以下内容参考自北大版《组合数学》。

k部分拆数

分拆：将自然数n写成递降正整数和的表示。

$$n=r_1+r_2+\cdots+r_k\quad r_1\ge r_2\ge\cdots\ge r_k\ge 1$$

和式中每个正整数称为一个部分。

分拆数□p_n□自然数n的分拆方法数。

自0开始的分拆数：

n	0	1	2	3	4	5	6	7
p_n	1	1	2	3	5	7	11	15

其中恰有k个部分的分拆，称为k部分拆数，记作p(n,k)□

本题要求计算k部分拆数p(n,k)□多组输入，其中n上界为10000□k上界为1000，对1000007取模。

显然□k部分拆数 p(n,k)同时也是下面方程的解数：

$$n-k=y_1+y_2+\cdots+y_k\quad y_1\ge y_2\ge\cdots\ge y_k\ge 0$$

如果这个方程里面恰有j个部分非0，则恰有p(n-k,j)个解。因此有和式：

$$p(n,k)=\sum_{j=1}^k p(n-k,j)$$

相邻两个和式作差，得：

$$p(n,k)=p(n-1,k-1)+p(n-k,k)$$

根据这个可以轻易地写出程序。

```
#include<stdio.h>
#include<string.h>

int p[10005][1005];/*将自然数n分拆为k个部分的方法数*/

int main()
{
    int n,k;
    while(~scanf("%d%d",&n,&k))
    {
        memset(p,0,sizeof(p));
        p[0][0]=1;
        int i;
        for(i=1;i<=n;++i)
        {
            int j;
            for(j=1;j<=k;++j)
            {
```

```
                if(i-j>=0)/*p[i-j][j]所有部分大于1*/
                {
                    p[i][j]=(p[i-j][j]+p[i-1][j-1])%1000007;/*p[i-1][j-1]至
少有一个部分为1。*/
                }
            }
        }
        printf("%d\n",p[n][k]);
    }
}
```

小结论一

生成函数：一种幂级数。各项的系数为数列中的对应项。

由等比数列求和公式，有：

$$\frac{1}{1-x^k }=1+x^k+x^2k+x^3k+⋯$$

$$1+p_1 x+p_2 x^2+p_3 x^3+⋯=\frac{1}{1-x} \frac{1}{1-x^2} \frac{1}{1-x^3}…$$

对于k部分拆数，生成函数稍微复杂。具体写出如下：

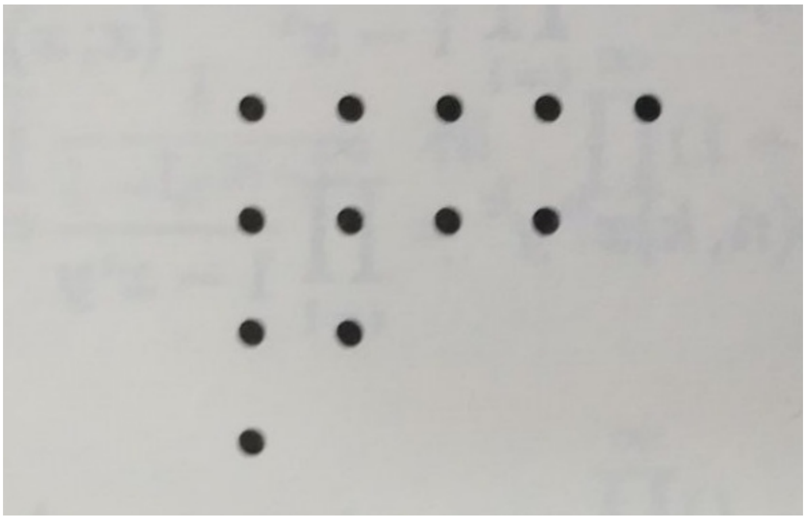
$$∑_{n,k=0}^∞ {p(n,k) x^n y^k }=\frac{1}{1-xy} \frac{1}{1-x^2 y} \frac{1}{1-x^3 y}…$$

小结论二

Ferrers图：将分拆的每个部分用点组成的行表示。每行点的个数为这个部分的大小。

根据分拆的定义Ferrers图中不同的行按照递减的次序排放。最长行在最上面。

例如：分拆12=5+4+2+1的Ferrers图。



将一个Ferrers图沿着对角线翻转，得到的新Ferrers图称为原图的共轭，新分拆称为原分拆的共轭。显然，共轭是对称的关系。

例如上述分拆12=5+4+2+1的共轭是分拆12=4+3+2+2+1。

最大k分拆数：自然数n的最大部分为k的分拆个数。

根据共轭的定义，有显然结论：

最大k分拆数与k部分拆数相同，均为p(n,k)

