

P1990 覆盖墙壁

题目描述

你有一个长为 N 宽为2的墙壁，给你两种砖头：一个长2宽1，另一个是L型覆盖3个单元的砖头。如下图：

```
0 0
0 00
```

砖头可以旋转，两种砖头可以无限制提供。你的任务是计算用这两种来覆盖 $N*2$ 的墙壁的覆盖方法。例如一个 $2*3$ 的墙可以有5种覆盖方法，如下：

```
012 002 011 001 011
012 112 022 011 001
```

注意可以使用两种砖头混合起来覆盖，如 $2*4$ 的墙可以这样覆盖：

```
0112
0012
```

给定 N ，要求计算 $2*N$ 的墙壁的覆盖方法。由于结果很大，所以只要求输出最后4位。例如 $2*13$ 的覆盖方法为13465，只需输出3465即可。如果答案少于4位，就直接输出就可以，不用加0，如 $N=3$ ，时输出5。

输入格式

一个整数 N ($1 \leq N \leq 1000000$)，表示墙壁的长。

输出格式

输出覆盖方法的最后4位，如果不足4位就输出整个答案。

输入样例

13

输出样例

3465

解析

对于此类题型，还是先来理解题意：

1、当长度只有1时，共有几种填充方案？答案是1种。如下所示，我们只能使用A类型的砖：



2、当长度为2时，情况会如何呢？很明显答案为2。

方案1

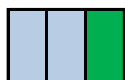


方案2

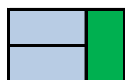


当长度为3时，就是题中例子，方案共五种，如下所示：

方案1



方案2



方案3



方案4

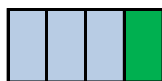


方案5

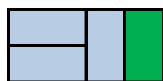


我们继续来分析长度为4时的总方案数，一共是11种。

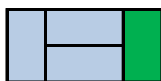
方案1



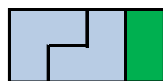
方案2



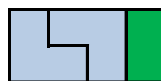
方案3



方案4



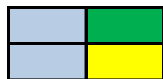
方案5



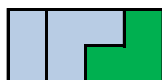
方案6



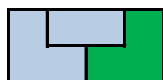
方案7



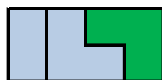
方案8



方案9



方案10



方案11



到了这里，我们就可以开始进行推导了。仔细观察上图，我们可以将以上的图像分成以下4类：

- 1、以  为最后一列，表示为A。
- 2、以  为最后一列，表示为B。
- 3、以  为最后一列，表示为C。
- 4、以  为最后一列，表示为D。

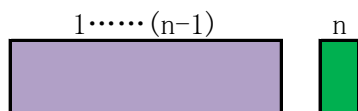
我们设 $F[n]$ 代表将长度为 n 的墙面完整填充的方案数，那么总的方案数可以表达如下：

$$F[n] = A+B+C+D$$


我们可以回想上楼梯的那道题，那么这道题的思路就是到达第 n 层时，我们可以只走1步（插入绿色条），走2步（插入黄绿横条），还可以走1步半（后两种情况）。

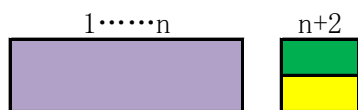
那么接下来的问题就转变成了如何计算A,B,C,D



先来看最好理解的A情况。



假设我们已经知道了填满长度为 $n-1$ 的墙面的方案数 $F[n-1]$ ，那么填满第 n 列的方案数是多少呢？

很明显， $F[n]$ 是等于 $F[n-1]$ 的。因为只有增加  这种方式。



同理，对于以  结尾的情况， $F[n]$ 是等于 $F[n-2]$ 的，因为它只能在最后添加 

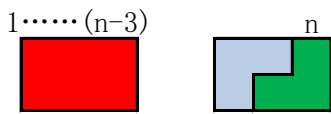
接下来，我们需要求解情况C，它的情况比较复杂。



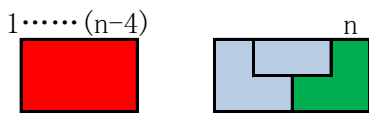
在这里，我们需要进行认真的思考，想要使用  进行填充，那么它之前的形状就必然是下面这个样子（红色代表使用任意方案填充至满）。也就是说，它必须要有一个凸出小块（紫色）



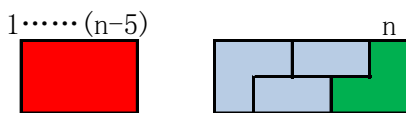
那么，我们的问题就转变成了：形成这个小凸块有多少种方案。尝试绘制一下：



根据上面的思路，这样的情况下： $F[n] = F[n-3]$





同上，这样的情况下： $F[n] = F[n-4]$





这种就是 $F[n] = F[n-5]$ ，等等，我们好像进入了一个无尽的循环……，是的，如果按照这样的方式找下去，将会无休无止。因此，对于紫色凸出的结构，并不适用于使用 $F[n]$ 递推。我们需要一个全新的推导公式。


我们定义 $H[n]$ 代表只被填充了一半的情况：如下图所示：



那么表以 和以 结尾的方案总数。也就是说，上文中的C+D，就等于 $H[n-1]$ 。

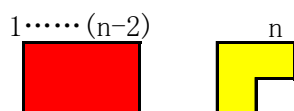
对于以 作为结尾的情况，我们可以分成两类：

第一类，前序第n和n-1列情况为，我们将其命名为E。

第二类，前序第n和n-1列状况为，我们将其命名为F。

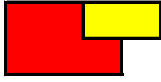
那么，被填充了一半的墙面的递推公式，我们就可以写成： $H[n] = (E + F)/2$ 。为什么除以2呢？那是因为还有另外一半的状态是正好颠倒的位置。


很明显，对于E来说，它的方案数取决于前面被完整填充的方案数，即 $E = F[n-2]$ ，如下所示：



对于F来说呢，它则等于前序H[n-1]的一半（另一半是黄色横条在下面的情况），即：

$F = H[n-1]/2$ ，如下图所示：



另一种以  结尾的方案总数与上面的情况一模一样，只是位置是颠倒的。因此最终的H[n]的递推公式为：

$$H[n] = 2 * F[n-2] + H[n-1]$$

那么总的递推公式就是：

$$F[n] = F[n-1] + F[n-2] + H[n-1]$$

$$H[n] = 2 * F[n-2] + H[n-1]$$

最后还剩下一个问题就是初始值的数据。很明显F[1]=1,F[2]=2，参考文章开始的图片。

那么H[n]呢？

显然，当n=1时，我们无论如何都无法构造出填满半列的方案。因此H[1]=0。

当n=2时，存在两种方案，分别如下：



编码

```
#include<bits/stdc++.h>

using namespace std;
//填满整列的方案数
int F[1000010] = {0};
//填满半列的方案数
int H[1000010] = {0};
//墙的长度
int n;

void Count() {
    //初始化数据
    F[1] = 1;
    F[2] = 2;
    H[2] = 2;
    //从第3列开始递推计算，带入公式即可
    for (int i = 3; i <= n; i++) {
        F[i] = F[i - 1] + F[i - 2] + H[i - 1];
```

```
        H[i] = 2 * F[i - 2] + H[i - 1];
        F[i] %= 10000;
        H[i] %= 10000;
    }
}

int main() {
    cin >> n;
    Count();
    //输出最终的结果
    printf("%d", F[n]);
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。

