# NAME- Abir Chakraborty| UID- 22BCS14321 | SECTION- 601/A 1 Climbing Stairs

class Solution {

public:

int climbStairs(int n) {

if (n <= 2) return n;

vector<int> dp(n + 1, 0);

dp[1] = 1;

dp[2] = 2;

for (int i = 3; i <= n; i++) {

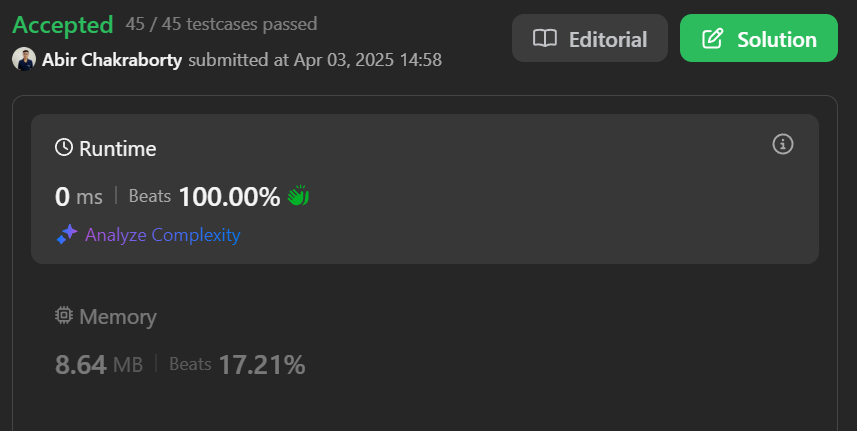
dp[i] = dp[i - 1] + dp[i - 2];

}

return dp[n];

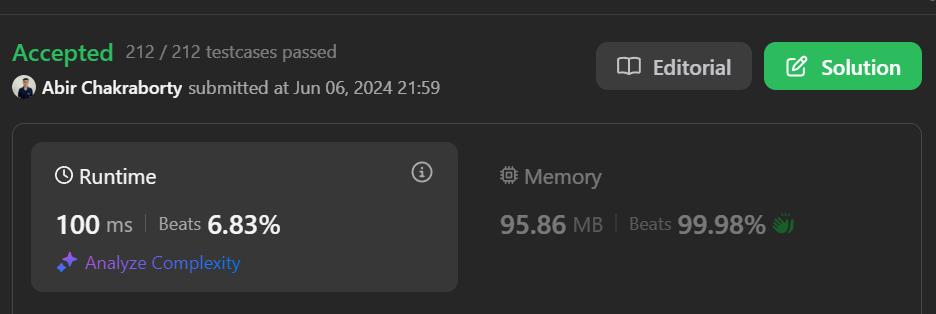
}

};



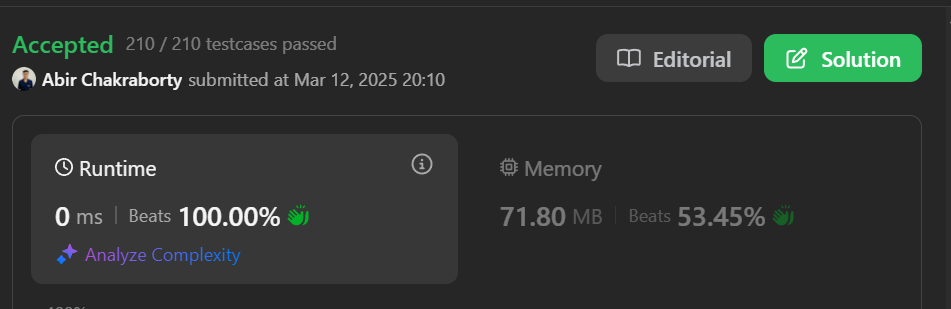
# Best Time to Buy and Sell a Stock

class Solution {public: int maxProfit(vector<int>& arr) { int maxPro = 0; int n = arr.size(); int minPrice = INT\_MAX; for (int i = 0; i < arr.size(); i++) { minPrice = min(minPrice, arr[i]); maxPro = max(maxPro, arr[i] - minPrice); } return maxPro; }};



# Maximum Subarray

class Solution {public: int maxSubArray(vector<int>& arr) { int n = arr.size(); long long maxi = LONG\_MIN; long long sum = 0; for (int i = 0; i < n; i++) { sum += arr[i]; if (sum > maxi) { maxi = sum; } if (sum < 0) { sum = 0; } } return maxi; }};



