愉快的代码【8.5】

【7.27】我们特意加入了题解系统，也就是说，姐姐也会重新做一遍给你们的题目（包括选做），然后在第二天的题目前给出姐姐自己的代码和注释作为题解或参考

如果觉得自己的代码略为臃肿，可以参考对比一下姐姐的代码；

如果觉得姐姐的代码不如自己的优秀，也可以尽情地嘲讽姐姐～

【7.30】我们特意加入了团队系统，因为感觉到你们有点像是独立学习的样子，比如说姐姐和你们之间有交流，但是你们之间有没有交流呢姐姐就感受不到啦

所以正好在洛谷上发现了一个团队系统，我们可以在这上面布置作业呀（当然姐姐也会继续以 pdf 形式布置作业，你们也还是要以 pdf 形式交作业哈），然后你们就可以在上面看到其它妹妹们的代码呀（包括 AC 代码和还未 AC 的整个过程的代码和分数呀），觉得她们表现不够自己好的话，就可以在群里尽情地嘲笑她们呀～

然后那上面还有一个比赛功能哇，具体形式和我们平时的机考差不多，暑假差不多结束了我们也会有一次期末模拟机考的哈～

如果你们开心的话，你们也可以联合起来给姐姐布置一次平时的作业呀，或者给姐姐安排一次机考呀，你们都是团队的管理员了哈

【8.1】准备给你们留个有趣的团队大作业：给姐姐安排一次机考～

具体时间、题数、难度、知识点待定～

【8.3】经过了某些人性与道德的思考，得出了一个奇怪的想法：

“我今天把代码解决了，明天姐姐的代码还有兴趣看嘛”

那就当天放出来好啦～

同样地：如果觉得姐姐的代码不如自己的优秀，也可以尽情地嘲讽姐姐～

今天的题目：

知识点：递归/递推

接下来我们就准备学习 DP 啦～

（不知道有多少妹妹发现今天还有有趣的事情呢

在正式学习之前，需要稍微巩固一下与 DP 有关的一些小思想

递归和递推：两种解状态转移题的方式

（状态转移题：题目可以分出很多个小状态，初始状态好求，最终状态可以从小状态中推导出来

递归是从后往前，递推是从前往后

这两种方法都比较常用，个人偏向递推多点

不过它们和 DFS、BFS 相似

有些题两种都可以写

有些题一种算法好写，另一种极难写

1、https://www.luogu.com.cn/problem/P1044

2、https://www.luogu.com.cn/problem/P1990

今天的答案：

8.5问题1：

```
/*
    洛谷P1044：栈
    思想：思路比较奇特，看最后一个出栈的数x，其中小于x的数有x-1个，大于x的数有n-x个。观察出一个现象：小于x的数入栈->小于x的数出栈->x入栈->大于x的数入栈->大于x的数出栈->x出栈，因此小于x的数会扎堆，大于x的数会扎堆。
    令dp[i]为i个数出栈序列总数目，因此最后一个出栈数为x的出栈序列总数目为dp[x-1]*dp[n-x]。x可以取1～n，因此dp[n] = dp[0]*dp[n-1]+dp[1]*dp[n-2]+...+dp[n-1]*dp[0]
    时间复杂度：O(n^2)
*/

// 递推：从前到后
#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;

int dp[20];
int main() {
    int i,j,n;
    cin >> n;
    dp[0] = 1; // 递推基础值
    for (i=1; i<=n; i++)
        for (j=1; j<=i; j++) dp[i] += dp[j-1]*dp[i-j]; // 递推
    cout << dp[n] << endl;
}

// 递归：从后往前
#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;

int a[20]; // 记忆化搜索：一种配合递归的技术（类似回溯配合DFS）
int dp(int n) {
    if (n == 0) return 1; // 递归基础值
```

```
    if (a[n] != 0) return a[n]; // 之前算过就不用再递归算啦～
    int ans = 0;
    for (int i=1; i<=n; i++) ans += dp(i-1)*dp(n-i); // 递归
    a[n] = ans; // 记忆化搜索
    return ans;
}

int main() {
    int n;
    cin >> n;
    cout << dp(n) << endl;
}
```

8.5问题2：

```
/*
    洛谷P1990：覆盖墙壁
    思想：同样尝试去寻找变化过程，会发现一个2*n的墙壁可以从一个2*(n-1)的墙壁加一块竖着放的长2宽1的砖头形成，可以从一个2*(n-2)的墙壁加两块横着放的长2宽1的砖头形成，可以从一个2*(n-2)然后第(n-1)列还突出一个砖块的墙壁加一块L型砖头形成。
    从上面可以看出，我们还需要求2*n然后第(n+1)列还突出一个砖块的墙壁的覆盖方法。那我们也会发现它可以从一个2*(n-1)的墙壁加一块L型砖头形成（L有两种摆放方式），可以从一个2*(n-1)然后第n列还突出一个砖块的墙壁加一块横着放的长2宽1的砖头形成。
    令 dp[i] 为2*i的墙壁的覆盖方法，dp2[i] 为2*i然后第(i+1)列还突出一个砖块的墙壁的覆盖方法，so dp[n] = dp[n-2]+dp[n-1]+dp2[n-2], dp2[n] = dp[n-1]*2+dp2[n-1]. 记得顺便取模
    时间复杂度：O(n)
*/

// 递推：从前到后
#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;

int dp[1000005],dp2[1000005];
int main() {
    int i,j,n;
    cin >> n;
    dp[1] = 1; // 手算递推初始值（应该很好算吧）
    dp[2] = 2;
    dp2[1] = 2;
    dp2[2] = 4;
    for (i=3; i<=n; i++) {
        dp[i] = (dp[i-2]+dp[i-1]+dp2[i-2])%10000; // 递推
        dp2[i] = (dp[i-1]*2+dp2[i-1])%10000;
    }
    cout << dp[n] << endl;
```

```
}

// 递归：从后往前（本地IDE可能会爆栈）
#include <iostream>
#include <algorithm>
#define ll long long
using namespace std;

int a[1000005],b[1000005]; // 记忆化搜索：一种配合递归的技术（类似回溯配合DFS）
int dp2(int n); // 因为dp要用到dp2
int dp(int n) {
    if (a[n] != 0) return a[n]; // 之前算过就不用再递归算啦～
    a[n] = (dp(n-2)+dp(n-1)+dp2(n-2))%10000; // 记忆化搜索
    return a[n];
}
int dp2(int n) {
    if (b[n] != 0) return b[n]; // 之前算过就不用再递归算啦～
    b[n] = (dp(n-1)*2+dp2(n-1))%10000; // 记忆化搜索
    return b[n];
}

int main() {
    int i,j,n;
    cin >> n;
    a[1] = 1; // 手算递归初始值（应该很好算吧）
    a[2] = 2;
    b[1] = 2;
    b[2] = 4;
    cout << dp(n) << endl;
}
```

Interesting thing:

https://www.luogu.com.cn/contest/32578