数学

清华大学 张华清

引入

- 考虑一个线性方程组：
- $$\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_1 + x_2 + 2x_3 = 9 \\ x_1 + 2x_2 + 3x_3 = 14 \end{cases}$$
- 我们想给它换个写法：
- $$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 6 \\ 9 \\ 14 \end{pmatrix}$$
- 矩阵　　向量
- 一个矩阵左乘一个列向量，得到的还是一个列向量
  - 咋乘的？把这个列向量横过来，和每一行对应位置相乘再相加。

x_1　x_2　x_3

6　9　14

引入

- 这个方程组的解是$(x_1, x_2, x_3) = (1,2,3)$。带入原方程组：
- $$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 6 \\ 9 \\ 14 \end{pmatrix}$$
- 一个矩阵可以看作是一个作用于向量的线性的运算/映射。
  - 即把向量的每一位带个权值（矩阵的第i行）加起来，得到新向量的第i个值。
  - *每个矩阵都和一个线性变换一一对应。*

f

a_1
a_2
a_3

$1\cdot a_1 + 1\cdot a_2 + 2\cdot a_3$
$1\cdot a_1 + 2\cdot a_2 + 3\cdot a_3$

矩阵优化线性递推

- 斐波那契数列的递推式：$F_{n+1}=F_n+F_{n-1}$。
- 能不能写成矩阵的形式？
- $\begin{cases} F_{n+1}=F_n+F_{n-1} \\ F_n=F_n \end{cases}$
- $\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix}=\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}$
- 那么$\begin{pmatrix} F_2 \\ F_1 \end{pmatrix}=\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$，$\begin{pmatrix} F_3 \\ F_2 \end{pmatrix}=\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$…
- $\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}=\underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\dots\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}}_{\text{n−1个}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$

引入

- 但，好像从右往左一个一个算，也不会变快啊?
- 我们只定义了矩阵对向量的乘法。
- 能不能相应的定义矩阵和矩阵间的乘法?
  - 毕竟，向量可以看作是特殊的矩阵。
- 可以的!
- 两个矩阵相乘仅当第一个矩阵A的列数和第二个矩阵B的行数相等时才能相乘。如A是m*n矩阵，B是n*p矩阵，它们的乘积C是一个m*p矩阵。
- 乘法满足：$C_{i,j} = \sum_{k=1}^{n} A_{i,k} * B_{k,j}$，记该运算为$C = A * B$。
  - 这个定义可能有些奇怪。但我们之后会由一个具体问题，给出一个比较合理的解释。
- 一个演示网站：http://matrixmultiplication.xyz/

引入

$(AB)\cdot C = A\cdot(BC)$

- 矩阵乘法没有交换律，但是它有结合律，即：
- $(AB)C = A(BC)$。

- 单位矩阵I：一个方阵（行数=列数），只有主对角线元素为1，其他都为0。
- 单位矩阵乘任何矩阵都得该矩阵（就像1一样）。$IA = AI = A$。

引入

- 注：我们之前说过，矩阵左乘一个向量，可以看作是对向量做一个线性的映射。
- 对于$u = ABv$，可以认为是v先作用一个B，得到一个v'，然后v'再作用一个A，得到u。
- 而我们对矩阵也定义矩阵乘法，而且发现矩阵乘法有结合律后，就有$u = (AB)v$。相当于是先由映射B和映射A，复合出了一个映射$C = AB$。v直接作用这个C映射，就直接得到u了。

矩阵优化线性递推

- 那么对于$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\cdots\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}}_{\text{n−1个}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$，我们可以写成：
- $\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
- 那，$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$就不用一个一个算了嘛？
- 不用！想想快速幂。矩阵的幂次也能用矩阵快速幂！

```
struct Matrixx{
	long long data[4][4];
	Matrixx(){
		memset(data,0,sizeof(data));
	}
}a,e;
const int p=1000000007;

Matrixx Mul(Matrixx a,Matrixx b){
	Matrixx ans;
	for(int i=1;i<=3;i++){
		for(int j=1;j<=3;j++){
			for(int k=1;k<=3;k++){
				ans.data[i][j]=(ans.data[i][j]+a.data[i][k]*b.data[k][j]%p)%p;
			}
		}
	}
	return ans;
}

Matrixx pow(Matrixx a,int k){
	Matrixx ans=e;
	while(k){
		if(k&1)ans=Mul(ans,a);
		a=Mul(a,a);
		k>>=1;
	}
	return ans;
}
```

高斯消元

- 用于求解线性方程组。
- 就是高斯消元，非常好理解！
- 以这个方程组为例。看我爬黑板！

$$\begin{Bmatrix} 3x & + & 2y & + & z & = & 10 \\ 5x & + & y & + & 6z & = & 25 \\ 2x & + & 3y & + & 4z & = & 20 \end{Bmatrix}$$

高斯消元

- 实现细节：如果在枚举到第i行时，该行第i列已经为0了，那么要从底下换一个不为0的。
- 为了减小精度误差，每次应该选第i列元素绝对值最大的。
- 如果有一行，枚举到它时，前面的系数已经全为0了
  - 后面等于的那个值不为0？无解
  - 为0？无穷多个解。

高斯消元

- 代码实现：细节不少，非常推荐大家自己去写一下

```
for(int i=1;i<=n;i++){
    int mxn=i;
    for(int j=i+1;j<=n;j++){
        if(a[j][i]>a[mxn][i])mxn=j;
    }
    for(int j=1;j<=n+1;j++)swap(a[i][j],a[mxn][j]);
    if(a[i][i]==0){
        puts("No Solution");
        return 0;
    }
    for(int j=1;j<=n;j++){
        if(j==i)continue;
        double h=a[j][i]/a[i][i];
        for(int k=i;k<=n+1;k++){
            a[j][k]-=a[i][k]*h;
        }
    }


}
for(int i=1;i<=n;i++){
    printf("%.2f\n",a[i][n+1]/a[i][i]);
}
return 0;
```