# House Robber

# class Solution {

# public:

# int rob(vector<int>& nums) {

# int n = nums.size();

# if (n == 1) {

# return nums[0];

# }

# vector<int> dp(n, 0);

# dp[0] = nums[0];

# dp[1] = max(nums[0], nums[1]);

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

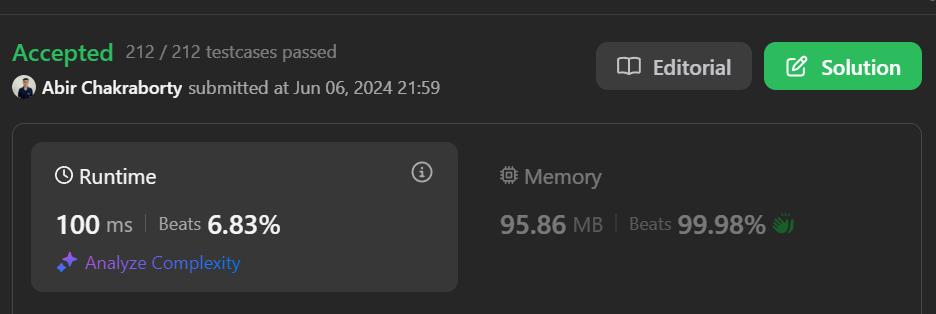
# dp[i] = max(dp[i - 1], nums[i] + dp[i - 2]);

# }

# return dp[n - 1];

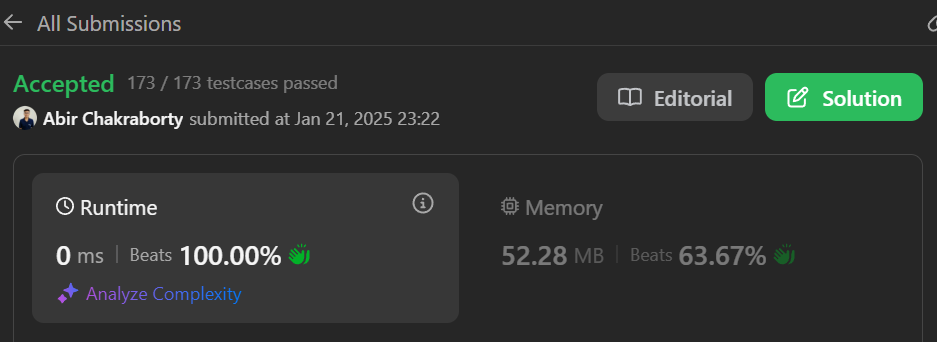
# }

};



# Jump Game

class Solution {public: bool canJump(vector<int>& nums) { int maxIndex=0; for(int i=0;i<nums.size();i++){ if(i>maxIndex){ return false; } maxIndex=max(maxIndex,i+nums[i]); } return true; }};



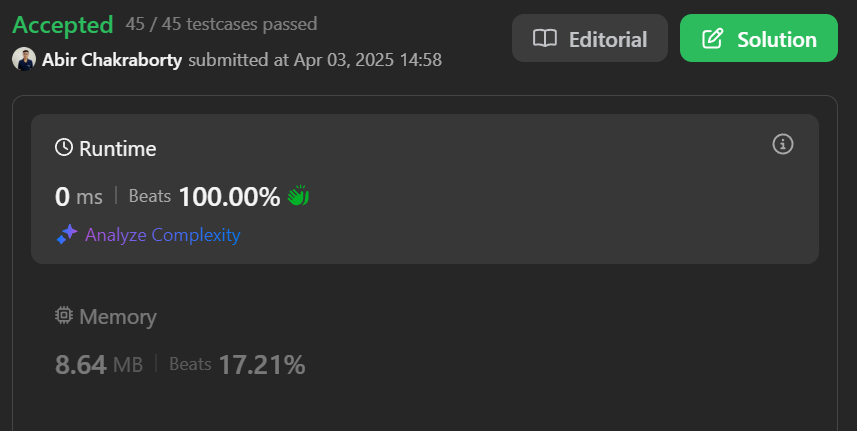
## Unique Paths

class Solution {public: int uniquePaths(int m, int n) { std::vector<int> aboveRow(n, 1); for (int row = 1; row < m; row++) { std::vector<int> currentRow(n, 1); for (int col = 1; col < n; col++) { currentRow[col] = currentRow[col - 1] + aboveRow[col]; } aboveRow = currentRow; } return aboveRow[n - 1]; }};



