | 5 } |

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

k = 10 : {2 3 9 -4} True

k = 15 : {-4 1 5 6 2 5} True

Idea1: Generate all subarray sums == k using carry forward

TC: O(N2) SC: O(1)

Idea2:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ar[9] = { | 2 | 3 | 9 | -4 | 1 | 5 | 6 | 2 | 5 } |
| Psum[9] = { | 2 | 5 | 14 | 10 | 11 | 16 | 22 | 24 | 29 } |

k = 12: Psum[6] : sum{0..6} - Psum[3] : sum{0..3} = sum{4..6} = 12

k = 19: Psum[8] : sum{0..8} - Psum[3] : sum{0..3} = sum{4..8}

obs: We need 2 ele in Psum[] with diff = k

a - b = k, fix a, Target = a - k

Tracing: Create HashSet at time of traversal itself

| k = 11 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ar[9] = { | 2 | 3 | 9 | -4 | 1 | 5 | 6 | 2 | 5 } |
| Cummdata a = | 0 → | 2 → | 5 → | 14 → | 10 → | 11 → | 16 | | |
| Target : | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | | | |
| a - k : | -9 | -6 | 3 | -1 | 0 | 5 | : Subarray exists | | |

HashSet

| |
|---|
| 2 5 14 |
| 10 11 |
| |

Tracing:

k = 10

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| ar[5] = { | 2 | 3 | 9 | -4 | 1 } |
| Cumm a = 0 → | 2 | 5 | 14 | 10 | 11 |
| Target: | | | | | |
| a - k : | -8 | -5 | 4 | 0 | 1 |
| | x | x | x | x | x |

HashSet

| |
|---|
| 2 5 14 |
| 10 11 |

Edge Case:
We are returning false, but we need to return True

Note: Initialize hashset with 0, to avoid Edge Cases

k = 10

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| ar[7] = { | 2 | 3 | 9 | -4 | 1 } |
| Cumm a = 0 → | 2 | 5 | 14 | 10 | |
| Target: | | | | | |
| a - k : | -8 | -5 | 4 | 0 | : Subarray exists |
| | x | x | x | ✓ | |

HashSet

| |
|---|
| 0 2 5 |
| 14 |

```
boolean targetSubarray(int ar[], int k){   TC: O(N) SC: O(N)
    int N = ar.length;
    long a = 0; // Cummulative sum, long => to avoid overflow
    HashSet<Long> hs = new HashSet<>();
    hs.add(0)
    for(int i=0; i<N; i++){
        a = a + ar[i] // CummlativeSum => // target = a - k
        if(hs.contains(a-k)){ // Subarray exist
            return true
        }
        else{
            hs.add(a);
        }
    }
    return false;
}
```

48 Count Subarray Sums = k

Given ar[N], count no: of subarrays with sum = k

k = 6

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Eg: ar[ ] = { | 3 | 7 | 4 | -4 | 1 | -2 | 5 | 6 | 2 | -8 | -10 } |
| Cumm a = 0 → | 3 | 10 | 14 | 10 | 11 | 9 | 14 | 20 | 22 | 14 | 4 |
| Target a-k : | -3 | 4 | 8 | 4 | 5 | 3 | 8 | 14 | 16 | 8 | -2 |
| | x | x | x | x | x | +1 | x | +2 | x | x | x |

ans = 3

HashMap

| | |
|---|---|
| <0, 1> | <11, 1> |
| <3, 1> | <9, 1> |
| <10, 2> | <20, 1> |
| <14, 3> | <22, 1> |
| <4, 1> | |

```
int targetSubarray(int ar[], int k){ TC: O(N)
                                     SC: O(N)
    int N = ar.length;
    HashMap<Long, Integer> hm;
    long a = 0; // Cummlatin sum
    hm.put(0,1);
    long cnt=0;
    for(int i=0; i< N; i++){
        a = a + ar[i] // Cummlatin sum
        // target = a-k, get frequency
        if( hm.containskey(a-k)){
            cnt = cnt + hm.get(a-k)
        // Insert a in hm
        if( hm.containskey(a)){ // inc freq of a
            int val = hm.get(a)
            hm.put(a, val+1)
        else{
            hm.put(a, 1)
    }
    return cnt;
}
```