Abstract

Given an array of n integers, your task to count the number of subarrays having sum is equal to x
.

Tutorial

Let's consider the prefix sum of the array. For more meanings, prefix sum of an array $x_1, x_2, ..., x_n$ is a array $y_0, y_1, y_2, ..., y_n$ (the sums of prefixes of the input x). More formally,

$$y_0 = 0$$
$$y_1 = x_1$$
$$y_2 = x_1 + x_2$$
$$...$$
$$y_n = x_1 + x_2 + ... + x_n$$

So let's call the prefix sum of input array a is sum. The problem is to find the number of subarray having sum x, so that means we have to find the number of i such that $sum[i] - sum[j] = x$ with $0 < j < i$.

The first idea that we can solve this problem by using two loops, one to iterate i and the other to iterate j. Following is a pseudo code of this idea:

```
cnt = 0
for i in [1:n]
    for j in [0:i-1]
        if sum[i] - sum[j] == x
            cnt += 1
return cnt
```

The time complexity of above algorithm is $O(n^2)$, which is not possible with n up to 10^5.

But notice that we can calculate the number of j when iterating i. Denote $F[i, p]$ as the number of indexs $j < i$ satisfying $sum[j] = p$. According the above observation, for each i we can calculate F in a linear complexity and the answer of the number j is $F[i, sum[i] - x]$.

```
    cnt = 0
    for i in [1:n]
        cnt += F[sum[i] - x]
        F[sum[i]] += 1
```

As the above pseudo code, we can reduce the variable i of F. To store F we can use a common data structure `Hash table`.

The complexity of this algorithm is $O(np)$ with p depending on the implementation of `hash table`.