# Agenda

1. Longest Subarray with sum = 0
2. Pair sum
3. Count of Pair Sum
4. Check if there exits a subarray with sum = 0
5. Check if there exits a subarray with sum = K

**< *Question* > :** Given an arr[ N ] and K.

Check if there exists a pair ( i , j ) such that, arr[ i ] + arr[ j ] = K && i != j

arr →

| 8 | 9 | 1 | -2 | 4 | 5 | 11 | -6 | 4 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

K = 6 true

K = 22 false

K = 8 true

***Quiz :***

| 3 | 5 | 1 | 2 | 1 | 2 |
|---|---|---|---|---|---|

, K = 7 ans → true

***BF Idea***

Consider all the pairs.

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

[ T.C → O(N²) , S.C → O(1) ]

Idea → use - hashset

arr →

| 8 | 9 | 1 | -2 | 4 | 5 | 11 | -6 | 4 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

8, 9, 1,
-2, 4, 5
11, -6

set

K = 6 ✓

K = 22 → Approach will fail here. ✗

K = 8

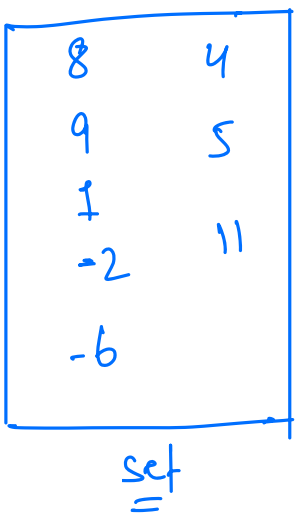
Two approaches

- use - hashset on the go
- Create hashmap initially.

idea-2. use hashset on the go -

arr →

| 8 | 9 | 1 | -2 | 4 | 5 | 11 | -6 | 4 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |




set

K = 6 ⇒ true.

K = 22 ⇒ false.

**< / > Code**

```
Hashset<int> set;

for( i=0; i<N; i++){
    if( set.search( k-arr[i] ) == true){
        return true;
    }
    set.add( arr[i] );
}

return false;
```

T.C → O(N)
S.C → O(N)

---

arr[] → [3, 5, 1, 2, 1, 2] , K=3
0 1 2 3 4 5

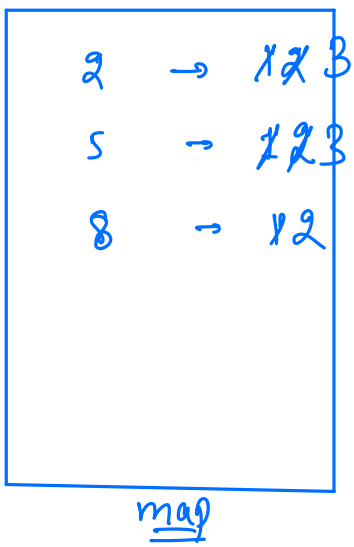
(2,3)
(2,5)
(4,3)
(4,5)
} → 4 pairs

# Count the number of pairs with sum = K

arr[ ] → [ 2 5 2 5 8 5 2 8 ] K = 10

0 1 2 3 4 5 6 7

idea → Hashmap on the go




K = 10

ans = 0 + 1 + 2 + 2 + 1 + 3

= 9

## </> Code

```
Hashmap < integer, integer > map ;

ans = 0;

for( i = 0; i < N; i++){
    if( map.search( K - arr[i] ) == true){
        ans += map.get( K-arr[i]);
    }

    if( map.search ( arr[i]) == false){
        map.add( arr[i], 1);
    }
    else{
        val = map.get (arr[i]);
        map.update(arr[i], val + 1);
    }
}
return ans;
```

adding curr element in map.

T.C → O(N)
S.C → O(N)

# Subarray with Sum 0

**< *Question* > :** Given an array of N elements. Check if there exists a subarray with sum equal to 0. $(1 \le N \le 10^5)$

N = 11 arr → [ 2 2 1 -3 4 3 1 -2 -3 2 ]

0 1 2 3 4 5 6 7 8 9

ans = true.

**BF Idea**

→ Consider all the subarrays and for every subarray check if sum = 0 or not.

```
for( i=0; i<N; i++){
    for(j=i; j<N; j++){
        sum=0;
        for(k=i; k<=j; k++){
            sum += arr[k];
        }
        if(sum==0) {return true;}
    }
}
return false;
```

T.C → $O(N^3)$
S.C → $O(1)$

using C.F or psum[]

$N^2$

## Idea -2 use psum[]

N = 10

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

psum[ ] → [ 2 4 5 2 6 9 10 8 5 7 ]

[ if elements are repeating in psum[], then we can say that we have a subarray with sum = 0

⇓

Simply check for repeating element in psum[]. ]

arr → [ -2 -1 3 5 ]