再看矩阵乘法

- 从这样一个问题来再看矩阵乘法。
- 给一张图。因为道路在检修，所以每一天，开通的道路数不一样。
- 第一天，u和v之间有mp1(u,v)条有向边。
- 第二天，u和v之间有mp2(u,v)条有向边。
- 求从x走到y，走恰好两天的方案数(每天走一条边)。

运算

- 显然是，从x出发，走到某个点w，再从w走到y。
- 总方案数为$\text{Ans}[x][y] = \sum_{1\le w\le n} mp1[x][w] * mp2[w][y]$
- mp1是一个矩阵，mp2是一个矩阵，经过这种运算之后得到了一个新矩阵Ans。

运算

- A矩阵的第x行这个向量(长度是A的列数)，点乘上B矩阵的第y列向量(长度是B的行数)。
  - 向量的点乘：对应位置相乘再相加。
- 就得到了Ans矩阵第x行第y列这个位置的值。
- 所以显然要A矩阵的列数等于B矩阵的行数。
- 新矩阵的行数等于A矩阵的行数，新矩阵的列数等于B矩阵的列数。
- 这就是矩阵乘法的定义，即
- $C_{i,j} = \sum_{k=1}^{n} A_{i,k} * B_{k,j}$

运算

$A + B$

- 矩阵乘法的性质：
- 有结合律，但没有交换律
- 对矩阵加法有分配律 $(A+B)C=AC+BC, C(A+B)=CA+CB$
- 对数乘有结合性：$k(AB)=(kA)B=A(kB)$

运算

- A是一个n行m列的矩阵。
- $I_n A = A I_m = A$。

运算

- 一个列数为n的行向量，乘上一个n行m列的矩阵，得到一个列数为m的行向量。
- 一个m列n行的矩阵，乘上一个行数为n的列向量，得到一个行数为m的列向量

优化DP

- 例题1：
- 把引入问题稍微改一下：不修道路了，每天u到v都有mp[u][v]条矩阵。
- 求走k天，从u走到v的方案数?

优化DP

- 考虑DP:ans[t][x][y]为走了t天，从x走到y的方案数。
- $ans[t][x][y] = \sum_{w=1}^{n} ans[t-1][x][w] * mp[w][y]$
- 还是枚举中间点嘛。
- 发现可以写成矩阵乘法的形式：$Ans_t = Ans_{t-1} * Mp$
- 那写下去?
- $Ans_t = Ans_0 * Mp^t$
- Ans0是单位矩阵。所以$Ans_t = Mp^t$

优化DP

- 类似整数的快速幂，可以进行矩阵快速幂。
- 所以原问题从tn^3优化到了$n^3\log t$。

优化DP

- 例题2：
- 衣食无忧的 Q老师 有一天突发奇想，想要去感受一下劳动人民的艰苦生活。具体工作是这样的，有 N 块砖排成一排染色，每一块砖需要涂上红、蓝、绿、黄这 4 种颜色中的其中 1 种。且当这 N 块砖中红色和绿色的块数均为偶数时，染色效果最佳。为了使工作效率更高，Q老师 想要知道一共有多少种方案可以使染色效果最佳，你能帮帮他吗？Input第一行为 T，代表数据组数。(1 ≤ T ≤ 100)接下来 T 行每行包括一个数字 N，代表有 N 块砖。(1 ≤ N ≤ 1e9)Output输出满足条件的方案数，答案模 10007。

优化DP

- 先考虑朴素DP。
- 我们需要记录什么？红，绿的奇偶情况。
- 设f[i][0]为i个格子，红、绿均为偶数。
- f[i][1]为i个格子，红绿均为奇数
- f[i][2]为i个各自，红绿一奇一偶
- （其实更自然的想法应该是记录红为奇绿为偶和红为偶绿为奇，但因为红色和绿色没有本质区别，所以可以放在一起记录。

优化DP

$$\begin{pmatrix} 2 & 0 & 1 \\ 0 & 2 & \\ 2 & 2 & 2 \end{pmatrix}$$

- 转移显然：
- f[i][0] = 2 * f[i-1][0] + C[i-1][2]
  f[i][1] = 2 * f[i-1][1] + C[i-1][2]
  f[i][2] = 2 * f[i-1][0] + 2 * B[i-1][1] + 2 * C[i-1][2]
- 我们先定义一个"转移系数"

优化DP

- f[i][0] = 2 * f[i-1][0] + C[i-1][2]
  f[i][1] = 2 * f[i-1][1] + C[i-1][2]
  f[i][2] = 2 * f[i-1][0] + 2 * B[i-1][1] + 2 * C[i-1][2]
- 能不能写成矩阵乘法的形式？
- 定义一个长为3的行向量F[i]，就是F[i]的第j个元素即为f[i][j]。
- 然后定义一个3*3的转移矩阵Mp。让F[i-1]*Mp=F[i]。
- Mp[x][y]是啥？就是f[i-1][x]到f[i][y]的转移系数。

(F[i-1][0] F[i-1][1] F[i-1][2])

(F[i][0] F[i][1] F[i][2])

O(log n)

优化DP

- 所以什么时候可以用矩阵快速幂优化DP呢？
- 转移形如矩阵乘法（i-1的x对i的y有一个mp[x][y]的贡献）
- 每个位置本质相同（这样每个位置的转移矩阵才是一样的，才能快速幂嘛。

优化DP

- 例题3：

一首**歌**可以看作由**音符**组成的一个长为n序列。

一共有m种**音符**。定义一首歌是**好听的**，当且仅当对于这首歌的任意一个长为x的连续子序列，至少有$x-1$种音符。

小Z想知道，一共有多少首好听的歌？他知道，有些事情不是想干就能干成的，所以仅仅知道这个答案对998244353取模的结果都能让他高兴好久！

因为他太菜了，请你教教他！

$$1 \le n \le 10^9, 1 \le x \le 300, x \le m \le 10^9$$

优化DP

- 等价于任意长为x的连续段，最多只有一对位置颜色相同。
- 怎样记录状态？只有最后x个位置有用（事实上，只有最后x-1个有用，不过就多记录一个吧，方便理解）
- 难道要把颜色都记录下来？
- 关于颜色的常见想法：不关心具体颜色。只关心相对关系。具体颜色只需要在转移时乘个系数即可（见代码）
- 我们只需要记录是哪一对点颜色相同即可。

