# 1) fibonacci number

02 December 2024 19:32

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. That is,
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
Given n, calculate F(n).

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

//memoiation:

```
int helper(int n, vector<int>&dp){
    if(n == 0 || n==1) return n;
    if(dp[n] != -1) return dp[n];
    return dp[n] = helper(n-1, dp) + helper(n-2, dp);
}
int fib(int n) {
    vector<int> dp(n+1, -1);
    return helper(n, dp);
}
```

//tabulation

```
int fib(int n) {
    if(n <= 1) return n;
    vector<int> dp(n+1, -1);
    dp[0] = 0;
    dp[1] = 1;
    for(int i = 2; i <= n; i++){
        dp[i] = dp[i-1] + dp[i-2];
    }
    return dp[n];
}
```

//space optimization

```
int fib(int n) {
    if(n <= 1) return n;
    int prev2 = 0;
    int prev = 1;
    for(int i = 2; i <= n; i++){
        int curr = prev + prev2;
        prev2 = prev;
        prev = curr;
    }
    return prev;
}
```

# 2) Climbing stairs

02 December 2024 19:54

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 <= 1) return n;
    int prev2 = 1;
    int prev = 1;
    for(int i = 2; i <= n; i++){
        int curr = prev + prev2;
        prev2 = prev;
        prev = curr;
    }
    return prev;
}
```

From <https://leetcode.com/problems/climbing-stairs/submissions/1468383397/>

# 3) Geek Jump

03 December 2024 01:54

Geek wants to climb from the 0th stair to the (n-1)th stair. At a time the Geek can climb either one or two steps. A height[N] array is also given. Whenever the geek jumps from stair i to stair j, the energy consumed in the jump is abs(height[i]- height[j]), where abs() means the absolute difference. return the minimum energy that can be used by the Geek to jump from stair 0 to stair N-1.

From <https://www.geeksforgeeks.org/problems/geek-jump/1?utm_source=youtube&utm_medium=collab_striver_ytdescription&utm_campaign=geek-jump>

//memoization

```
int helper(vector<int>& height, vector<int> &dp, int n){
    if(n == 0)return 0;
    if(dp[n] != -1) return dp[n];
    int jump2 = INT_MAX;
    int jump1 = helper(height, dp, n-1) + abs(height[n] - height[n-1]);
    if(n > 1) jump2 = helper(height, dp, n-2) + abs(height[n] - height[n-2]);
    return dp[n] = min(jump1, jump2);
}
int minCost(vector<int>& height) {
    int n = height.size();
    vector<int> dp(n, -1);
    return helper(height, dp, n-1);
}
```

//tabulation

```
int minimumEnergy(vector<int>& height, int n) {
    vector<int>dp(n, -1);
    dp[0] = 0;

    for(int i = 1; i < n; i++){
        int jump1 = dp[i-1] + abs(height[i] - height[i-1]);
        int jump2 = INT_MAX;
        if(i > 1) jump2 = dp[i-2] + abs(height[i] - height[i-2]);

        dp[i] = min(jump1, jump2);
    }
    return dp[n-1];
}
```

//space

```
int minimumEnergy(vector<int>& height, int n) {
    int prev = 0;
    int prev2 = 0;

    for(int i = 1; i < n; i++){
```

```
        int jump1 = prev + abs(height[i] - height[i-1]);
        int jump2 = INT_MAX;
        if(i > 1) jump2 = prev2 + abs(height[i] - height[i-2]);

        int curr = min(jump1, jump2);
        prev2 = prev;
        prev = curr;
    }
    return prev;
}
```

# 4) Geek Jump 2/ Minimal Cost

03 December 2024 02:52

There is an array **arr** of heights of stone and Geek is standing at the first stone and can jump to one of the following: Stone i+1, i+2, ... i+k stone, where k is the maximum number of steps that can be jumped and cost will be $|h_i-h_j|$ is incurred, where j is the stone to land on. Find the minimum possible total cost incurred before the Geek reaches the last stone.

From <https://www.geeksforgeeks.org/problems/minimal-cost/1?utm_source=youtube&utm_medium=collab_striver_ytdescription&utm_campaign=minimal-cost>

//recursive solution

```
int helper(int k, vector<int>&arr, int n){
    if(n == 0) return 0;
    int mini = INT_MAX;

    for(int i = 1; i <= k; i++){
        int jump = INT_MAX;
        if(n-i >= 0) jump = helper(k, arr, n-i) + abs(arr[n] - arr[n-i]);
        mini = min(mini, jump);
    }
    return mini;
}
int minimizeCost(int k, vector<int>& arr) {
    int n = arr.size();
    return helper(k, arr, n-1);
}
```

//memoization

