```
// Given an array having both positive and negative
integers. The task is to compute the length of the largest
subarray with sum 0.
// Expected Time Complexity: O(N).
// Expected Auxiliary Space: O(N).
```

Brute Force: Two nested for loops. Outer loop for Starting point of sub-array and inner loop for Ending point of sub-array.

Optimized: Time complexity: O(n); Space complexity: O(n)

Create a variable sum and while traversing the input array, for every index add the value of the element into the sum variable and then store the sum-index pair in a *hash-map*. So, **if the same value appears twice in the array, it will be guaranteed that the particular array will be a zero-sum sub-array**.

Mathematical Proof:

prefix(i) = arr[0] + arr[1] +...+ arr[i]
prefix(j) = arr[0] + arr[1] +...+ arr[j], j > i
ifprefix(i) == prefix(j) then prefix(j) - prefix(i) = 0 that means *arr[i+1] + .. + arr[j] = 0*, So a sub-array has zero sum , and the length of that sub-array is j-i+1

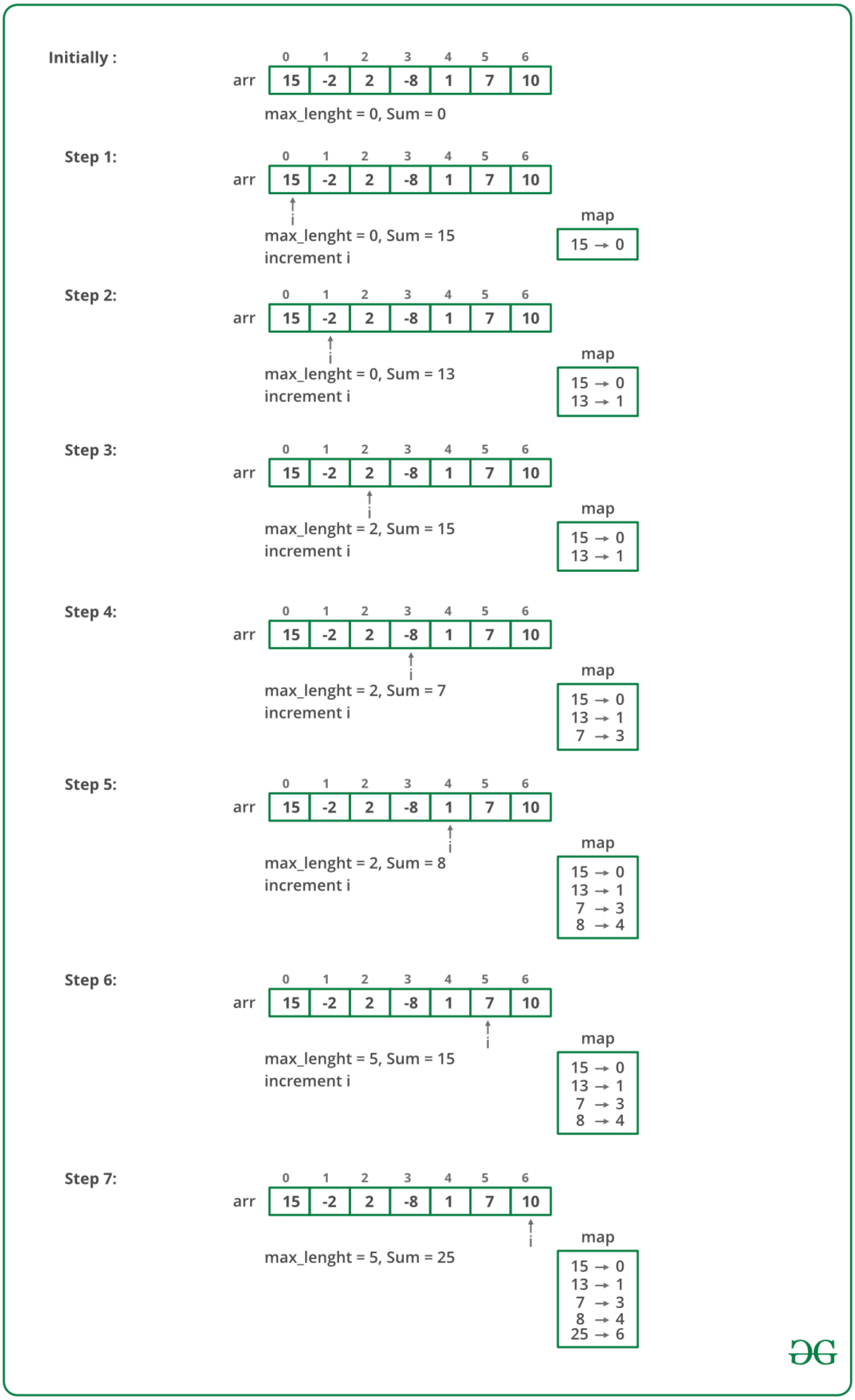
- If a sum value comes out to be zero for an index, it means that from 0^{th} index to that index there is a zero sum sub-array.

- The **hash-map will store** <sum, index> **only the very first index when a particular sum value is encountered**. That <sum, index> value will not be changed when we get the same sum value again.

- When we get a particular sum value again, we'll compare the maximum length with (that index – the first index of that sum value). Because the **maximum length of zero sum sub-array will be the one with the largest difference b/w the corresponding index of that sum value's last occurrence and the first occurrence**.

- The unordered hash-map has space complexity of O(n) when we define it. Operations like checking, finding a key, insertion, deletion have time complexity of O(1). We couldn't use vector because –
  - ➢ Sum value would become the index of this vector. So, we would have needed a vector of very large length.
  - ➢ Checking if a sum value has occurred previously or not, would be nearly impossible, since that sum value index would be already

present regardless of it has previously occurred or not. What initial value we could have given to such vector ?


Initially :
arr 15 -2 2 -8 1 7 10
max_lenght = 0, Sum = 0
Step 1:
max_lenght = 0, Sum = 15
increment i
map
15 → 0
Step 2:
max_lenght = 0, Sum = 13
increment i
15 → 0
13 → 1
Step 3:
max_lenght = 2, Sum = 15
increment i
15 → 0
13 → 1
Step 4:
max_lenght = 2, Sum = 7
increment i
15 → 0
13 → 1
7 → 3
Step 5:
max_lenght = 2, Sum = 8
increment i
15 → 0
13 → 1
7 → 3
8 → 4
Step 6:
max_lenght = 5, Sum = 15
increment i
15 → 0
13 → 1
7 → 3
8 → 4
Step 7:
max_lenght = 5, Sum = 25
15 → 0
13 → 1
7 → 3
8 → 4
25 → 6