优化DP

- 状态数还是有点多。真的需要记录一对点吗？
- ——只需要记录两个点中靠前的那个即可！！
- 因为实际上，记录的信息的作用只是让你知道：啥时候两个相同的种，靠前的那个不再是后x个了；也就是说可以再产生一对相同的。所以你发现，靠后的那个对你来说没用。

- 我们应该学会根据数据范围推测做法，优化算法。

优化DP

- 怎么转移?

```
f[x][0]=get_A(m,x);
long long h=get_A(m,x-1);for(int i=1;i<=x-1;i++){
    f[x][i]=h*(x-i)%p;//还有这么多可以放的
}
for(int i=x+1;i<=n;i++){//从i-1转移来:
    //f[i-1][0]:
    f[i][0]=add(f[i][0],f[i-1][0]*(m-x+1)%p);
    for(int j=1;j<=x-1;j++)f[i][j]=add(f[i][j],f[i-1][0]);//只能放对应位置
    f[i-1][1]:即将没了
    f[i][0]=add(f[i][0],f[i-1][1]*(m-x+1)%p);
    for(int j=1;j<=x-1;j++)f[i][j]=add(f[i][j],f[i-1][1]);//只能等于对应位置

    for(int j=2;j<=x-1;j++){
        f[i][j-1]=add(f[i][j-1],f[i-1][j]*(m-x+2)%p);
    }
}
    for(int i=x;i<=n;i++){
```

优化DP

- 又是每个位置本质相同。把转移系数写成矩阵的形式，就可以啦！

优化DP

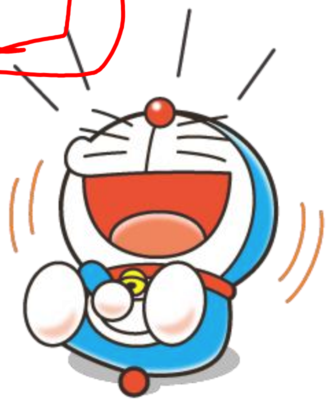
- 广义矩阵乘法：
- 例题1*：刚刚我们求了：走t步，从x走到y的方案数。
- 现在我们给每条边加一个边权（两点之间最多一条边）。求走t步，从x走到y的最大边权和。
- DP方程从$ans[t][x][y] = \sum_{w=1}^{n} ans[t-1][x][w] * mp[w][y]$变成了$ans[t][x][y] = max_{w=1}^{n}(ans[t-1][x][w] + mp[w][y])$
- （注意，如果没有边，边权不是0而是-inf。

$n^3 \log n$

$m \sim n^2$

$(\times, +)$

$(+, max)$



优化DP

- 这也可以用矩阵快速幂。只不过把乘法换成加法，把求和换成去max。
- 就(*,+),(+,max),(+,min)矩阵乘法都满足结合律，都可以用矩阵快速幂。

优化DP

- 广义矩阵乘法:
- 例题1**：刚刚我们求了：走t步，从x走到y的方案数/最大边权。
- 现在我们求，走t步，从x走到y的可达性。即，能不能从x走到y。
- $n<=1000$

- 当然，我们可以先求方案数，看看是否大于0即可。
- 但可以毕竟可达性是更弱的一个问题。所以应该有更快的方法:

优化DP

- DP方程从$ans[t][x][y] = \sum_{w=1}^{n} ans[t-1][x][w] * mp[w][y]$ 变成了
- $ans[t][x][y] = \mathrm{OR}_{1 \le w \le n}(ans[t-1][x][w] \;\&\; mp[w][y])$
- 就是先and，只要有一个是1就都是1。

ans[t][x][y] | mp[w][y]

优化DP

- 这样写。

Mul

n^3

```
    Mat anss;
    for(int i=1;i<=n;i++){
        for(int k=1;k<=n;k++){
            if(a.mt[i][k]){
                anss.mt[i]|=b.mt[k];//对于所有的j，都有anss[i][k]|=b[k][j];
            }
        }
    }
    return anss;
```

- 其实好感性理解：如果i能到k，那么k能到的地方，i都能到。

(&, OR)

矩阵优化线性递推plus

- f[i]=f[i-1]+f[i-2]
- f[i]=f[i-1]+f[i-2]+c
- f[i]=f[i-1]+f[i-2]+i
- f[i]=sum[i-1]+f[i-2]

矩阵优化线性递推plus

- 只需要额外记录一个”1”，或”sum”

数据结构维护矩阵乘法

- 考虑这样一个简单问题是否可行：
- 一个序列，每个位置是一个矩阵。
- 单点修改，每次询问一个区间内的矩阵，从左到右乘起来，得到的矩阵是多少？

数据结构维护矩阵乘法

- 线段树即可！ （就像普通乘法那样做就行。
- 为啥可以呢？因为他是有结合律的！

优化DP——动态DP

- 刚才讲了用矩阵快速幂优化DP，那要求每个位置转移矩阵一样。
- 这里则不然，每个位置转移矩阵不一样（所以n不能是1e9那么大）
- 使用数据结构维护矩阵，可以支持带修/多次询问某一段的值。
- （洛谷模板是树上的。但并不一定都是树上问题。

SP1716 GSS3 - Can you answer these queries III

n 个数，q 次操作

操作 `0 x y` 把A_x 修改为y

操作 `1 l r` 询问区间$[l,r]$ 的最大子段和

SP1716 GSS3 - Can you answer these queries III

- 先考虑怎么用DP写：
- f[i]为以i结尾的最大子段和是多少。
- 则f[i]=max(f[i-1]+v[i],v[i])
- 设g[i]为1...i中，最大子段和为多少。则g[i]是f[i]的前缀最大值，即：
- g[i]=max(g[i-1],f[i])

