7.23 简单数论和相关

Quanzhou No.7 Middle School

July 23, 2017

概要

- “数学是科学的皇后，数论是数学的皇后”
- 虽然标题是数论，但显然不够讲一节课，所以会有别的东西
- 再次强调：code 只是参考用，不要直接抄到代码里

关于下午的考试

- 会有一个送分题，不会有反 AK 题。
- 做完后请认真检查，不要错失 AK。

前言

- NOIP 中的数论有很多重要算法。
- 以下出现示例代码的算法都属于需要熟练掌握的范围。

基础概念

- 对于正整数 x, y，如果存在整数 a 使 $x = ay$，则记 $y|x$，y 是 x 的一个约数。
- 对于大于 1 的正整数 x，如果 x 只有 1 和自身两个约数，则 x 是一个素数。
- 否则 x 是一个合数。
- 一般约定 1 即不是素数也不是合数。
- $x \mod k$ 表示 x 对 k 做带余除法的余数。
- $x \equiv y \pmod{k}$，读作 x 和 y 模 k 同余，当且仅当 $k|(x-y)$。
  - 有时候偷懒会写作 x=y (mod k)。

op mod P

- $(a+b) \mod P = a \mod P + b \mod P$
  - or is it?
- 对减法和乘法也成立
- 除法呢?
  - P 不是质数时，除法不一定有意义
  - $1/6 \mod 6 = ???$
    - 这是类似整数除法中 $1/0$ 的情形
  - $4/2 \mod 6$ 是什么鬼? 它等于 2 还是等于 5?

素数的判定

- 必须掌握至少 $O(\sqrt{N})$ 的判定法（直接从 2 开始枚举因子）。

素数密度定理

- 前 N 个正整数中大约有 $N/\ln N$ 个素数。
- 从另外一个角度来说，第 P 个素数大小大约是 $P\ln P$。

- 允许预处理出小范围素数，只枚举素数因子，优化到 $O(\sqrt{N}/\ln N)$。

Miller-Rabin

- 输入整数 N，输入长度是 $\log N$，因此 $O(\sqrt{N})$ 的算法并不是多项式（“有效”）算法。
- 素数判定问题存在有效算法，原版 Miller-Rabin 一轮测试在 $O(\log^3 N)$。
- 听说有人想学……？

质因子分解

- 对素数判定算法进行小修改可以得到质因素分解算法，复杂度类似。
- 目前没有已知的有效分解算法。
- Pollard's Rho 算法
  - 估计复杂度为 $O(n^{1/4})$ 级别，但没有精确估计
  - NOIP 级别不需要掌握……

欧拉函数

- $\phi(N)$ 表示 1~N 中和 N 互质的数字个数。
- $\phi(16) =?$
- $\phi(37) =?$
- $\phi(12) =?$
- 计算一个数的欧拉函数：
  $\phi(N) = N \times (1 - 1/p_1)(1 - 1/p_2) \cdots$
  - p_i 是 N 的素因子，重复不计
  - 复杂度同质因子分解
- 对于素数 p，$\phi(p) = p - 1$

欧拉定理

若 a 和 p 互质，则 $a^{\phi(p)} = 1(\mod p)$。
特别地，若 p 是质数，$a^{p-1} = 1(\mod p)$（费马小定理）。
可以用群论知识证明。

GCD & LCM

- 对于两个数 x 和 y，如果 $d|x$，$d|y$，称 d 是 x 和 y 的公约数。
- 1 是所有数的公约数。
- x 和 y 的最大公约数记为 $gcd(x, y)$。
- x 和 y 互质 $\leftrightarrow gcd(x, y) = 1$。
- 类似的，如果 $x|d$，$y|d$，称 d 是 x 和 y 的公倍数。
- x 和 y 的最小公倍数记为 $lcm(x, y)$。

