【第二十五周】递推算法及解题套路

1、70. 爬楼梯

1、我们用 f(i) 表示爬到第 i 级台阶的方案数，考虑最后一步可能跨了一级台阶，也可能跨了两级台阶，所以我们可以列出如下式子：f(i) = f(i - 1) + f(i - 2)

2、它意味着爬到第 x 级台阶的方案数是爬到第 i−1 级台阶的方案数和爬到第 i−2 级台阶的方案数的和。很好理解，因为每次只能爬 1 级或 2 级，所以 f(i) 只能从 f(i−1) 和 f(i−2) 转移过来，而这里要统计方案总数，我们就需要对这两项的贡献求和

```
 /**
 * @param {number} n
 * @return {number}
 */
var climbStairs = function(n) {
    let f = new Array(n + 1).fill(0);
    f[0] = 1,f[1] = 1;
    for(let i = 2; i <= n;i++) f[i] = f[i - 1] + f[ i - 2];
    return f[n];
};
```

2、746. 使用最小花费爬楼梯

1、假设数组 cost 的长度为 n，则 n 个阶梯分别对应下标 0 到 n−1，楼层顶部对应下标 n，问题等价于计算达到下标 n 的最小花费。可以通过动态规划求解

2、创建长度为 n+1 的数组 dp，其中 dp[i] 表示达到下标 i 的最小花费

3、由于可以选择下标 0 或 1 作为初始阶梯，因此有 dp[0]=dp[1]=0。

4、当 2≤i≤n 时，可以从下标 i−1 使用 cost[i−1] 的花费达到下标 i，或者从下标 i−2 使用 cost[i−2] 的花费达到下标 i。为了使总花费最小，dp[i] 应取上述两项的最小值，因此状态转移方程如下：dp[i]=min(dp[i−1]+cost[i−1],dp[i−2]+cost[i−2])

5、依次计算 dp 中的每一项的值，最终得到的 dp[n] 即为达到楼层顶部的最小花费。

```
 /**
 * @param {number[]} cost
 * @return {number}
 */
var minCostClimbingStairs = function(cost) {
    let n = cost.length;
    let dp = new Array(n + 1).fill(0);
    // 根据题意添加元素0
    cost.push(0);
    // 初始华边界条件
    dp[0] = cost[0],dp[1] = cost[1];
```

```
    for(let i = 2;i <= n;i++) dp[i] = Math.min(dp[i - 1],dp[i - 2]) + cost[i];
    return dp[n];
};
```

3、[120. 三角形最小路径和]

1、把大问题拆分为子问题，它们的区别在于问题的规模。

2、规模在这里是：层高。

3、base case 是当矩阵行高只有 1 时，它的最优路径是显而易见的。

4、从顶考察，每个点两个选择，2其实是不知道选3还是4.5、从底部考察，6、5、7都很清楚选择哪个点。6 + min(4,1) = 6+1 =7;5 + min(1,8) = 5 + 1 = 6;7 + min(8,3) = 7 +3=10;

5、有了一层高的「最优路径」，我们推出两层高的「最优路径」；有了两层高的「最优路径」，我们推出三层高的「最优路径。

6、......

7、triangle.length-1层高的「最优路径」推出 triangle.length 层高的「最优路径」

```
  /**
  * @param {number[][]} triangle
  * @return {number}
  */
 var minimumTotal = (triangle) => {
   const n = triangle.length;
   // 初始化dp数组
   const dp = new Array(n);
   for (let i = 0; i < n; i++) {
     dp[i] = new Array(triangle[i].length);
   }

   for (let i = n - 1; i >= 0; i--) { // 自底而上遍历
     for (let j = 0; j < triangle[i].length; j++) { // 同一层的
       if (i == n - 1) {  // base case 最底层
         dp[i][j] = triangle[i][j];
       } else { // 状态转移方程，上一层由它下面一层计算出
         dp[i][j] = Math.min(dp[i + 1][j], dp[i + 1][j + 1]) + triangle[i][j];
       }
     }
   }
   return dp[0][0];
 };
```

4、[300. 最长递增子序列]

1、DP数组用来存储该位置的最长子序列长度。

2、设nums[j] < nums[i], dp[i] = Math.max(dp[i], dp[j] + 1),即上一个比它小的数的最长子序列加1

3、遍历dp数组，找出最大值。

```
/**
 * @param {number[]} nums
 * @return {number}
 */

var lengthOfLIS = function(nums) {
    let dp = new Array(nums.length);
    let ans = 0;
    for(let i = 0; i < nums.length;i++){
        dp[i] = 1;
        for(let j = 0; j < i;j++){
            if(nums[j] >= nums[i]) continue;
            dp[i] = Math.max(dp[i],dp[j] + 1);
        }
        ans = Math.max(dp[i],ans);
    }
    return ans;
};
```