SP1716 GSS3 - Can you answer these queries III

$D[l,r]$

- 怎么写成广义矩阵乘法的形式?
- v[i]放在转移矩阵里。分别考虑f[i-1],g[i-1]对f[i],g[i]的贡献就行啦!
- 有点困难？困难在单独的v[i]咋整?
- 单独在行向量里加一个元素——0
- 即考虑一个行向量Ai=[f[i],g[i],0]，考虑A[i-1]怎么转移到A[i]即可

$m\log n$

SP1716 GSS3 - Can you answer these queries III

- 实现上：线段树上每个叶子节点就存一个转移矩阵。
- 其他节点，存它这个区间的矩阵从左到右乘起来是啥。
- 然后询问时，查询线段树上[l,r]的转移矩阵的成绩。就像回答区间和那样就可以啦！
- 然后用初始向量[0,-INF,0]去乘一下，发现答案就是max(A[1][2],A[3][2])（下标从1计数。

6569 [NOI Online #3 提高组] 魔法值

- 就感觉像是，你这个1，是通过一条路径传给首都的。
- 也就是说，x这个点到首都有几条长恰好为ai的路径，首都就会被异或几次f_i。
- 于是我们只需要关心所有点到首都长为a_i路径方案数的奇偶性。如果为奇数则答案异或上f_i。——这不是我们讲过的问题嘛！
- 但是有多组询问欸，直接矩阵快速幂复杂度q*n^3loga，好像过不去

6569 [NOI Online #3 提高组] 魔法值

- 法一：求方案数的奇偶性，也即对2取模。
- %2意义下，乘法相当于&，加法相当于^。
- 可以使用bitset优化。
- 即A矩阵的第x行和B矩阵的第y列&起来，再看看几个1。奇数个ans[x][y]就为1，否则为0。

6569 [NOI Online #3 提高组] 魔法值

- 法二：多组询问的经典套路。
- 我们要求一个Ans_k=Ans_0*Mp^k，其中Ans_i是一个长度为n的行向量，表示从1出发，走恰好i步走到x这个点的方案数%2.
- 就我们是在求一个行向量和log k个n*n的矩阵的成绩。
- 那么，我们先预处理Mp^k。
- 然后，算答案的时候，我们别先算这log k个方阵的乘积啊！
- 我们就用Ans_0向量从左乘到右！因为向量乘矩阵复杂度是O(n^2)的！
- 这样复杂度就从q*n^3logt，变成了n^3logt+q*n^2logt

6569 [NOI Online #3 提高组] 魔法值

- 两个优化方法使用其一即可过。
- 两个都用巨大快。

数论

- 前置知识：
  - 模运算
  - 快速幂
  - 进制转换
  - 埃拉托斯特尼筛
  - 唯一分解定理
  - 辗转相除法

线性筛素数（欧拉筛）

- 埃氏筛，每个合数会被筛多次，具体来说，会被筛它的质因数个数次。所以即使它接近O(n)，也不是O(n)
- 所以有一个更高明的筛法，即线性筛
- 每个合数会且只会被它的最小质因数筛一遍
- 所以总复杂度就是O(n)的。
- 具体看代码

线性筛素数（欧拉筛）

nloglogn

O(n)

```
for(int i=2;i<=n;i++){
    if(isn_pri[i]==false){
        sta[++tot]=i;
    }
    for(int j=1;j<=tot&&i*sta[j]<=n;j++){
        isn_pri[i*sta[j]]=true;
        if(i%sta[j]==0){
            break;
        }
    }
}
```

- 比较重要的就是这句：if(i%sta[j]==0)break;
- i%sta[j]==0，说明第j个质数(sta[j])是i的质因数，并且它是i的最小质因数（因为是从小到大枚举的）。
- 那么下一个质数sta[j+1]，就比i的最小质因数大了
- 那么sta[j+1]就不是i*sta[j+1]的最小质因数了。所以我们break掉。
- i*sta[j+1]会在后面被筛到（具体来说，会在新的i’=i*sta[j+1]/sta[j]时被筛到

裴蜀定理

- 当且仅当gcd(a,b)|c，也即c是gcd(a,b)的倍数时，ax+by=c才有并且一定有整数解。

a, b, c

- 换言之，a和b的线性组合能表示出的数，都是gcd(a,b)的倍数。

ax+by=c

- 证明必要性：设gcd(a,b)=g。因为a是g的倍数，b是g的倍数，所以ax,by都是g的倍数，所以c必须是g的倍数。也就是说，如果c不是g的倍数，则一定没有解。

ax+by=g ?

- 证明充分性——即如果c是g的倍数，ax+by=c一定有解
- ——扩展欧几里得算法本身即是一个构造性的证明。

ax+by=gcd(a,b)

扩展欧几里得算法

- 求解这样的问题：给定a,b,c，求$ax+by=c$的任意一组合法整数
- 转化成求$ax+by=gcd(a,b)$的一组解，之后把x,y分别乘上$c/gcd(a,b)$就好啦
- 举个例子：$4x+6y=14$
- 我们先求出$4x+6y=2$的任意一组解，比如说$x=2,y=-1$或$x=-1,y=1$
- 然后再x,y同时都乘上$14/2=7$，那么$x=14,y=-7$或$x=-7,y=7$则都是解了。
- 首先，一个特殊情况时，当$b=0$时，$gcd(a,b)=a$，我们直接让$x=1,y=0$就ok了。