唯一分解定理

- 大家都知道一个数可以写成若干个素数幂次的乘积。
- $N = p_1^{k_1} p_2^{k_2} \cdots p_m^{k_m}$。
- $gcd(A, B) = p_i^{k_i}$，其中 $k_i = \min(kA_i, kB_i)$
- $lcm(A, B) = p_i^{l_i}$，其中 $l_i = \max(kA_i, kB_i)$
- 计算一个数的欧拉函数：
  $\phi(N) = N \times (1 - 1/p_1)(1 - 1/p_2) \cdots$
  - 嘴炮：每出现一个质因子，$1/p$ 的数被筛除，且过程互相独立
  - 严格证明需要等到 CRT

辗转相除

- $gcd(x, y) = gcd(x \mod y, y)$，仅当 $x \mod y$ 不为 0 时
- 利用这个公式可以不断缩小问题，直至 $x \mod y = 0$ 停止
- 务必保证可以默写算法
- 求 lcm：利用 $gcd(x, y)lcm(x, y) = xy$
  - 证明？

Code

- 代码只是参考，不保证可以直接塞到题目里用

```
int gcd(int x, int y) {
  if ((x % y) == 0)
    return y;
  else
    return gcd(y, x % y);
}

int NR_gcd(int x, int y) {
  while (y != 0) {
    int r = y;
    y = x % y;
    x = r;
  }
  return x;
}
```

Extended GCD

- 给定 a, b，求一组整数 x, y，使得 $ax + by = gcd(a, b)$。
  - 可以证明这样的整数存在。
  - 例如当 $a = 3, b = 5$ 时，一组解为 $x = 2, y = -1$。
- 假设已知 $a'x' + b'y' = gcd(a, b)$ 的一组解 (x', y')，其中 $a' = b, b' = a \mod b = a - kb$。
  - 这是原来 gcd 算法的一步：$gcd(a, b) = gcd(a', b')$
  - 也就是说 $x = ky + x'$ 是 x 对 y 的带余除法，$k = x/y$。
- 那么 $bx' + (a - kb)y' = ay' + b(x' - ky')$，故 $x = y', y = x - ky'$ 是原问题的一组解。
- 执行算法时，实际上先算出了 $gcd(a, b)$，然后套上述公式套回去。

Code

注意 x 和 y 的存储方式。

```
int exgcd(int a, int b, int &x, int &y) {
  if ((a % b) == 0) {
    x = 1; y = 0;
    return b;
  } else {
    int ret = exgcd(b, a % b);
    int t = x;
    x = y;
    y = t - (a/b)*y;
    return ret;
}
```

几个小问题

以下不加说明则均为整数解。

1. 求 $ax + by = c$ 的一个解。
2. 求 $ax + by = gcd(a, b)$ 的所有解。
3. 求 $ax \equiv 1(\mod b)$ 的一个解。
   - 这个就是所谓的乘法逆元。
4. 求 $ax + by + cz = d$ 的一个解。

Solution

1. 假设 $c = k \times gcd(a, b)$，则将 exgcd 的结果乘上 k 即可。
2. 用 exgcd 得到一组解 x_0, y_0 之后，所有解形式为 $x = x_0 + k(b/gcd(a, b)), y = y_0 - k(a/gcd(a, b))$
3. 用 exgcd 解 $ax + by = 1$。
   - 也可以使用欧拉定理，算 $a^{\phi(b)-1}$。如果 b 是质数，就是 a^{b-2}。
4. 先解出 $ax + by = gcd(a, b)$ 的一组解，然后解 $gcd(a, b) \times x + cy = d$，最后反向带入。

素数筛

- 求 1~N 中哪些数是素数。
- 可以对每个数单独判断，时间复杂度是 $O(N\sqrt{N})$ 的（假设你不会 Miller-Rabin）。
- 换一个角度，只要排除出 2 以外 2 的倍数，3 以外 3 的倍数，……，直到 $\sqrt{N}$ 以上为止。
- 这样复杂度比较不好估计，但是上限是 $O(N \log N)$。
- 务必熟练掌握的算法 +1。