互异分拆数

互异分拆数pd_n自然数n的各部分互不相同的分拆方法数Different

互异偶部分拆数pe_n自然数n的部分数为偶数的互异分拆方法数Even

互异奇部分拆数po_n自然数n的部分数为奇数的互异分拆方法数Odd

因此有：

$${pd}_n={pe}_n+{po}_n$$

本题要求计算互异分拆数pd_n多组输入，其中n上界为50000，对1000007取模。

同样地，定义互异k部分拆数pd(n,k)表示最大拆出k个部分的互异分拆，是这个方程的解数：

$$n=r_1+r_2+⋯+r_k\quad r_1>r_2>⋯>r_k≥1$$

完全同上，也是这个方程的解数：

$$n-k=y_1+y_2+⋯+y_k\quad y_1>y_2>⋯>y_k≥0$$

这里与上面不同的是，由于互异，新方程中至多只有一个部分非零。有不变的结论：恰有j个部分非0，则恰有pd(n-k,j)个解，这里j只取k或k-1因此直接得到递推：

$$pd(n,k)=pd(n-k,k-1)+pd(n-k,k)$$

代码如下。代码中将后一位缩减了空间，仅保留相邻两项。

```
#include<stdio.h>
#include<string.h>

int pd[50005][2];/*将自然数n分拆为k个部分的互异方法数*/

int main()
{
    int n;
    while(~scanf("%d",&n))
    {
        memset(pd,0,sizeof(pd));
        pd[0][0]=1;
        int ans=0;
        int j;
        for(j=1;j<350;++j)
        {
```

```
            int i;
            for(i=0;i<350;++i)
            {
                pd[i][j&1]=0;/*pd[i][j]只与pd[][j]和pd[][j-1]有关*/
            }
            for(i=0;i<=n;++i)
            {
                if(i-j>=0)/*pd[i-j][j]所有部分大于1*/
                {
                    pd[i][j&1]=(pd[i-j][j&1]+pd[i-
j][(j-1)&1])%1000007;/*pd[i-j][j-1]至少有一个部分为1。*/
                }
            }
            ans=(ans+pd[n][j&1])%1000007;
        }
        printf("%d\n",ans);
    }
}
```

小结论三

奇分拆数：自然数n的各部分都是奇数的分拆方法数。

有一个显然的等式：

$$∏_{i=1}^∞ (1+x^i) =\frac{∏_{i=1}^∞ (1-x^{2i}) }{∏_{i=1}^∞ (1-x^i) }=∏_{i=1}^∞ \frac{1}{1-x^{2i-1} }$$

最左边是互异分拆数的生成函数，最右边是奇分拆数的生成函数。两者对应系数相同，因此，奇分拆数和互异分拆数相同，均为pd_n

分拆数

本题要求计算分拆数p_n多组输入，其中n上界为50000，对1000007取模。

单独观察分拆数的生成函数的分母部分：

$$∏_{i=1}^∞ (1-x^i) $$

将这部分展开，可以想到互异分拆，与互异分拆拆出的部分数奇偶性有关。

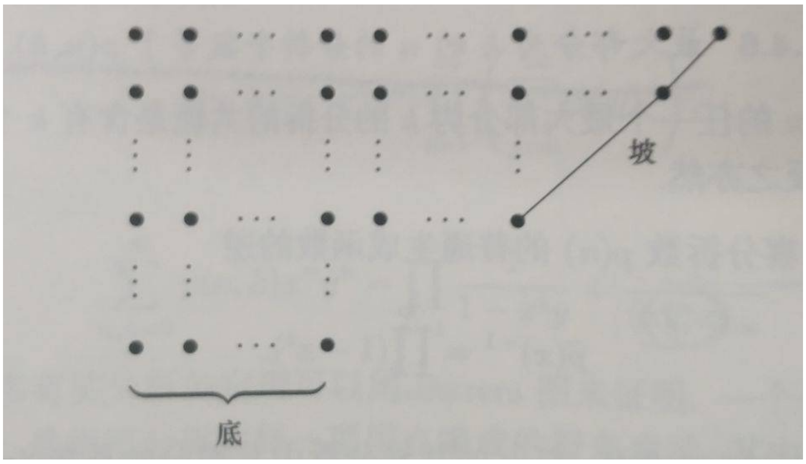
具体地，互异偶部分拆在展开式中被正向计数，互异奇部分拆在展开式中被负向计数。因此展开式中各项系数为两方法数之差。即：

