# Count Sequence

In some 1D array, sometimes we want to count the sequence with some specific condition. For such problem, we normally think as if we are doing a dynamic programming. We scan the array from left to right and stand at every position and think how many such sequences which shall satisfy the condition with the end element as current one.

## 446. Arithmetic Slices II - Subsequence

Hard

A sequence of numbers is called arithmetic if it consists of at least three elements and if the difference between any two consecutive elements is the same.

For example, these are arithmetic sequences:

1, 3, 5, 7, 9

7, 7, 7, 7

3, -1, -5, -9

The following sequence is not arithmetic.

1, 1, 2, 5, 7

A zero-indexed array A consisting of N numbers is given. A **subsequence** slice of that array is any sequence of integers (P0, P1, ..., Pk) such that 0 ≤ P0 < P1 < ... < Pk < N.

A **subsequence** slice (P0, P1, ..., Pk) of array A is called arithmetic if the sequence A[P0], A[P1], ..., A[Pk-1], A[Pk] is arithmetic. In particular, this means that k ≥ 2.

The function should return the number of arithmetic subsequence slices in the array A.

The input contains N integers. Every integer is in the range of -231 and 231-1 and 0 ≤ N ≤ 1000. The output is guaranteed to be less than 231-1.

**Example:**

**Input:** [2, 4, 6, 8, 10]

**Output:** 7

**Explanation:**

All arithmetic subsequence slices are:

[2,4,6]

[4,6,8]

[6,8,10]

[2,4,6,8]

[4,6,8,10]

[2,4,6,8,10]

[2,6,10]

### Analysis:

For every number, you pick every previous number to calculate the difference, and add the number of arithmetic numbers to it. The arithmetic numbers already count in the previous should be the number of arithmetic slices added to this number.

/// <summary>

/// Leet code #446. Arithmetic Slices II - Subsequence

///

/// A sequence of numbers is called arithmetic if it consists of at

/// least three elements and if the difference between any two

/// consecutive elements is the same.

///

/// For example, these are arithmetic sequences:

/// 1, 3, 5, 7, 9

/// 7, 7, 7, 7

/// 3, -1, -5, -9

///

/// The following sequence is not arithmetic.

/// 1, 1, 2, 5, 7

/// A zero-indexed array A consisting of N numbers is given. A

/// subsequence slice of that array is any sequence of integers

/// (P0, P1, ..., Pk) such that 0 ≤ P0 < P1 < ... < Pk < N.

///

/// A subsequence slice (P0, P1, ..., Pk) of array A is called

/// arithmetic if the sequence A[P0], A[P1], ..., A[Pk-1], A[Pk]

/// is arithmetic. In particular, this means that k ≥ 2.

///

/// The function should return the number of arithmetic subsequence

/// slices in the array A.

///

/// The input contains N integers. Every integer is in the range of -2^31

/// and 2^31-1 and 0 ≤ N ≤ 1000. The output is guaranteed to be less than

/// 2^31-1.

///

/// Example:

/// Input: [2, 4, 6, 8, 10]

/// Output: 7

///

/// Explanation:

/// All arithmetic subsequence slices are:

/// [2,4,6]

/// [4,6,8]

/// [6,8,10]

/// [2,4,6,8]

/// [4,6,8,10]

/// [2,4,6,8,10]

/// [2,6,10]

/// </summary>

int LeetCodeArray::numberOfArithmeticSlicesII(vector<int>& A)

{

int result = 0;

vector<unordered\_map<long long, int>> dp(A.size());

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

{

for (int j = 0; j < i; j++)

{

long long diff = (long long)A[i] - (long long)A[j];

if (dp[j].count(diff) > 0)

{

dp[i][diff] += dp[j][diff];

result += dp[j][diff];

}

dp[i][diff]++;

}

}

return result;

}

## 368. Largest Divisible Subset

Medium

Given a set of **distinct** positive integers, find the largest subset such that every pair (Si, Sj) of elements in this subset satisfies:

Si % Sj = 0 or Sj % Si = 0.

If there are multiple solutions, return any subset is fine.

**Example 1:**

**Input:** [1,2,3]

**Output:** [1,2] (of course, [1,3] will also be ok)

**Example 2:**

**Input:** [1,2,4,8]

**Output:** [1,2,4,8]

### Analysis:

Sort the array and build longest divisible chain.

/// <summary>

/// Leet code #368. Largest Divisible Subset

///

/// Given a set of distinct positive integers, find the largest

/// subset such that every pair (Si, Sj) of elements in this subset

/// satisfies: Si % Sj = 0 or Sj % Si = 0.

///

/// If there are multiple solutions, return any subset is fine.

///

/// Example 1:

/// nums: [1,2,3]

/// Result: [1,2] (of course, [1,3] will also be ok)

///

/// Example 2:

/// nums: [1,2,4,8]

/// Result: [1,2,4,8]

/// </summary>

vector<int> LeetCodeArray::largestDivisibleSubset(vector<int>& nums)

{

vector<int> result;

vector<vector<int>> set\_map(nums.size());

sort(nums.begin(), nums.end());

for (size\_t i = 0; i < nums.size(); i++)

{

for (size\_t j = 0; j < i; j++)

{

if ((nums[i] % nums[j]) == 0)

{

if (set\_map[i].size() < set\_map[j].size())

{

set\_map[i] = set\_map[j];

}

}

}

set\_map[i].push\_back(nums[i]);

if (set\_map[i].size() > result.size()) result = set\_map[i];

}

return result;

}