扩展欧几里得算法

- 首先，有一个特殊情况时，当b=0时，gcd(a,b)=a，我们直接让x=1,y=0就ok了。（gcd(a,0)=a是定义。

$ax+by=gcd(a,b)$

当$b=0$时

$x=1, y=0$

$ax=gcd(a,0)$

$=a$

扩展欧几里得算法

- 我们由辗转相除法，类比一下扩展欧几里得算法。
- 辗转相除法中，我们原来想求gcd(a,b)，但不会求。
- 这时，我们发现，gcd(a,b)=gcd(b,a%b)
- 于是，我们转而去求gcd(b,a%b)。
- 还不会求就再递归一层。
- 这个过程就是在"递归求解"
- 啥时候停下来呢？当b=0时，我们会求gcd(a,b)了，所以直接一路返回就好了

gcd(a,0)=a

扩展欧几里得算法

- 我们来举一个例子：
- 想求gcd(6,4)，不会求，递归下去。
- gcd(6,4)等于gcd(4,6%4)=gcd(4,2)，不会求，递归下去。
- gcd(4,2)=gcd(2,0)。b=0，会求了！返回2
- 所以gcd(4,2)=2
- 所以gcd(6,4)=2

扩展欧几里得算法

x=x'
y=y'

- 我们回到扩展欧几里得算法
- 现在，我们想求$ax+by=gcd(a,b)$的一组解(x,y)，不会求。
- 咋办?
- ——我们先求出$(b)x'+(a\%b)y'=gcd(b,a\%b)$的一组解$(x',y')$。我们希望由$(x',y')$，推出$(x,y)$。
- 在辗转相除法中，直接有$gcd(a,b)=gcd(b,a\%b)$，比较简单。
- 但扩展欧几里得算法中，并不是$x=x'$,$y=y'$，而是需要稍微计算一下：

扩展欧几里得算法

- 咋算呢：
- 我们已经知道了：$bx' + (a\%b)y' = gcd(b,a\%b)$的一组解。
- 我们又知道：$gcd(a,b) = gcd(b,a\%b)$

$bx' + (a - [\frac{a}{b}] \cdot b) \cdot y' = gcd(a,b)$

- 所以说：$bx' + (a\%b)y' = gcd(a,b)$
- 根据模运算的定义，$a\%b=a-[a/b]*b$。

$ay' + b(x' - [\frac{a}{b}]y') = gcd(a,b)$

- 带入进去：$ay' + b(x' - [a/b]y') = gcd(a,b)$。这就是根据已知信息推出的一个结果。
- 诶，我们想求啥来着？$ax+by=gcd(a,b)$的一组解。
- 那岂不就是：$x=y'$，$y=x'-[a/b]y'$ 恰好就是$ax+by=gcd(a,b)$的一组解嘛！

扩展欧几里得算法

- 所以我们的思路就是：
- 想求ax+by=gcd(a,b)的一组解(x,y)，不会求。
- 这时，我们发现，如果我们求出了bx′+(a%b)y′=gcd(b,a%b)的一组解(x′,y′)，我们就能求出(x,y)（x=y′,y=x′-[a/b]y′）
- 于是，我们转而去求bx′+(a%b)y′=gcd(b,a%b)的一组解(x′,y′)（相当于原来的b成为了新的a，原来的a%b成为了新的b）
- 还不会求就再递归一层
- 这个过程就是递归求解
- 啥时候停下来呢？我们有一个特殊情况！就是b=0的时候，我们会解！只需要让x=1,y=0即可！

扩展欧几里得算法

- 也举个例子：
- 想求6x+4y=gcd(6,4)的解(x,y)，不会解，递归下去
- 转而去求4x’ +2y’ =gcd(4,2)的解(x ‘,y’)，不会解，递归下去
- 转而去求2x’ ’ +0y’ ’ =gcd(2,0)=2的解(x’ ’ ,y’ ’)，会解了！x’ ’ =1,y’ ’ =0
- 返回一层！x’ =y’ ’ =0，y’ =x’ ’ -[a/b]y’ ’ =1-[4/2]*0=1
- 再返回一层！x=y’ =1,y=x’ -[a/b]y’ =0-[6/4]*1=-1
- 所以解完了！x=1,y=1就是6x+4y=gcd(6,4)的一组解！
- （注意，每一层的a,b都是这一层的a,b.

扩展欧几里得算法

- 那考虑辗转相除法的过程，最后一定递归到b=0的情况，此时让x'=1,y ‘=0一步一步回带，就可以了

扩展欧几里得算法

- 看看代码?

```
long long exgcd(long long a, long long b, long long &x, long long &y){
//  cout<<a<<' '<<b<<endl;
    if(b==0){
        x=1, y=0;
        return a;
    }
    long long h=exgcd(b, a%b, x, y);
    long long xx=x, yy=y;
    x=yy, y=xx-a/b*yy;
    return h;
}
```

- xx,yy就是x',y'

扩展欧几里得算法

gcd　　　　(x_0, y_0)

- 现在，我们求出了ax+by=c的某一组解x0,y0，我们可不可以由这一组解，得到所有的(x,y)满足ax+by=c? 即我们找到所有的通解。

gcd

- 我们先感性的想一想，要么是x0变大些、y0变小些，要么是x0变小些，y0变大些。（废话hh
- 那么x0最小要变大多少呢?

$$ax_0+by_0=g$$

扩展欧几里得算法

$ax_0 + by_0 = g$

- 假设x0变大了r，那么ax变成了a(x0+r)，变大了ar
- 所以by需要减小ar。
- 我们还要求y得是个整数，所以需要保证ar是b的倍数。
- 不妨设H=ar，我们想求最小的r，也就是求最小的H，满足H是b的倍数。同时H也是a的倍数。
- 那么H是啥？a和b的最小公倍数啊！

扩展欧几里得算法

- 所以说H=lcm(a,b)=ar
- 又有lcm(a,b)=a*b/gcd(a,b)=ar
- 所以r=b/gcd(a,b)
- 那么此时b0也要减少a/gcd(a,b)
- 所以通解就是x=x0+k*(b/gcd(a,b))，y=y0-k*(a/gcd(a,b))。
- k是任意的整数（可以是负数昂）

