C1-B 神奇的数列 题解

题目描述

在一次愉快的网上摸鱼中，`hls`突然发现了一个神奇的数列。这个数列满足如下递推式：

$$H_n = \begin{cases} \sum_{i=1}^{n} H_{i-1}H_{n-1}, & n \geq 2 \\ 1, & n = 0, 1 \end{cases}$$

现在`hls`想知道第i项的值是多少，你能帮帮他嘛？

由于答案可能很大，因此输出需要对 `998244353` 进行取模。

输入

本题测试点包含多组数据。

第一行有1个整数，表示数据组数Q。

接下来Q行，每行一个整数n，表示查询H_n的大小。

数据保证$1 \leq Q \leq 1000$，$1 \leq n \leq 1000$。

输出

对于每组数据，输出Q行，每行为对应H_n取模后的值。

输入样例

```
5
1
3
5
15
20
```

输出样例

```
1
5
42
9694845
574654302
```

解题思路

由于n的取值范围是$[1, 1000]$，所以只需要求出从H_1到H_{1000}的值。

首先可以想到两种思路，一是纯函数递归实现（输入n，输出$f(n)$），二是数组实现（提前打表`a[1]`到`a[1000]`的值，只需要输出对应的`a[n]`即可。

- 若用递归函数实现，首先计算时间复杂度，递推关系式为：

$$H(n) = H(0) * H(n-1) + H(1) * H(n-2) + \ldots + H(n-1) * H(0)$$
$$\ldots$$
$$H(1) = 1$$
$$H(0) = 1$$

- 由前几条递推式估算出，时间复杂度大致为$O(n!)$，所以纯递归的方法必`TLE`。

- 若用数组打表实现，虽然空间复杂度为$O(n)$，比上面的占用空间多，但是由于每次计算H_n的时候，只需要做n次相乘和$n-1$次相加，即$2n-1$次运算，那么计算前n项共需要：

$$1 + 3 + 5 + \ldots + 2n - 1 = \frac{n * (1 + 2n - 1)}{2} = n^2$$

- 次运算，故时间复杂度为$O(n^2)$，比纯递归的时间复杂度要低得多，而且本题数据量为10^3，用数组打表的方法，时间完全足够。

又因为本题有大量的相乘累加，如果先算出结果再对`998244353`求模，很容易出现数据溢出，对此我们同时采用一下两种方法来应对：

- 简单粗暴，所有的数据都用`long long`而不用`int`存，但是要记得格式化输入输出的时候要用`%lld`而不是`%d`。
- 根据求模运算法则（这里只有加减乘，前三条是基础，后两条可推导得到）：

$$(a+b)\%p=(a\%p+b\%p)\%p$$
$$(a-b)\%p=(a\%p-b\%p)\%p$$
$$(a*b)\%p=(a\%p*b\%p)\%p$$

$$a^b\%p=((a\%p)^b)\%p$$
$$(\sum_i^n x)\%p=(\sum_i^n x\%p)\%p$$

- 我们可以利用第一、三条对计算过程进行优化，防止数据溢出。

代码实现

```
    int Q;
    scanf("%lld", &Q);
    int n;

    long long list[1001] = { 0 };
    list[0] = 1;
    list[1] = 1;
    int plus = 0;//累加项
    for (int i = 2; i <= 1000; i++) {
        for (int j = 1; j <= i; j++) {
            plus = (list[j - 1] % MOD) * (list[i - j] % MOD) %
MOD;//计算每个累加项（求模法则三）
            list[i] = ((list[i] % MOD) + (plus % MOD)) % MOD;//累
加求和（求模法则一）
        }
    }
```

（后面的输出部分省略）