5) Fibonacci no

10 November 2024 23:28

The **Fibonacci numbers**, commonly denoted F(n) form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from 0 and 1

From <https://leetcode.com/problems/fibonacci-number/description/>

```
int fib(int n) {
    if(n==0 || n == 1) return n;
    return fib(n-1) + fib(n-2);
}
```

6) Climbing stairs (same as fibonacci)

11 November 2024 00:45

You are climbing a staircase. It takes n steps to reach the top.
Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

From <https://leetcode.com/problems/climbing-stairs/description/>

```
int climbStairs(int n) {
    if (n == 0 || n == 1) return 1;
    return climbStairs(n-1) + climbStairs(n-2);
}
```

From <https://leetcode.com/problems/climbing-stairs/submissions/1120976202/>

9) Subsequences of String

12 November 2024 13:32

You are given a string 'STR' containing lowercase English letters from a to z inclusive. Your task is to find all non-empty possible subsequences of 'STR'.
A Subsequence of a string is the one which is generated by deleting 0 or more letters from the string and keeping the rest of the letters in the same order.

From <https://www.naukri.com/code360/problems/subsequences-of-string_985087?count=25&page=1&search=&sort_entity=order&sort_order=ASC&leftPanelTabValue=PROBLEM>

```
void helper(const string& str, string& temp, vector<string>& ans, int index) {
    if (index >= str.length()) {
        if (!temp.empty()) {
            ans.push_back(temp);
        }
        return;
    }

    // Exclude the current character
    helper(str, temp, ans, index + 1);

    // Include the current character
    temp.push_back(str[index]);
    helper(str, temp, ans, index + 1);
    temp.pop_back();  // Backtrack
}
vector<string> subsequences(const string& str) {
    vector<string> ans;
    string temp;
    helper(str, temp, ans, 0);
    return ans;
}
```

15) Add strings

12 November 2024 14:30

Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string.

From <https://leetcode.com/problems/add-strings/description/>

```
    void helper(string &num1, int p1, string &num2, int p2, string &ans, int carry =
0){
        if(p1 < 0 && p2 < 0){
            if(carry != 0)
                ans.push_back(carry + '0');

            return;
        }
        int n1 = (p1 >= 0? num1[p1]:'0') - '0';
        int n2 = (p2 >= 0? num2[p2]:'0') - '0';
        int sum = n1 + n2 + carry;
        int digit = sum % 10;
        carry = sum/10;
        ans.push_back(digit + '0');
        helper(num1, p1-1, num2, p2-1, ans, carry);
    }
    string addStrings(string num1, string num2) {
        int p1 = num1.size()-1;
        int p2 = num2.size()-1;
        string ans = "";
        helper(num1, p1, num2, p2, ans);
        reverse(ans.begin(), ans.end());
        return ans;
    }
```

20) Integer to English words

16 November 2024 16:06

Convert a non-negative integer num to its English words representation.

From <https://leetcode.com/problems/integer-to-english-words/description/>

```
vector<pair<int, string>> mp =
{{1000000000, "Billion"},
 {1000000, "Million"},
 {1000, "Thousand"},
 {100, "Hundred"},
 {90, "Ninety"},
 {80, "Eighty"},
 {70, "Seventy"},
 {60, "Sixty"},
 {50, "Fifty"},
 {40, "Forty"},
 {30, "Thirty"},
 {20, "Twenty"},
 {19, "Nineteen"},
 {18, "Eighteen"},
 {17, "Seventeen"},
 {16, "Sixteen"},
 {15, "Fifteen"},
 {14, "Fourteen"},
 {13, "Thirteen"},
 {12, "Twelve"},
 {11, "Eleven"},
 {10, "Ten"},
 {9, "Nine"},
 {8, "Eight"},
 {7, "Seven"},
 {6, "Six"},
 {5, "Five"},
 {4, "Four"},
 {3, "Three"},
 {2, "Two"},
 {1, "One"}};
string numberToWords(int num) {
    if(num == 0) return "Zero";
    for(auto it:mp){
        if(num >= it.first){
            string a = "";
            if(num >= 100){
                a = numberToWords(num / it.first) + " ";
            }
            string b = it.second;
            string c = "";
            if(num % it.first != 0){
                c = " " + numberToWords(num % it.first);
            }
            return a+b+c;
        }
    }
    return "";
}
```

From <https://leetcode.com/problems/integer-to-english-words/submissions/1125301209/>

26) Count Disarrangement (DP se krna is optimal)

17 November 2024 01:28

You are given **n** balls numbered from 1 to **n** and there are **n** baskets numbered from 1 to **n** in front of you. The **i**th basket is meant for the **i**th ball. Calculate the number of ways in which **no** ball goes into its respective basket.