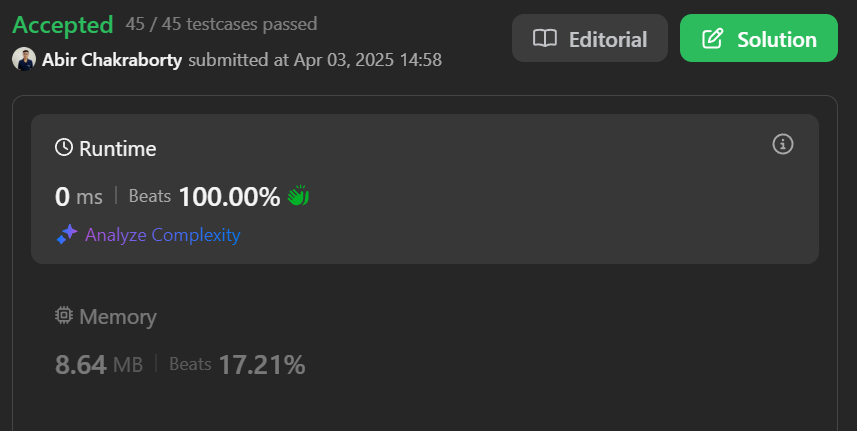
## Coin Change

class Solution {public: int coinChange(vector<int>& coins, int amount) { vector<int> minCoins(amount + 1, amount + 1); minCoins[0] = 0; for (int i = 1; i <= amount; i++) { for (int j = 0; j < coins.size(); j++) { if (i - coins[j] >= 0) { minCoins[i] = min(minCoins[i], 1 + minCoins[i - coins[j]]); } } } return minCoins[amount] != amount + 1 ? minCoins[amount] : -1; }};



1. **Longest Increasing Subsequence**

class Solution { // 256 ms, faster than 42.84%public: int lengthOfLIS(vector<int>& nums) { int n = nums.size(); vector<int> dp(n, 1); for (int i = 0; i < n; ++i) for (int j = 0; j < i; ++j) if (nums[i] > nums[j] && dp[i] < dp[j] + 1) dp[i] = dp[j] + 1; return \*max\_element(dp.begin(), dp.end()); }};



1. **Maximum Product Subarray**

class Solution {

public:

int maxProduct(vector<int>& nums) {

int n = nums.size(); //size of array.

double pre = 1, suff = 1;

double ans = INT\_MIN;

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

if (pre == 0) pre = 1;

if (suff == 0) suff = 1;

pre \*= nums[i];

suff \*= nums[n - i - 1];

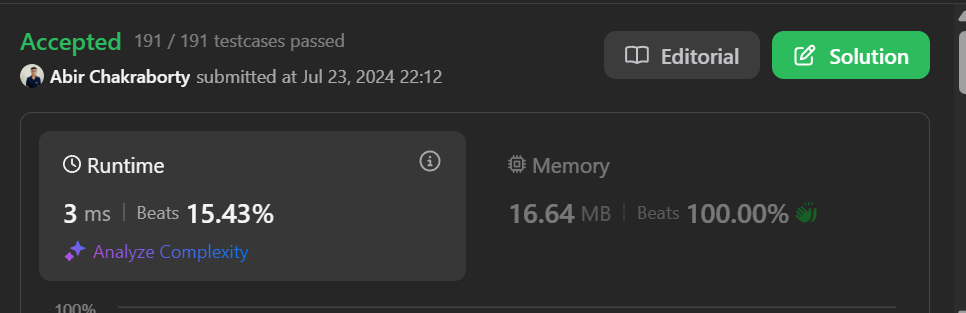
ans = max(ans, max(pre, suff));

}

return ans;

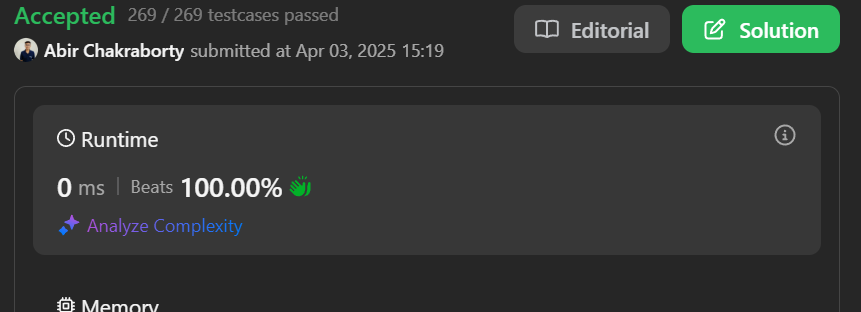
}

};



1. **Decode Ways**

class Solution {public: int numDecodings(std::string s) { if (s.empty() || s[0] == '0') { return 0; } int n = s.length(); std::vector<int> dp(n + 1, 0); dp[0] = 1; dp[1] = 1; for (int i = 2; i <= n; ++i) { int oneDigit = s[i - 1] - '0'; int twoDigits = std::stoi(s.substr(i - 2, 2)); if (oneDigit != 0) { dp[i] += dp[i - 1]; } if (10 <= twoDigits && twoDigits <= 26) { dp[i] += dp[i - 2]; } } return dp[n]; }};



1. **Best Time to Buy and Sell a Stock with Cooldown**

class Solution {public: int maxProfit(vector<int>& prices) { int buy(INT\_MIN), sell(0), prev\_sell(0), prev\_buy; for (int price : prices) { prev\_buy = buy; buy = max(prev\_sell - price, buy); prev\_sell = sell; sell = max(prev\_buy + price, sell); } return sell; }};