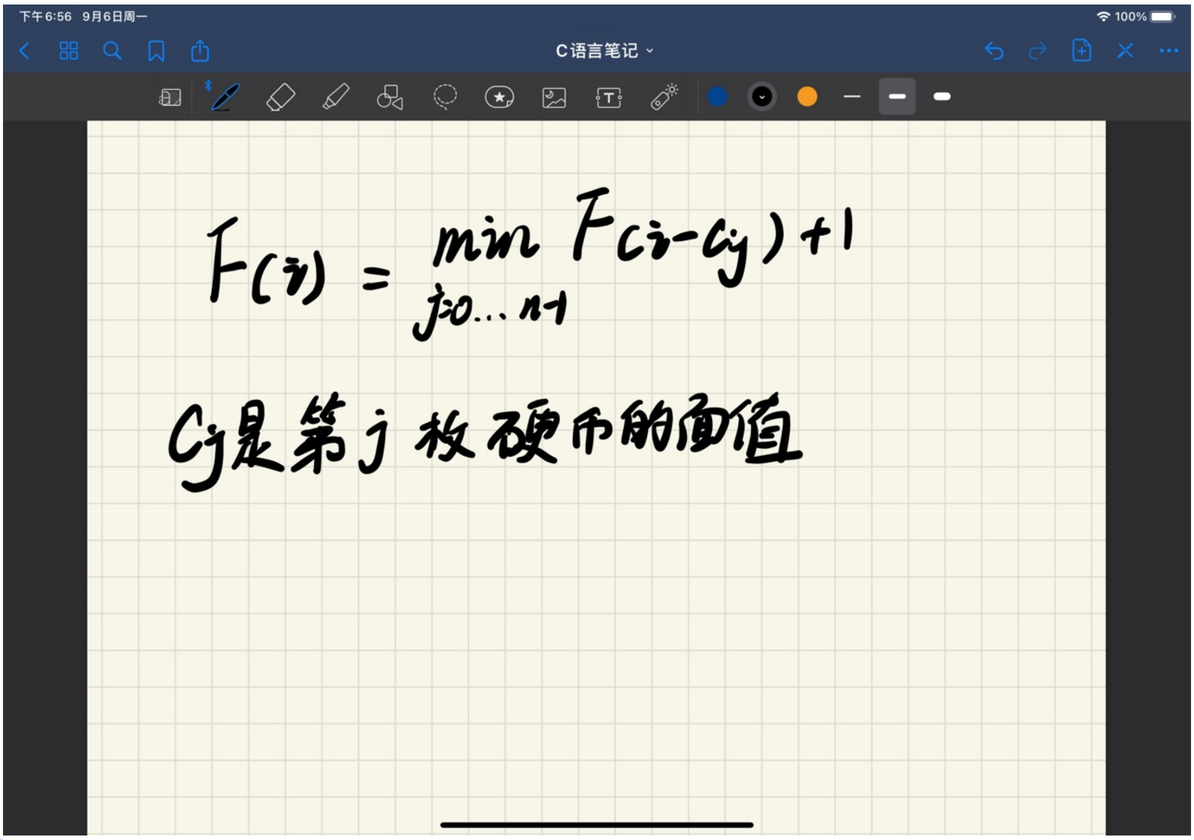
5、53. 最大子序和

用DP求解，因为这道题求的是最大和的连续子数组, 所以DP[i]取决于DP[i - 1] + nums[i] > nums[i],如果nums[i]大，则从nums[i]开始单独成一段，否则与之前的连成一段。所以可得方程dp[i] = Math.max(dp[i - 1] + nums[i])

```
 /**
 * @param {number[]} nums
 * @return {number}
 */
var maxSubArray = function(nums) {
    for(let i = 1; i < nums.length;i++) nums[i] += nums[i - 1];
    let pre = 0, ans = -Infinity;
    for(const x of nums){
        ans = Math.max(x - pre,ans);
        pre = Math.min(x,pre);
    }
    return ans;
};
```

6、322. 零钱兑换

1、我们采用自下而上的方式进行思考。仍定义 F(i) 为组成金额 i 所需最少的硬币数量，假设在计算 F(i) 之前，我们已经计算出 F(0)−F(i−1) 的答案。 则 F(i) 对应的转移方程应为



2、其中 c_j 代表的是第 j 枚硬币的面值，即我们枚举最后一枚硬币面额是 c_j，那么需要从 i - c_j 这个金额的状态 F(i - c_j) 转移过来，再算上枚举的这枚硬币数量 1 的贡献，由于要硬币数量最少，所以 F(i) 为前面能转移过来的状态的最小值加上枚举的硬币数量 1 。

```
/**
 * @param {number[]} coins
 * @param {number} amount
 * @return {number}
 */
var coinChange = function(coins, amount) {
    let dp = new Array(amount + 1);
    dp[0] = 0;
    // 用现有硬币无法拼凑面额
    for(let i = 1; i <= amount;i++) dp[i] = -1;
    for(let i = 1; i <= amount;i++){
        for(const x of coins){
            // 当前硬币用不上
            if(i < x) continue;
            // 前序状态时非法的
            if(dp[i - x] == -1) continue;
            // 等于-1，或者
            if(dp[i] == -1 || dp[i] > dp[i - x] + 1) dp[i] = dp[i - x] + 1;
        }
    }
    return dp[amount];
};
```

开课吧