$$∑_{i=0}^∞ ({pe}_n-{po}_n) x^n =∏_{i=1}^∞ (1-x^i) $$

接下来说明，多数情况下，上述两方法数相等，在展开式中系数为0；仅在少数位置，两方法数相差1或-1。

这里只能借助构造对应的办法。

画出每个互异分拆的Ferrers图。最后一行称为这个图的底，底上点的个数记为b□Bottom□□连接最上面一行的最后一个点与图中某点的最长45度角线段，称为这个图的坡，坡上点的个数记为s□Slide□□



要想在互异偶部分拆与互异奇部分拆之间构造对应，就要定义变换，在保证互异条件不变的前提下，使得行数改变1：

变换A□当b小于等于s的时候，就将底移到右边，成为一个新坡。

变换B□当b大于s的时候，就将坡移到下边，成为一个新底。

这两个变换，对于多数时候的n□恰有一个变换可以进行，就在互异偶部分拆与互异奇部分拆之间构造了一个一一对应。已经构造了一一对应的两部分分拆个数相等，因此这时展开式中第n项系数为0。

变换A不能进行的条件：底与坡有一个公共点，且b=s□这种情形只发生于：

$$n=b+(b+1)+\cdots+(b+b-1)=\frac{b(3b-1)}{2}$$

这时，展开式中第n项为：

$$\prod_{i=0}^{b-1} (-x)^{b+i} =(-1)^b \prod_{i=0}^{b-1} x^{b+i} =(-1)^b x^n$$

变换B不能进行的条件：底与坡有一个公共点，且b=s+1□这种情形只发生于：

$$n=(s+1)+(s+2)+\cdots+(s+s)=\frac{s(3s-1)}{2}$$

这时，展开式中第n项为：

$$\prod_{i=1}^s (-x)^{s+i} =(-1)^s \prod_{i=1}^s x^{s+i} =(-1)^s x^n$$

至此，我们就证明了：

$$(1-x)(1-x^2)(1-x^3)\ldots=\cdots+x^{26}-x^{15}+x^7-x^2+1-x+x^5-x^{12}+x^{22}-\ldots=\sum_{k=-\infty}^{+\infty} (-1)^k x^{\frac{k(3k-1)}{2}} $$

将这个式子整理，对比两边各项系数，就得到递推式。

$$(1+p_1 x+p_2 x^2+p_3 x^3+\cdots)(1-x-x^2+x^5+x^7-x^{12}-x^{15}+x^{22}+x^{26}-\ldots)=1$$

$$p_n=p_{n-1}+p_{n-2}-p_{n-5}-p_{n-7}+\cdots$$

这个递推式有无限项，但是如果规定负数的分拆数是0（0的分拆数已经定义为1），那么就简化为了有限

项。

本题中分拆数的计算采用这个方法。附上代码：

```
#include<stdio.h>

long long a[100010];
long long p[50005];

int main()
{
    p[0]=1;
    p[1]=1;
    p[2]=2;
    int i;
    for(i=1;i<50005;i++)/*递推式系数1,2,5,7,12,15,22,26...i*(3*i-1)/2,i*(3*i+1)/2*/
    {
        a[2*i]=i*(i*3-1)/2;/*五边形数为1,5,12,22...i*(3*i-1)/2*/
        a[2*i+1]=i*(i*3+1)/2;
    }
    for(i=3;i<50005;i++)/*p[n]=p[n-1]+p[n-2]-p[n-5]-p[n-7]+p[12]+p[15]-...+p[n-i*[3i-1]/2]+p[n-i*[3i+1]/2]*/
    {
        p[i]=0;
        int j;
        for(j=2;a[j]<=i;j++)/*有可能为负数,式中加1000007*/
        {
            if(j&2)
            {
                p[i]=(p[i]+p[i-a[j]]+1000007)%1000007;
            }
            else
            {
                p[i]=(p[i]-p[i-a[j]]+1000007)%1000007;
            }
        }
    }
    int n;
    while(~scanf("%d",&n))
    {
        printf("%lld\n",p[n]);
    }
}
```