From <https://www.geeksforgeeks.org/problems/dearrangement-of-balls0918/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=practice_card>

```
int countDer(int n) {
    if(n == 1 || n == 2) return n-1;

    return (n-1)*(countDer(n-1) + countDer(n-2));
}
```

30) Generate all binary string

22 November 2024 15:16

Given an integer **N** , Print all binary strings of size N which do not contain consecutive 1s. A binary string is that string which contains only 0 and 1.

From <https://www.geeksforgeeks.org/problems/generate-all-binary-strings/0>

```
void helper(int N, string prefix, char lastChar, vector<string> &ans) {
    // Base case: If the current string's length equals N, print it
    if (prefix.length() == N) {
        ans.push_back(prefix);
        return;
    }

    // Append '0' to the string and recurse
    helper(N, prefix + "0", '0', ans);

    // Append '1' only if the last character was not '1'
    if (lastChar != '1') {
        helper(N, prefix + "1", '1', ans);
    }
}

vector<string> generateBinaryStrings(int num){
    vector<string>ans;
    helper(num, "", '0', ans);
    return ans;
}
```

31) Pow(x, n)

23 November 2024 02:13

Implement pow(x, n), which calculates x raised to the power n (i.e., x^n).

From <https://leetcode.com/problems/powx-n/description/>

```
double solve(double x, long n) {
    if(n == 0) return 1;
    if(n < 0) return 1/solve(x, -n);
    if(n%2 == 0) return solve(x*x, n/2);
    return x*solve(x*x, (n-1)/2);
}
double myPow(double x, int n) {
    return solve(x, (long)n);
}
```

32) Count good numbers

23 November 2024 03:01

A digit string is **good** if the digits **(0-indexed)** at **even** indices are **even** and the digits at **odd** indices are **prime** (2, 3, 5, or 7).

- For example, "2582" is good because the digits (2 and 8) at even positions are even and the digits (5 and 2) at odd positions are prime. However, "3245" is **not** good because 3 is at an even index but is not even.

Given an integer n, return *the **total** number of good digit strings of length* n. Since the answer may be large, **return it modulo** $10^9 + 7$.
A **digit string** is a string consisting of digits 0 through 9 that may contain leading zeros.

From <https://leetcode.com/problems/count-good-numbers/description/>

```
int MOD = 1e9+7;
// Function to compute (x^n) % MOD
long pow(long x, long n) {
    x = x%MOD;
    if (n == 0) return 1; // Base case: x^0 = 1
    if(n%2 == 0) return pow(x*x, n/2)%MOD;
    return x*pow(x*x, (n-1)/2)%MOD;
}
int countGoodNumbers(long n) {
    long even = (n + 1) / 2;         // Count of even indices
    long odd = n / 2;               // Count of odd indices
    long first = pow(5, even) % MOD; // Power of 5 for even places
    long second = pow(4, odd) % MOD; // Power of 4 for odd places
    return (int) ((first * second) % MOD); // Final result modulo MOD
}
```

33) Permutation Sequence

23 November 2024 18:24

The set [1, 2, 3, ..., n] contains a total of n! unique permutations.
By listing and labeling all of the permutations in order, we get the following sequence for n = 3:

1. "123"
2. "132"
3. "213"
4. "231"
5. "312"
6. "321"

Given n and k, return the kth permutation sequence.

From <https://leetcode.com/problems/permutation-sequence/description/>

```
string getPermutation(int n, int k) {
    int fact = 1;
    vector<int> numbers;
    for(int i = 1;i<n;i++) {
        fact = fact * i;
        numbers.push_back(i);
    }
    numbers.push_back(n);
    string ans = "";
    k = k - 1;
    while(true) {
        ans = ans + to_string(numbers[k / fact]);
        numbers.erase(numbers.begin() + k / fact);
        if(numbers.size() == 0) {
            break;
        }

        k = k % fact;
        fact = fact / numbers.size();
    }
    return ans;
}
```

34) Generate binary strings wo adjacent zeroes (based on problem 30)

23 November 2024 19:15

You are given a positive integer n.
A binary string x is **valid** if all
substrings
of x of length 2 contain **at least** one "1".
Return all **valid** strings with length n, in *any* order.

From <https://leetcode.com/problems/generate-binary-strings-without-adjacent-zeros/description/>

```
void helper(int n, vector<string>& ans, string temp, char ch){
    if(temp.length() == n){
        ans.push_back(temp);
        return;
    }
    helper(n, ans, temp + "1", '1');
    if(ch != '0') helper(n, ans, temp + "0", '0');
}
vector<string> validStrings(int n) {
    vector<string> ans;
    helper(n, ans, "", '1');
    return ans;
}
```