

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:

<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

Leetcode May Challenge DAY: 25

Let us change all 0 in our array with -1. Then we need to find the contiguous subarray with sum equal to zero. There are still $O(n^2)$ different contiguous subarrays, but we do not need to evaluate them all. If we evaluate all cumulative sums, then we need to find two cumulative sums which are equal and have the biggest distance. Example:

```
nums = [1, 0, 0, 1, 1, 1, 1, 0, 1] -> [1, -1, -1, 1, 1, 1, 1, -1, 1] -> [1, 0, -1, 0, 1, 2, 3, 2, 3]
```

and the biggest distance between equal elements is 4, element number 0 and element number 4.

We are going to keep `ind` hashtable, where for each value of cumulative sum we keep indexes for the first and for the last element with this value in our cumulative sum. Continue with our example:

```
ind[0] = [1,3]
```

```
ind[1] = [0,4]
```

```
ind[-1] = [2,2]
```

```
ind[2] = [5,7]
```

```
ind[3] = [6,8]
```

Complexity We need to go through our `nums` twice: first, when we build cumulative sums and then when we create our `ind` hash-table, hence we have $O(n)$ time complexity. Also, there can be at most $2n + 1$ elements in our hash-table - with values from $-n$ to n (in fact not more than n of them will be there, because we have only n cumulative sums). So time complexity is also $O(n)$.

1. Python

class Solution:

```
def findMaxLength(self, nums: List[int]) -> int:
```

```
    nums = list(accumulate([x * 2 - 1 for x in nums]))
```

```
    ind = defaultdict(list)
```

```
    ind[0] = [-1,-1]
```

```
    for i in range(len(nums)):
```

```
        if not ind[nums[i]]:
```

```
            ind[nums[i]] = [i,i]
```

```
        else:
```

```
            ind[nums[i]][1] = i
```

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

```
max_len = 0  
for i in ind:  
    max_len = max(max_len, ind[i][1] - ind[i][0])  
return max_len
```



Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

Explanation:-

Suppose, $A = [0, 0, 0, 1, 1, 0]$

We need to find max length subarray with equal 0 and 1s.

Let's solve this example

-> replace every 0 in A to -1

Now, $A = [-1, -1, -1, 1, 1, -1]$

Algorithm:-

-> traverse through whole array keep track of the sum

-> for each index store sum in map

-> if we get sum that is already existed in map,

-> that means we get a combination of 0 and 1's

-> So, update max.

$A = [-1, -1, -1, 1, 1, -1]$

map = {0, -1} -> initially sum is 0

sum = 0, max = 0, index = 0

-> traverse through array

index = 0

sum = -1

-1 is not in map, so add sum and index to map

map = {0 : -1, -1 : 0}

index = 1

sum = -2

-2 is not in map, so add sum and index to map

map = {0 : -1, -1 : 0, -2 : 1}

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

index = 2

sum = -3

-3 is not in map, so add sum and index to map

map = {0 : -1, -1 : 0, -2 : 1, -3 : 2}

index = 3

sum = -2

-2 is already in map

so, update max

max = max(max, index - map[-2])

max = max(0, 3 - 1)

so, max = 2.

index = 4

sum = -1

-1 is already in map

so, update max

max = max(2, index - map[-1])

max = max(2, 4 - 0)

max = 4

index = 5

sum = -2

-2 is already in map

so, update max

max = max(4, index - map[-2])

max = max(4, 5 - 1)

max = 4

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

So, **max** subarray **length** is **4**.

IF YOU HAVE **ANY** DOUBTS, FEEL FREE **TO** ASK

IF YOU UNDERSTAND, **DON'T FORGET TO UPVOTE.**

TIME:- $O(N)$

SPACE:- $O(N)$



ENGINEERING LIBRARY

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCjLMu9mayAibQST1eWwq0cg>

2. C++

```
class Solution {  
public:  
    int findMaxLength(vector<int>& nums) {  
        unordered_map<int, int> map;  
        map[0] = -1;  
        int ans = 0, sum = 0;  
        for(int i = 0; i < nums.size(); i++){  
            sum += (nums[i] == 0) ? -1 : 1;  
            if(map.find(sum) != map.end())  
                ans = max(ans, i - map[sum]);  
            else  
                map[sum] = i;  
        }  
        return ans;  
    }  
};
```

Follow me for more interesting programming questions and RAW Code visit my GitHub profile:
<https://www.github.com/shubhamthrills> <https://www.linkedin.com/in/shubhamsagar>

Subscribe our YouTube Channel for more videos: <https://www.youtube.com/channel/UCJLMu9mayAibQST1eWwq0cg>

3. JAVA

```
class Solution {  
    public int findMaxLength(int[] nums) {  
        HashMap<Integer, Integer> map = new HashMap<>();  
        map.put(0, -1);  
        int sum = 0, max = 0;  
        for(int i = 0; i < nums.length; i++){  
            sum += (nums[i] == 0) ? -1 : 1;  
            if(map.containsKey(sum))  
                max = Math.max(max, i - map.get(sum));  
            else  
                map.put(sum, i);  
        }  
        return max;  
    }  
}
```