**[29. Divide Two Integers](https://leetcode.com/problems/divide-two-integers/)**

Solved

Medium

Topics

Companies

Given two integers dividend and divisor, divide two integers **without** using multiplication, division, and mod operator.

The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would be truncated to 8, and -2.7335 would be truncated to -2.

Return *the****quotient****after dividing*dividend*by*divisor.

**Note:**Assume we are dealing with an environment that could only store integers within the **32-bit** signed integer range: [−231, 231 − 1]. For this problem, if the quotient is **strictly greater than** 231 - 1, then return 231 - 1, and if the quotient is **strictly less than** -231, then return -231.

class Solution {

public:

    int divide(int dividend, int divisor) {

        if (dividend == INT\_MIN && divisor == -1) {

            return INT\_MAX;

        }

        int ans = dividend / divisor;

        return ans;

    }

};

[**228. Summary Ranges**](https://leetcode.com/problems/summary-ranges/)

Solved

Easy

Topics

Companies

You are given a **sorted unique** integer array nums.

A **range** [a,b] is the set of all integers from a to b (inclusive).

Return *the****smallest sorted****list of ranges that****cover all the numbers in the array exactly***. That is, each element of nums is covered by exactly one of the ranges, and there is no integer x such that x is in one of the ranges but not in nums.

Each range [a,b] in the list should be output as:

Ans :

#include <vector>

#include <string>

using namespace std;

class Solution {

public:

    vector<string> summaryRanges(vector<int>& nums) {

        vector<string> result;

        if (nums.empty()) return result;

        int start = nums[0];

        for (int i = 1; i <= nums.size(); ++i) {

            if (i == nums.size() || nums[i] != nums[i - 1] + 1) {

                if (start == nums[i - 1]) {

                    result.push\_back(to\_string(start));

                } else {

                    result.push\_back(to\_string(start) + "->" + to\_string(nums[i - 1]));

                }

                if (i < nums.size()) start = nums[i];

            }

        }

        return result;

    }

};

Code

Testcase

Testcase

Test Result

[**2827. Number of Beautiful Integers in the Range**](https://leetcode.com/problems/number-of-beautiful-integers-in-the-range/)

Solved

Hard

Topics

Companies

Hint

You are given positive integers low, high, and k.

A number is **beautiful** if it meets both of the following conditions:

* The count of even digits in the number is equal to the count of odd digits.
* The number is divisible by k.

Return *the number of beautiful integers in the range* [low, high]

Ans :

class Solution {

public:

    int dp[12][12][12][2][22][2];

    int G(string& number, int k, int pos=0,int even=0, int odd=0, int tight = 1, int summ=0, bool zero = true){

        if (pos >= number.length()){

            return (even==odd && !zero && summ == 0);

        }

        if (dp[pos][even][odd][tight][summ][zero] != -1)

            return dp[pos][even][odd][tight][summ][zero];

        int res = 0;

        for(int digit = 0; digit <= (tight ? number[pos]-'0' : 9); digit++){

            int newTight = (tight && digit == number[pos]-'0');

            int newEven = even + (digit%2==0 && (digit>0 || !zero));

            int newOdd = odd + (digit&1);

            res += G(number, k, pos+1, newEven, newOdd, newTight, (summ\*10+digit)%k, (zero && digit == 0));

        }

        dp[pos][even][odd][tight][summ][zero] = res;

        return res;

    }

    int numberOfBeautifulIntegers(int low, int high, int k) {

        string lows = to\_string(low-1);

        string highs = to\_string(high);

        memset(dp,-1,sizeof(dp));

        int upper = G(highs,k);

        memset(dp,-1,sizeof(dp));

        int lower = G(lows,k);

        return upper - lower;

    }

};

[**2156. Find Substring With Given Hash Value**](https://leetcode.com/problems/find-substring-with-given-hash-value/)

Solved

Hard

Topics

Companies

Hint

The hash of a **0-indexed** string s of length k, given integers p and m, is computed using the following function:

* hash(s, p, m) = (val(s[0]) \* p0 + val(s[1]) \* p1 + ... + val(s[k-1]) \* pk-1) mod m.

Where val(s[i]) represents the index of s[i] in the alphabet from val('a') = 1 to val('z') = 26.

You are given a string s and the integers power, modulo, k, and hashValue. Return sub,*the****first******substring****of*s*of length*k*such that*hash(sub, power, modulo) == hashValue.

The test cases will be generated such that an answer always **exists**.

A **substring** is a contiguous non-empty sequence of characters within a string

Ans :

class Solution {

public:

    string subStrHash(string s, int power, int modulo, int k, int hashValue) {

        long long cur = 0, res = 0, pk = 1, n = s.size();

        // Iterate from the end of the string

        for (int i = n - 1; i >= 0; --i) {

            cur = (cur \* power + (s[i] - 'a' + 1)) % modulo;

            if (i + k >= n) {

                // Update pk only for the first k characters

                pk = pk \* power % modulo;

            } else {

                // Adjust the current hash when a character goes out of the window

                cur = (cur - ((s[i + k] - 'a' + 1) \* pk % modulo) + modulo) % modulo;

            }

            // If the hash matches the target value, update the result

            if (cur == hashValue) {

                res = i;

            }

        }

        return s.substr(res, k);

    }

};

[**91. Decode Ways**](https://leetcode.com/problems/decode-ways/)

Solved

Medium

Topics

Companies

You have intercepted a secret message encoded as a string of numbers. The message is **decoded** via the following mapping:

"1" -> 'A'  
"2" -> 'B'  
...  
"25" -> 'Y'  
"26" -> 'Z'

However, while decoding the message, you realize that there are many different ways you can decode the message because some codes are contained in other codes ("2" and "5" vs "25").

For example, "11106" can be decoded into:

* "AAJF" with the grouping (1, 1, 10, 6)
* "KJF" with the grouping (11, 10, 6)
* The grouping (1, 11, 06) is invalid because "06" is not a valid code (only "6" is valid).

Note: there may be strings that are impossible to decode.  
  
Given a string s containing only digits, return the **number of ways** to **decode** it. If the entire string cannot be decoded in any valid way, return 0.

The test cases are generated so that the answer fits in a **32-bit** integer.

Ans :

class Solution {

public:

    int numDecodings(string s) {

        if (s.empty() || s[0] == '0') return 0;

        int n = s.size();

        vector<int> dp(n + 1, 0);

        dp[0] = 1;

        dp[1] = 1;

        for (int i = 2; i <= n; ++i) {

            if (s[i - 1] != '0') dp[i] += dp[i - 1];

            if (s[i - 2] == '1' || (s[i - 2] == '2' && s[i - 1] <= '6')) dp[i] += dp[i - 2];

        }

        return dp[n];

    }

};