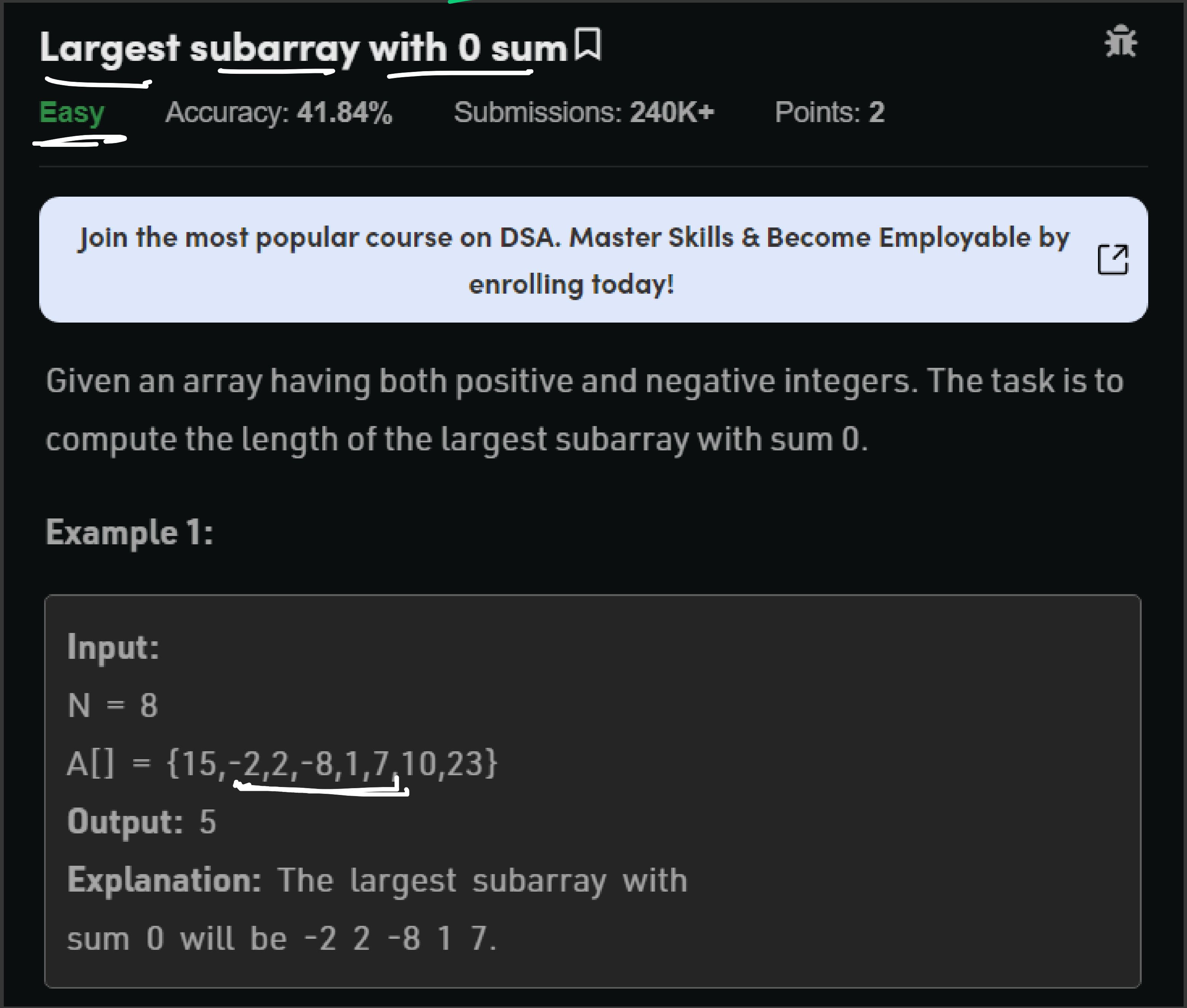
Longest subarray length with zero sum



Largest subarray with 0 sum

Easy   Accuracy: **41.84%**   Submissions: **240K+**   Points: **2**

Given an array having both positive and negative integers. The task is to compute the length of the largest subarray with sum 0.