$ar = \mathrm{lcm}(a,b)$

$r = \dfrac{\mathrm{lcm}(a,b)}{a}$

$\dfrac{b}{\gcd(a,b)}$

扩展欧几里得算法

- 另一种思考方式：还是考虑x0变大一些，y0变小一些。
- ax0+by0=gcd(a,b)
- =>ax0+H-H+by0=gcd(a,b)
- a(x0+H/a)+b(y0-H/b)=gcd(a,b)
- 所以H是a的倍数，H是b的倍数，最小的H就是lcm(a,b)
- 所有合法的H都是 lcm(a,b)的倍数。

同余方程转不定方程

题目描述

求关于x的同余方程 $ax \equiv 1 \pmod{b}$ 的最小正整数解。

C　$ax+by=C$

$ax \equiv 1 \mod b$

x

$ax+ky=1$

$ax+kb=1$

$ax+by=1$

$\exists y$

同余方程

- $ax \equiv 1 \bmod b$?
- 看起来和我们上面”ax+by=c”的柿子…长得很不一样诶..吗?
- 它等价于$ax+by=1$!
- 那就可以用扩展欧几里得算法直接求啦!
- 要求输出最小正整数解。通解是啥? $x0+b/gcd(a,b)$
- 本题中，$gcd(a,b)=1$。
- 所以输出$(x0\%b+b)\%b$就好了。为啥先%再加再%? 这是负数取模到正数的方法。

同余方程

- ax≡1 mod b?
- 看起来和我们上面”ax+by=c”的柿子…长得很不一样诶..吗?

同余方程

- 如果$A \equiv C \bmod B$
- 那么A和C就相差了若干个B。
- 也就是说，$A \equiv C \bmod B$，等价于A+By=C。其中y是整数

- 那么$ax \equiv 1 \bmod p$就等价于ax+py=1，其中x,y都是整数。
- ax+py=1有无穷多组解(x,y)，那么 $ax \equiv 1 \bmod p$ 也有无穷多组解x。至于y，是我们引入的一个工具人，不用再管它了。

逆元—扩欧

- 给定一个整数a，和模数p，求一个整数x，满足$a*x \equiv 1 \mod p$
- 则x就是a在模p意义下的逆元。
- 咳咳，不就是上面的题嘛！用扩欧做就好啦
- 所以你发现，不是每个数a在模p意义下都有逆元。有解的条件和$ax+py=1$有解条件一样：a,p需要互质。

逆元—费马小定理

- 这是求解模数为质数的逆元的另一种方法。
- 当p为质数时，如果a和p互质，那么我们有：
- ap-1≡1 mod p
- 那么a*ap-2≡1 mod p
- 我们不是要求a*x≡1 mod p的x嘛
- 那么ap-2就是x咯！

$a^{p-1} \equiv 1$

$a^{p-1} \equiv 1 \mod p$

$a \cdot (a^{p-2}) \equiv 1 \mod p$

$a \cdot x \equiv 1 \mod p$

有理数取模

- 很多题都会用到，比如某个题的答案是个分数a/b，它让你输出a/b对p取模的值。
- 意思就是要你求出b在模p意义下的逆元x，然后输出a*x%p即可。
- 其实逆元满足bx≡1 mod p，也就是说x≡1/b mod p
- 或者说x在模p意义下就相当于1/b了。（这很重要！）

$\frac{a}{b} = a \cdot \frac{1}{b}$ 逆元

有理数取模

- 2613 【模板】有理数取余
- 这个题魔鬼的地方是，a,b都超级大。
- 不要怕，它不会难为你。
- 分子分母分别对p取模后再求逆元再乘，答案是一样的嘛
- 所以用类似快读的方法，边读入边对p取模就可以了

$(a \cdot b)\%p$

$= (a\%p \cdot b\%p)\%p$

$(a \cdot 10 + x_i - '0')\%p$

$(a\%p \cdot 10\%p + x_i - '0'\%p)\%p$

逆元—线性求逆元

- 3811 【模板】乘法逆元
- 要你线性地求出1...n内每个数在mod p意义下的逆元。
- 这里介绍一种我常用的，比较好理解的算法。

逆元—线性求逆元

- 首先，我们预处理出1..n内每个数的阶乘jc[i]=i!
- 然后，我们算出jc[n]的逆元jcn[n]
- 然后，我们算出1...n内每个数的阶乘的逆元：
- jcn[i]=jcn[i+1]*(i+1)
- 考虑jcn[i]相当于1/(1*2*3*..*i)
- 而jcn[i+1]相当于1/(1*2*3*...*i*(i+1))

$a^{\varphi} \equiv 1$, $\frac{1}{a}$

① $\frac{1}{i}$ $i!$

② $\frac{1}{n!}$

③ $\frac{1}{i!} = \frac{1}{(i+1)!}(i+1)$

④ $\frac{1}{i} = \frac{1}{i!} \cdot (i-1)!$

逆元—线性求逆元

- 最后，我们就可以求出每个数的逆元啦！
- inv[i]=jcn[i]*((i-1)!)
- 考虑jcn[i]相当于1/(1*2*3*..*(i-1)*i)
- 而inv[i]相当于1/i

模运算的一个结论：

- 分配律： $c(x \mod y) = cx \mod cy$

同余方程的几个结论：

- 对于同余方程 $cx \equiv cy \bmod cp$， 我们可以两边，连着模数，一起除掉这个c，得到$x \equiv y \bmod p$这是常见的操作。
- 证明：
- $cx\%cp = cy\%cp$
- 由上页，$cx\%cp = c(x\%p), cy\%cp = c(y\%p)$
- 那么$x\%p = y\%p$，即$x \equiv y \bmod p$

同余方程的几个结论：

- 当我们得到同余方程
- $ax \equiv ay \ mod \ p$时，我们可能下意识地去两边去掉a，但这是不对的。
- 只有当a,p互质的时候才能这么做，因为这个时候才有逆元。
- 那么当a,p不互质时怎么办？
- 想办法让他们互质

