Subarray sum equals K

arr = [9, 4, 20, 3, 10, 5]

tar = 33

return count of unique subarray

⇒ [9, 4, 20] ⇒ (2)

[20, 3, 10]

(1) Brute force

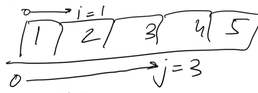
take sum of all subarrays

```
for (i=0 to n) {
    sum = 0
    for (j=i to n) {
        sum += arr[j]
        if (sum == k) count++
    }
}
```

TC: $O(n^2)$

(2) Optimal solution

(1) Prefix Sum



ps[i] = 3

ps[j] = 10

subArrSum(i+1, j) = ps[j] - ps[i]

⇓

subArrSum(i, j) = ps[j] - ps[i-1]

(2) K = ps[j] - ps[i-1]

ps[i-1] = ps[j] - k

9	4	20	3	10	5

9	13	33	36	46	51	ps

```
prefixSum[n]
ps[0] = arr[0]
for (i=0 to n) {
    ps[i] = ps[i-1] + arr[i]
}
```

case 1:

9	4	20	3	10	5

j

9	13	33	36	46	51	ps

ps[i-1] = ps[j] - k

= 46 - 33

= 13

∴ we get our subarray

case 2:

9	4	0	20	3	10	5

j

9	13	13	33	36	46	51

Now 13 is at 2 indexes so we will increase the count by 2 because 2 valid subarrays exist (one with 0 & one without)

we will use unordered map for this

ps	freq
9	1
13	2
33	1
36	1

When j = 10 then we need 13 so count is incremented by 2

```
for (j = 0 to n) {
    if (ps[j] == k) count++;
    val = ps[j-k]
    if (m.find(val) != m.end()) {
        count += m[val];
    }
    if (m.find(ps[j]) == m.end()) {
        m[ps[j]] = 0
    }
    m[ps[j]]++;
}
return count
```

TC: O(n)