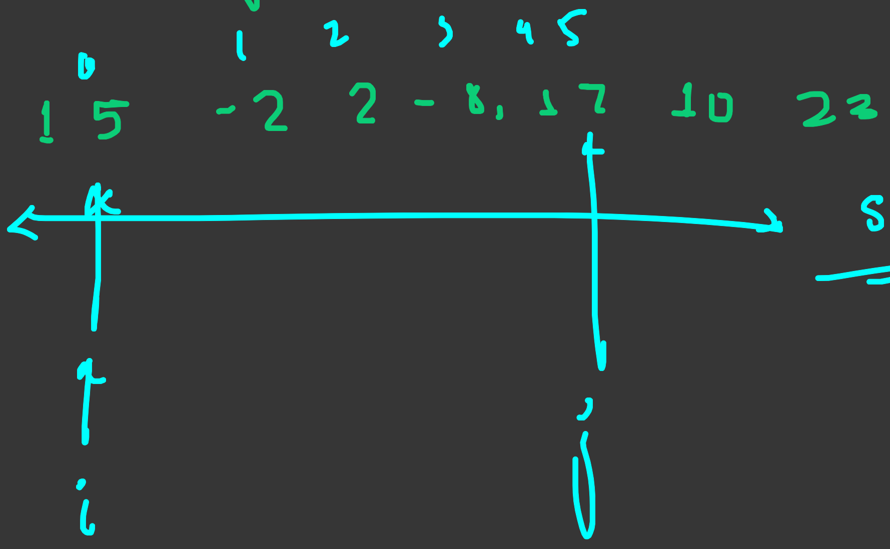
Example 1:

```
Input:
N = 8
A[] = {15,-2,2,-8,1,7,10,23}
Output: 5
Explanation: The largest subarray with
sum 0 will be -2 2 -8 1 7.
```

arr[] = { 15, $\overbrace{-2, 2, -8, 1, 7}^{0}$, 10, 23 } = 5

Brute force:

- Generate all subarray and sum it and whosoever sum is zero count the length and find the longest length.



subarray → sum if sum = 0 length

count

max

5-0

5+1

$O(N^2)$

```
for(i=0; i<n; i++)
    for(j=i; j<n; j++)
        sum += arr[j];
        if (sum == 0)
            ~~return~~ length = max(length, j-i+1);
```

$i \longrightarrow j$

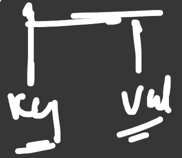
$\boxed{j - i + 1}$

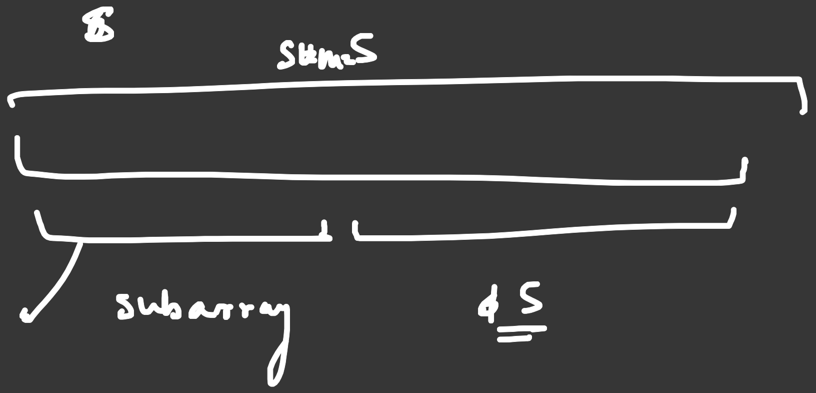
Time Complexity: $O(N^2)$

Optimal Solution:

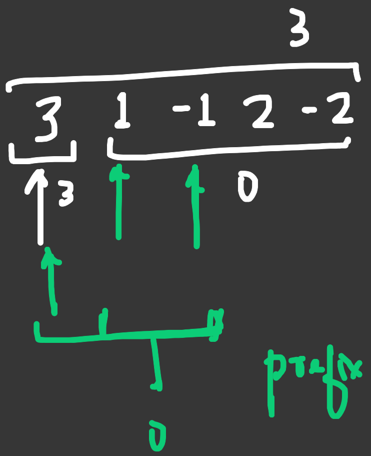
subarray sum = 0

Hashmap <int, int> mp;





0



prefix = 3 + 1 = 4 - 1 = 3

S

- prefix Sum
- hashmap

eg: 1 -1 3 -2 2 -8 1 7 9 6

0 1 2 3 4 5 6 7 8 9

sum = ~~0~~ 1 - 1 = ~~0~~ 3 - 2 = 1 + 2 = 3

max Subarray = 0

-5

1

(3, 2)

(1, 0)

↳ Hashmap

Algo

↳ if (sum==0)

    maxi = i+1;

else{

    if(mp.find(sum) != mp.end())

    currind - mp[sum] → length

    map → ind

    maxi =

1 + 1 = 2 - 2 = 0

1 1 - 2

Pseudocode

```
unordered_map <int, int> mp;

for (i=0; i<n; i++)
    sum += arr[i];

    if (sum == 0)  maxi = i+1;
                              is sum present in the map?
    else if ( mp.find( sum) != mp.end())
            maxi = max( maxi, i - mp[sum]+1 );

    else  mp.insert({sum, i});
```

T.C. = $\boxed{O(N \log N)}$ ↓ map

S.C = O(N)

O(N)

↑

(unordered map
assume) operation
O(1)