同余方程的几个结论：

- 求出d=gcd(a,p)
- 结合上一页：两边连同模数p一起除以d，令$a' = \frac{a}{d}, p' = \frac{p}{d}$
- 那么我们得到：$a'x \equiv a'y \bmod p'$
- 注意此时a′ 和p′ 互质了，换言之a′ 在模p′ 意义下有逆元了，那么我们就可以两边除掉a′ ，得到$x \equiv y \bmod p'$

同余方程的几个结论：

- 总之，我们得到了：
- $ax \equiv ay \ mod \ p, d = \gcd(a,p)$，则$x \equiv y \ mod \ \frac{p}{d}$

同余方程的几个结论：

关于同余方程的一个常用结论

对于一个同余方程$x \equiv a \mod m$，设m的质因数分解为$m = \prod p_i^{vi}$

那么这个方程等价于方程组$\forall i, x \equiv (a\% p_i^{vi}) \mod p_i^{vi}$

- 为啥呢?
- 考虑CRT的过程，我们对于方程组，我们知道它在[0,m)之内有且仅有一个解，而这个解就是上面同余方程x=a mod m的解。
- 数论题常常有这样的操作：对于模数的每个质因数次幂分别求解，最后用CRT合并，就是这个原理了。比如扩展卢卡斯定理（这里并不讲。）

[NOI Online #1 提高组]最小环

- 我们只考虑这个题和数学有关的一点部分，即：
- 总共有m个点，编号分别为0,1,2...m-1
- x从0号点开始，每次跳到(x+n)%m，它的轨迹是什么?

[NOI Online #1 提高组]最小环

- 是一个环。
- 这个环长是多少？或者说，x跳了几次又回到了0号点?
- 假设跳了t次，那么x现在在(n*t)%m这个点。
- 那么nt需要是m的倍数。
- 那么nt 既是n的倍数，又是m的倍数，那么nt最小是?
- lcm(n,m)!
- 那么t=?
- lcm(n,m)/n=m/gcd(n,m)

[NOI Online #1 提高组]最小环

- 如果x从0,1,2...,m-1分别作为起点开始跳，总共形成几个环?
- $m/t=m/(m/gcd(n,m))=gcd(n,m)$

[NOI Online #1 提高组]最小环

- x从0号点开始跳，会经过哪些点?
- 0,d,2d,3d…m-d
- 其中d是gcd(n,m)
- 这相当于是问$kn\%m$有几个可能的取值
- 联想讲过的同余方程转不定方程
- 实际上是问kn+pm=A，对于哪些A有解！
- 哪些A有解呢？不就是裴蜀定理告诉我们的，A需要时gcd(n,m)的倍数嘛
- 注意这里，实际上裴蜀定理要求k,p取值范围是整数（不一定是正的）
- 但这里k必须是正的，但p可以是负的，所以没影响。（我们讲过通解怎么求了，只要有解，k多正，p就可以多负去抵消）

[NOI Online #2 提高组] 涂色游戏

- 可以填两种颜色的，填哪种颜色？
- 不妨令n=p1,m=p2且n<m （这样方便一点hhh
- 显然填蓝对应颜色hhh

- 显然不需要考虑到1e20，因为它以lcm(n,m)为周期循环。

[NOI Online #2 提高组] 涂色游戏

- 考虑k最小可以是多少？即连续相同颜色的球最多几个？
- 显然最多连续相同颜色的球，他们的颜色是红色（即n对应颜色）
- 因为n<m，则两个蓝球之间最多只能有一个红球。

[NOI Online #2 提高组] 涂色游戏

- 那么最多连续几个?
- 只需要让某个红球后第一个蓝球出现的尽量早。
- 最早是多早?
- 即kn%m的最小值!
- 和之前一样，由裴蜀定理，是gcd(n,m)

[NOI Online #2 提高组] 涂色游戏

- 那么我们可以得到，答案是(m-1-gcd(n,m))/n+1（整除）
- 有一些情况需要特判

[NOI Online #2 提高组] 涂色游戏

- 这个题还有一个值得我们注意的地方：
- 他让我们求，对于一个给定的k，是否合法。
- 我们不去直接验证这一个k是否合法，而是去求合法k的最小值。
- 这个题是k越大越容易满足条件，所以求最小值即可。
- 如果不是k越大越容易满足条件，可能还得求最大值。
- 这个想法看起来很自然，但有的时候却不容易想到。
- 在OI和文化课中都会见到这种题。

欧拉定理

$\varphi(m)$: 1 --- m 中，有多少个正整数和 m 互质

$\varphi(1)=1$ $\varphi(3)=2$ $\varphi(4)=2$

- 若$\gcd(a, m) = 1$，则$a^{\phi(m)} \equiv 1 \bmod m$
- 证明：
- 取$1 \dots m-1$中所有与m互质的数$r_1, r_2, r_3 \dots r_{\phi(m)}$。
- 那么$ar_1\%m, ar_2\%m, ar_3\%m \dots ar_{\phi(m)}\%m$也是同样的这些互质的数。
- 那么$r_1 r_2 r_3 \dots r_{\phi(m)} \equiv ar_1 ar_2 ar_3 \dots ar_{\phi(m)} \bmod m$
- 两边约去$r_1 r_2 r_3 \dots r_{\phi(m)}$，得到$a^{\phi(m)} \equiv 1 \bmod m$。

卢卡斯定理

$\binom{n}{m} = C(n,m)$

- 卢卡斯定理：对于质数p，我们有：
- $\binom{n}{m} \bmod \mathrm{p} = \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor} \cdot \binom{n \bmod p}{m \bmod p} \bmod p$
- 可以用于n, m很大，但p很小的求解。$\binom{n \bmod p}{m \bmod p}$范围已经很小了，可以直接求解。$\binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor}$呢？再用卢卡斯定理，递归下去求解！

