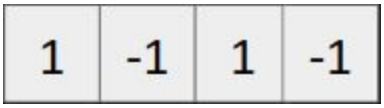
Number of subarrays with sum zero

Problem

Given an array a[] of size n. Our task is count the number of subarrays with sum zero.

Example



Subarrays with sum zero are:

Number of subarrays with sum equal to zero are 6.

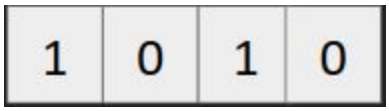
Brute force approach

1. Iterate over all the subarrays using nested loops, simultaneously calculate *sum* and maintain a variable *cnt.* Increment count whenever sum adds up to zero.

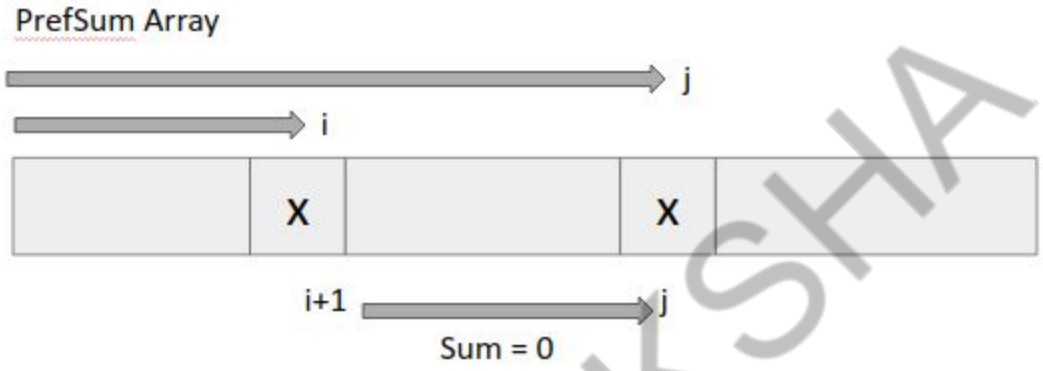
Time Complexity: $O(n^2)$

Optimized approach

1. Compute prefix sum array. For the above example, It will look like



2. Main idea: For an array of prefix sum if a value repeats in prefix sum array at indices *i* and *j*, it denotes sum of elements from indices *i+1* till *j* is zero.



So, we have to find the number of ways in which we can choose two same valued elements in the PrefSum array. (Special case for PrefSum[i]=0, as they add upto zero from starting index)

3. Declare a map (say freq) denoting frequency of elements in the prefix sum array.

map<int,int> freq;

4. Iterate over the PrefSum array and just do

freq[PrefSum[i]]++;

5. After our freq map is created, then apply permutation and combination formula to choose 2 elements from the group of m identical items. ($^{m}C_{2}$) for each key.

Code

```
void solve()
{
    int n;
    cin >> n;

    vi a(n);

    rep(i,0,n)
        cin >> a[i];

    map<int,int> cnt;

    int prefsum = 0;

    rep(i,0,n)
    {
        prefsum += a[i];
        cnt[prefsum]++;
    }

    int ans = 0;

    for(auto it:cnt)
    {
        int c = it.ss;

        ans += (c*(c-1))/2;

        if(it.ff == 0)
            ans += it.ss;
    }
    cout << ans << endl;
}

signed main()
{
    speed;

    solve();
    return 0;
}
```