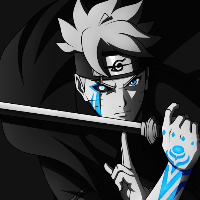
**Problem: Continuous subarray sum**

**Link :** https://leetcode.com/problems/continuous-subarray-sum/solutions/5276981/prefix-sum-hashmap-patterns-7-problems

**Prefix Sum + HashMap Patterns ( 7 problems )**

[](https://leetcode.com/u/mercer80/)

[mercer80](https://leetcode.com/u/mercer80/)



1078

647

3 hours ago

C++

Hash Table

Prefix Sum

**Prefix Sum + Hashmap Approach**

We can apply this approach in problems related to Subarrays, problems where we are asked to

1. Count the subarrays with some condition
2. Find Maximum Length Subarray with some condition
3. Check if subarray exists with some given condition

Mainly the problems dealing with Subarray sum .

*Approach works on the following principle*

If the prefix sum up to ith index is X, and the  
prefix sum up to jth index is Y and  
it is found that Y = X +k, then the required subarray is found with i as start index and j as end index.

To store the index value and the sum of elements up to that index a hashmap can be used.

*Why we need this approach of Hashmap +Prefix Sum ?*  
So basically for subarray related problems Brute Force method takes O(n^2) time to process each subarray + extra time in processing .  
But with this approach only O(n) time is taken .  
Now you might wonder that what is stopping you from using Sliding window approach for such problems

Sliding window is only applicable when we know for sure if the prefixsum is an increasing or decreasing function(i.e. Monotonous in nature)

So for problems where negative input is given this approach of PrefixSum + Hashmap is the best way to solve such problems.

**1. Count Subarrays with some given condtion**

Data Structures needed :

* **Integer Variable** for cumulative sum
* **Unordered Map**
* Map **Initialisation** : **mp[0]=1**

Lets solve 3 problems using Prefix Sum + Hashmap pattern

[**✏️Problem 1.1 --> 560. Count Subarray sum equals k**](https://leetcode.com/problems/subarray-sum-equals-k/)

**Condition : Sum of subarray equals k**  
Negative values **ALLOWED** in Input

Code

/\*Given an array of integers arr and an integer k,

return the total number of subarrays whose sum equals to k.\*/

class Solution {

public:

int subarraySum(vector<int>& arr, int k) {

int count = 0;

int sum = 0;

unordered\_map<int, int> mp;

mp.insert({0, 1});

for (int it : arr) {

sum += it;

count += mp[sum - k];

mp[sum]++;

}

return count;

}

};

[**✏️Problem 1.2 --> 974. Count Subarray Sums Divisible by K**](https://leetcode.com/problems/subarray-sums-divisible-by-k/)

**Condition** : Subarrays that have a sum divisible by k.  
Negative values **ALLOWED** in Input

Code

class Solution {

public:

int subarraysDivByK(vector<int>& nums, int k) {

int n = nums.size();

int count = 0;

int sum = 0;

unordered\_map<int, int> mp;

mp[0] = 1;

for (int i : nums) {

sum = (sum + i) % k;

if (sum < 0)

sum = sum + k; // ADD k if sum negative to make it positive

count += mp[sum];

mp[sum]++;

}

return count;

}

};

[**✏️Problem 1.3--> 930. Count Binary Subarrays with given SUM**](https://leetcode.com/problems/binary-subarrays-with-sum/)

**Condition** : (Binary)Subarrays with given Sum

**Negative values are not allowed in input. Because of this reason, we can also solve this problem using Sliding window too.**

**Sliding window is only applicable when we know for sure if the prefixsum is an increasing or decreasing function(i.e. Monotonous in nature)**

Code

/\*Given a binary array nums and an integer goal,

return the number of non-empty subarrays with a sum goal.\*/

class Solution {

public:

int numSubarraysWithSum(vector<int>& nums, int goal) {

int count = 0;

int sum = 0;

unordered\_map<int, int> mp;

mp.insert({0, 1});

for (int it : nums) {

sum += it;

count += mp[sum - goal];

mp[sum]++;

}

return count;

}

};

**2. Maximum length Subarray with given condition**

Data Structures needed :

* **Integer Variable** for cumulative sum
* **Unordered Map**
* Map **Initialisation** : **mp[0]=-1** Here we are dealing with index that's why we initalize it to -1

[**✏️Problem 2.1 --> 525.Contiguous Array**](https://leetcode.com/problems/contiguous-array/)

**Condition** : Binary Subarray with an equal number of 0 and 1

Code

/\*Given a binary array nums, return the maximum length

of a subarray with an equal number of 0 and 1\*/

class Solution {

public:

int findMaxLength(vector<int>& nums) {

int ans=0;

unordered\_map<int,int>mp;

mp[0]=-1;

int one=0,zero=0;

for(int i=0;i<nums.size();i++){

if(nums[i]==0)zero++;

else one++;

int diff=zero-one;

if(mp.count(diff))

ans=max(ans,i-mp[diff]);

else

mp[diff]=i;

}

return ans;

}

};

**✏️Problem 2.2 --> 325.Maximum Size Subarray Sum Equals k (Premium)]**

***Problem is Available on other Platforms.***

Problem Statement : Given an array nums and a target value k, find the maximum length of a subarray that sums to k. If there isn't one, return 0 instead.