```
int helper(int k, vector<int>&arr, int n ,vector<int> &dp){
    if(n == 0) return 0;
    if(dp[n] != -1) return dp[n];
    int mini = INT_MAX;

    for(int i = 1; i <= k; i++){
        int jump = INT_MAX;
        if(n-i >= 0) jump = helper(k, arr, n-i, dp) + abs(arr[n] - arr[n-i]);
        mini = min(mini, jump);
    }
    return dp[n] = mini;
}
int minimizeCost(int k, vector<int>& arr) {
    int n = arr.size();
    vector<int> dp(n, -1);
    return helper(k, arr, n-1, dp);
}
```

//tabulation

```
int helper(int k, vector<int>&arr, int n ,vector<int> &dp){
```

```
    dp[0] = 0;

    for(int i = 1; i < n; i++){
        int mini = INT_MAX;
        for(int j = 1; j <= k; j++){
            int jump = INT_MAX;
            if(i-j >= 0) jump = dp[i-j] + abs(arr[i] - arr[i-j]);
            mini = min(mini, jump);
        }
        dp[i] = mini;
    }
    return dp[n-1];
}
int minimizeCost(int k, vector<int>& arr) {
    int n = arr.size();
    vector<int> dp(n, -1);
    return helper(k, arr, n, dp);
}
```

# 5) House Robber

04 December 2024 16:30

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.
Given an integer array nums representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

From <https://leetcode.com/problems/house-robber/description/>

//memoization

```
int helper(vector<int>& nums, int n, vector<int>& dp){
    if(n < 0) return 0;
    if(dp[n] != -1)return dp[n];
    int option1 = nums[n] + helper(nums, n-2, dp);
    int option2 = 0 + helper(nums, n-1, dp);
    return dp[n] = max(option1, option2);
}
int rob(vector<int>& nums) {
    int n = nums.size();
    vector<int>dp(n, -1);
    return helper(nums, n-1, dp);
}
```

From <https://leetcode.com/problems/house-robber/submissions/1470053083/>

//tabulation

```
int solve(vector<int>& arr, int n, vector<int>& dp) {
   dp[0] = arr[0];

   for (int i = 1; i < n; i++) {

      int pick = arr[i];
      if (i > 1)
         pick += dp[i - 2];
      int nonPick = dp[i - 1];
      dp[i] = max(pick, nonPick);
   }
   return dp[n - 1];
}
   int rob(vector<int>& nums){
      int n = nums.size();
      int index = 0;
      vector<int> dp(n, -1);
      return solve(nums, n, dp);
   }
```

From <https://leetcode.com/problems/house-robber/submissions/1470075130/>

//space optimization

```
int solve(vector<int>& arr, int n) {
        int prev = arr[0];
        int prev2 = 0;

        for (int i = 1; i < n; i++) {
            int pick = arr[i] + prev2;
            int nonPick = prev;
            int curr = max(pick, nonPick);
            prev2 = prev;
            prev = curr;
        }
        return prev;
    }
    int rob(vector<int>& nums){
       int n = nums.size();
       return solve(nums, n);
    }
```

From <https://leetcode.com/problems/house-robber/submissions/1470077827/>

# 6) House Robber II

04 December 2024 16:33

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are **arranged in a circle.** That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have a security system connected, and **it will automatically contact the police if two adjacent houses were broken into on the same night**.
Given an integer array nums representing the amount of money of each house, return *the maximum amount of money you can rob tonight **without alerting the police***.

From <https://leetcode.com/problems/house-robber-ii/description/>

```
int helper(vector<int>& nums, int n, vector<int>& dp){
    if(n < 0) return 0;
    if(dp[n] != -1)return dp[n];
    int option1 = nums[n] + helper(nums, n-2, dp);
    int option2 = 0 + helper(nums, n-1, dp);
    return dp[n] = max(option1, option2);
}
int rob(vector<int>& nums) {
    int n = nums.size();
    if(n == 1) return nums[0];
    vector<int> temp1,temp2;
    vector<int>dp1(n-1, -1);
    vector<int>dp2(n-1, -1);
    for(int i = 1; i < n; i++){
        temp1.push_back(nums[i]);
    }
    for(int i = 0; i < n-1; i++){
        temp2.push_back(nums[i]);
    }
    int option1 = helper(temp1, n-2, dp1);
    int option2 = helper(temp2, n-2, dp2);
    return max(option1, option2);
}
```