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|

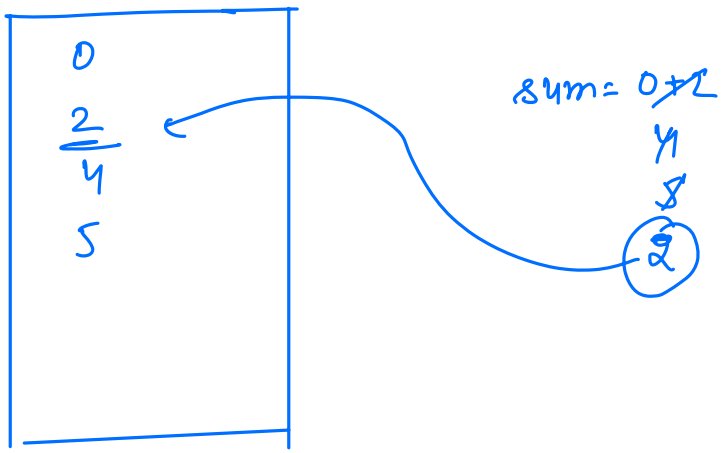
psum[ ] → [ -2 -3 0 5 ]

To handle this case → insert 0 initially in the set.

dry-run

N = 10

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




Code →

```
Hashset <Integer> set;
set.add(0);
Sum = 0;
for( i = 0; i < N; i++){
    sum += arr[i];
    if( set.search(sum) == true){
        return true;
    }
    set.add(sum);
}
return false;
```

T.C → O(N)
S.C → O(N)

**< Question > :** Given an array[ N ]. Check if there is a subarray with sum = k ( $1 \le N \le 10^6$ )

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

K = 11 → true
K = 10 → true

arr[ ] → [ 5 10 20 100 105 ]
0 1 2 3 4

K = 110 → false.

**BF Idea**

→ Consider all the subarrays and for every subarray check if sum = K or not.

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

T.C → $O(N^3)$
S.C → $O(1)$

using C.F or psum[]

$O(N^2)$

**Idea** ⇒ check for a pair with difference = K in psum[].

⇒ hashset on the go

arr[ ] → [ 2 3 9 -4 1 5 6 2 5 ] K = 15

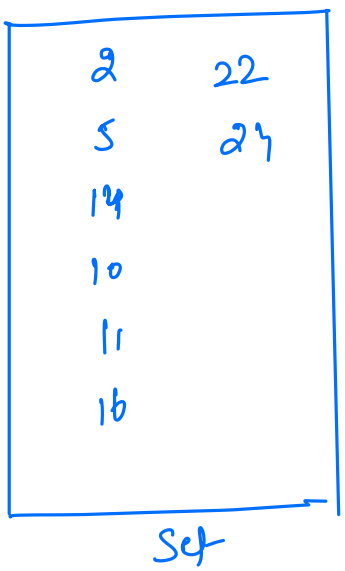
0 1 2 3 4 5 6 7 8

psum[] → [ 2 5 14 10 11 16 22 24 29 ] , K=15

0 1 2 3 4 5 6 7 8

a - ? = K

a - K = ?



## </> Code

```
HashSet<int> set;

sum=0;

for(i=0; i<N; i++){
    sum += arr[i];
    if(set.search(sum-K) == true){
        return true;
    }
    set.add(sum);
}

return false;
```

T.C → O(N)
S.L → O(N)

→ **Longest subarray with sum = 0**

arr[] → [ 2 5 3 4 -6 -1 5 -2 7 -10 ]

0 1 2 3 4 5 6 7 8 9

psum[] → [ 2 7 10 14 8 7 12 10 17 7 ]

idea → store the first occurence of sum.

arr[] → [ 2 5 3 4 -6 -1 5 -2 7 -10 ]

0 1 2 3 4 5 6 7 8 9

sum → f.o.

| | | |
|---|---|---|
| 0 | → | -1 |
| 2 | → | 0 |
| 7 | → | 1 |
| 10 | → | 2 |
| 14 | → | 3 |
| 8 | → | 4 |
| 12 | → | 6 |
| 17 | → | 8 |

map

sum = 2 7 10 14 8 7 12 10 17 7

ans = 4

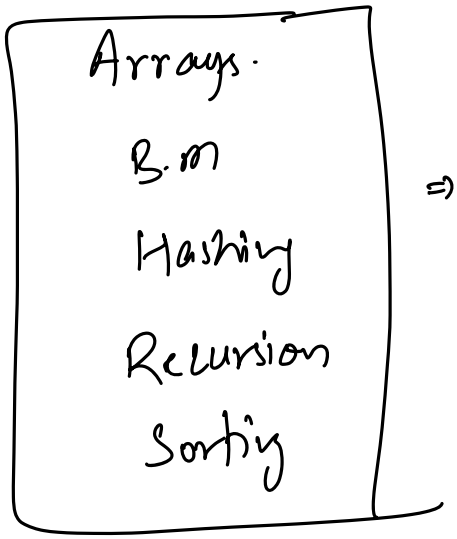
ans = Max(ans, i - map.get(sum))

Code →

todo:

---

Contest. –




⇒ Assign. & Additional

→ Pair sum divisible by M.