Condition: subarray that sums to k

Negative values **ALLOWED** in Input

Code

//arguments of this code might differ

//from leetcode version of this problem but

//the idea reamins same

int lenOfLongSubarr(int A[], int N, int K) {

int pre\_sum = 0; //prefix sum

int res = 0;

unordered\_map<int, int> mp; {pref sum , index}

mp[0] = -1;

for(int i = 0; i < N; i++) {

pre\_sum += A[i];

if(mp.find(pre\_sum - K) != mp.end()) // pre\_sum - K found in hash

res = max(res, i - mp[pre\_sum - K]);

if(mp.find(pre\_sum) == mp.end()) // Check if prefix\_sum exists in hash

mp[pre\_sum] = i;

}

return res;

}

};

**✏️Problem 2.3 -->**[**1658. Minimum Operations to Reduce X to Zero**](https://leetcode.com/problems/minimum-operations-to-reduce-x-to-zero/)

**Problem Reduced to Above problem of Maximum size subarray sum**

Code

/\*given an integer array nums and an integer x.

In one operation, you can either remove the leftmost

or the rightmost element from the array nums

and subtract its value from x

Return the minimum number of operations

to reduce x to exactly 0 if it is possible, otherwise, return -1.\*/

class Solution {

public:

int minOperations(vector<int>& nums, int x) {//start

int n = nums.size();

int total = accumulate(nums.begin(), nums.end(), 0);

int rem = total - x;

if (rem == 0)

return nums.size();

int length = maxSubArrayLen(rem, nums);

if (length == 0)

return -1;

return n - length;

}

int maxSubArrayLen(int k, vector<int>& A) {//Code for Maximum size subarray given sum

int sum = 0;

int res = 0;

unordered\_map<int, int> mp;

mp[0] = -1;

for (int i = 0; i < A.size(); i++) {

sum += A[i];

if (mp.find(sum - k) != mp.end())

res = max(res, i - mp[sum - k]);

if (mp.find(sum) == mp.end())

mp[sum] = i;

}

return res;

}

};

**3. Check if a subarray exists with given condition**

**✏️Problem 3.1 :**[**523. Continuous Subarray Sum**](https://leetcode.com/problems/continuous-subarray-sum/)

Given an integer array nums and an integer k, return true if nums has a good subarray or false otherwise.

Conditions: Length of subarray is at least two  
The sum of the elements of the subarray is a multiple of k

**Why sliding window not applicable here despite positive input?**  
*We need to check if the sum of the subarray is a multiple of k, which adds complexity. We can't determine this condition by simply adjusting the window size. if instead of modulo the sum was asked to be compared then we can for a logic to decrement or increment window size*

Code

class Solution {

public:

bool checkSubarraySum(vector<int>& nums, int k) {

unordered\_map<int,int>mp;

mp[0]=-1;

//sum of the elements of the subarray is a multiple of k

int sum=0;

for(int i=0;i<nums.size();i++){

sum+=nums[i];

if(mp.find(sum%k)!=mp.end()){

if(i-mp[sum%k]>=2)

return true;

}

else

mp[sum%k]=i;

}

return false;

}

};

[Previous](https://leetcode.com/problems/continuous-subarray-sum/solutions/5277027/java-beats-100-explained/?envType=daily-question&envId=2024-06-08)

[🔥 || Java beats 100% || Explained || 🔥](https://leetcode.com/problems/continuous-subarray-sum/solutions/5277027/java-beats-100-explained/?envType=daily-question&envId=2024-06-08)

[Next](https://leetcode.com/problems/continuous-subarray-sum/solutions/5277106/mod-k-prefix-sum-hash-table-from-mle-to-110ms-beats-99-40/?envType=daily-question&envId=2024-06-08)

[mod k+prefix sum+hash table||From MLE to 110ms Beats 99.40%](https://leetcode.com/problems/continuous-subarray-sum/solutions/5277106/mod-k-prefix-sum-hash-table-from-mle-to-110ms-beats-99-40/?envType=daily-question&envId=2024-06-08)

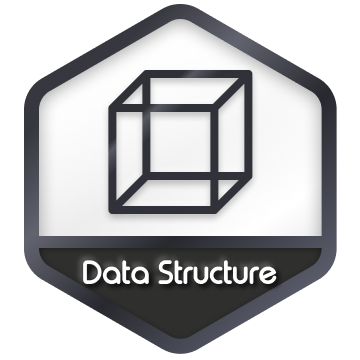
Comments (1)

Sort by:Best

PreviewComment

[](https://leetcode.com/u/jiayujerryLiu/)

[jiayujerryLiu](https://leetcode.com/u/jiayujerryLiu/)



an hour ago

bro nice job, was looking for this pattern, getting stuck on these like crazy

0

Reply

1

11

1

C++

Auto

2

3

4

5

6

7

8

9

10

11

1

using namespace std;

class Solution {

public:

    vector<vector<int>> dp;

    bool func(vector<int>& nums, int k, int i, int j, int sum) {

        if (i > j) {

            return false;

        }

#include <vector>

Saved

Ln 6, Col 25

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

nums =

[23,2,4,6,7]

k =

6

Output

true

Expected

true

Contribute a testcase

Tag