6669 [清华集训2016] 组合数问题

×1024 ×1024

- 小葱想知道如果给定 n,m 和 一个质数k，求对于所有的$0 \le i \le n, 0 \le j \le \min(i,m)$ 有多少对 (i,j) 满足 C(i,j)是k 的倍数。

4

..8

对于 100% 的测试点，$1 \le n, m \le 10^{18}$，$1 \le t, k \le 100$，且 k 是一个质数。

6669 [清华集训2016] 组合数问题

- 我们更近一步地使用卢卡斯定理。
- 假设n的k进制为$n=(n_1n_2\ldots n_r)_k, m=(m_1m_2\ldots m_r)_k$。
- 那么$C(n,m)=\prod_{i=1}^{r}C(n_i,m_i)$。
- $C(n,m)=0$当且仅当存在$C(n_i,m_i)=0$，当且仅当存在$m_i>n_i$。

6669 [清华集训2016] 组合数问题

- 使用数位DP！
- 设f[i][0/1][0/1][0/1];//考虑较低的i位，n是否贴上界，m是否贴上界，是否已经存在一位$C(n_i,m_i)$为0。

```
int dp(int x,bool is_n,bool is_m,bool flg){
    if(f[x][is_n][is_m][flg]!=-1)return f[x][is_n][is_m][flg];
    if(x==0)return flg;
    int ans=0;
    for(int i=0;i<=(is_n?ta[x]:K-1);i++){
        for(int j=0;j<=(is_m?tb[x]:K-1);j++){
            ans=add(ans,dp(x-1,is_n&(i==ta[x]),is_m&(j==tb[x]),flg|(i<j)));
        }
    }
    return f[x][is_n][is_m][flg]=ans;
}
```

中国剩余定理

- “有物不知其数，三三数之剩二，~~五五数之剩三~~，七七数之剩二。问物几何？” ——《孙子算经》
- 解决这样的问题：求解如下所示的线性同余方程组。其中模数mi两两互质。

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

$x \equiv 2 \mod 3$

$x \equiv 3 \mod 5$

$x \equiv 2 \mod 2$

$x \equiv 2 \mod 3$

中国剩余定理

M是m的倍数

$(x\%M)\%m = x\%m$

- 结论：令$M=\prod_{i=1}^{n} m_i$。只要mi两两互质，那么对于任意的a1…an，在[0,M)内有一个解。
- 解的给出是构造性的：令$M_i=\prod_{j!=i} m_j = M/m_i$。令$t_i$为$M_i$在模$\mathrm{m}_i$下的逆元。即$M_i t_i \equiv 1 \; mod \; m_i$.（*正是求逆元的过程用到了“两两互质”*）
- 令$x=(\sum_{i=1}^{n} a_i t_i M_i) \; mod \; M$. 我们可以检验所有的线性方程都被满足。

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

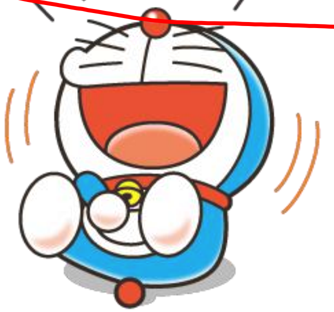
$(a_i t_i M_i)\%$

$x \bmod m_i = ?$

$x = a_i t_i M_i$

当$i!=j$，$a_i t_i M_i \bmod m_j = 0$

当$i=j$ $a_i t_i M_i \bmod m_i$



中国剩余定理

$x \leftrightarrow (x\%m_1, x\%m_2 \ldots x\%m_n)$

- 中国剩余定理最重要的地方可能不在于这个解法，而在于它告诉我们这样一件事：
- 对于两两互质的m_1,m_2,m_3…m_n，$[0, m_1 m_2 \ldots m_n)$这M $= \prod_{i=1}^{n} m_i$个数，和$([0, m_1), [0, m_2), \ldots, [0, m_n))$ 这样的$\prod_{i=1}^{n} m_i$个n元组间，存在一一对应。
- 当我们想求一个在$[0, m_1 m_2 \ldots m_n)$内的数时，我们只需要转而去求它模$m_1, m_2, \ldots m_n$依次是多少。
- ——把M分解为质因数整数幂的成绩，对于每个质因数整数幂作为模数分别求解（这往往比任意合数好求），再用CRT合并，是数论题常见的解法！

$[0,6)$

$([0,1), [0,2))$

$(0,0) - 0$

$(0,1) - 4$

$(0,2) - 2$

$(1,0) \sim 3$

$(1,1) \sim 1$

$(1,2) \sim$

2480 [SDOI2010]古代猪文

$a^k \equiv a^{k \% \varphi(p)}$

题目大意：求$G^{\sum_{d|n} C_n^d} \ mod\ 999911659$

- n, G 1e9级。

$\binom{n}{d} \bmod 999911658$

2480 [SDOI2010]古代猪文

- 注意到999911659为质数，显然使用欧拉定理：$a^b \bmod m = a^{b \bmod \phi(m)} \bmod m.$
- 故只需要求$\sum_{d|n} C(n,d)\ mod\ 999911658.$
- 枚举n的每个因数d分别求解。

2480 [SDOI2010]古代猪文

- 我们不会求1e9级别的组合数模1e9级的大合数。但我们会求1e9级别的组合数模小质数。（Lucas定理）
- 注意到999911658的质因数分解为999911658=2*3*4697*35617。用Lucas定理分别求模2，3，4697，35617的值。然后使用CRT合并即可。

$$\left\{\begin{array}{l}\binom{n}{d} \bmod 2\ \checkmark \\ \qquad\quad\ 3\ \checkmark \\ \binom{n}{d} \bmod 4697\ \checkmark \\ \qquad\quad\ 35617\ \checkmark\end{array}\right. \Longleftrightarrow \binom{n}{d} \bmod 99\ldots$$

谢谢大家！