Code

- 实际上会内嵌在别的函数里，不会把一个整型数组传来传去。

```
void work(int n, int primes[]) {
  int m;
  bool check[maxN]; // Assume starts all false
  for (int i = 2; i < N; i++)
    if (!check[i]) {
      primes[++m] = i;
      for (int k = i * i; k < N; k += i) check[k]
    = true;
  }
}
```

CRT

- CRT 是中国剩余定理的简称，不是那个显示器……
- 给定 n 个同余方程 $x = r_i (\mod a_i)$，求 x 的一个解。
- 固定 a_i，给定很多组 r_i，每个询问处理时间 $O(n)$。
- 求一组 x_i，每个 x_i 满足 $x_i \mod a_j = 1 (i = j)$
  - 解方程 $A_i x = 1 (\mod a_i)$ 即可，A_i 是 $a_{i \neq j}$ 乘起来的值。
- 每组 r_i 的一个可行解就是 $\sum r_i x_i$。
- 常见应用：题目要求模的数不是素数，但是一些素数之积，那么先用各个素数取模，最后用 CRT 合并。

CRT

- 事实上，CRT 讲的是这么一件事情：

真·中国剩余定理

- 若 a_i 两两互质，$A = \prod a_i$。
- 那么对于任意 $0 \le d < A$，$x = d \pmod A$ 等价于 $x = (d \bmod a_i) \pmod{a_i}$ 对所有 i 成立。
- 另外，对任意 d，这一公式都有解。

- 于是我们回头可以证明欧拉函数了。
- 取 N 的不重复质因数分解。
- 由 CRT，每个小于 N 的非负整数 k 都可以一一对应到 $\{k \bmod p_i^{d_i}\}$ 上，所以我们考虑有多少个这样的集合。
- k 和 N 互质，当且仅当 $p \nmid k \bmod p_i^{d_i}$，有 $(p-1)p_i^{d_i-1}$ 种取法。
- 最后用一次乘法原理就结束了。

Code

- 事实上理解了并不需要默写……
- 另外实际实现时需要注意溢出问题（包括上下溢出）
- 代码只是参考，不保证可以直接塞到题目里用

```
//Note: x will be filled with correct values
int work(int N, int a[], int r[], int x[]) {
  int M = 1, tmp, ans = 0;
  for (int i = 0; i < N; i++) M *= a[i];
  for (int i = 0; i < N; i++)
    exgcd(M / a[i], a[i], x[i], tmp);
  for (int i = 0; i < N; i++)
    ans = (ans + M / a[i] * x[i] * r[i]) % M;
  return ans;
}
```

CRT 不适用的情形

- 在 a_i 不互质时怎么做？
- 对于 $n = 2$ 的情形：$x = r_0 (\mod a_0), x = r_1 (\mod a_1)$
- 此时 $x = r_0 + k_0 a_0 = r_1 + k_1 a_1$，即 $a_0 k_0 - a_1 k_1 = r_1 - r_0$
- 解出 x，合并方程 $x = x_0 (\mod lcm(a_0, a_1))$
- 两两合并即可
- 复杂度怎么算？$O(n \log a_i)$？$O(n^2 \log a_i)$？

模 P 逆元的快速预处理

- 这是一个偶尔可以用到的 trick
- 核心思想：
  - $inv[P \mod i] * (P \mod i) = 1$
  - $(inv[P \mod i]) * (i * (-P/i)) = 1$
  - $inv[i] = (P - P/i) * inv[P \mod i]$
  - 具体推导略，前面多出一项 P 是为了防止负数取模出事
- 代码只是参考，不保证可以直接塞到题目里用

```
int[] work(int P, int inv[]) {
  inv[1] = 1;
  for (int i = 2; i < P; i++)
    inv[i] = (P - P / i) * inv[P % i] % P;
}
```

线性素数筛

- 另外一个有用的 trick
- 在使用一般素数筛时，一个合数会被筛除多次
  - 比如 6 在 2 确定为素数和 3 确定为素数时，都会被筛一次
- 保证每个合数只被自己最小的质因子筛除就可以了
- 脑筋急转弯时间：给定一个数 M'，和一个质数 p。如何判断 p 是 $M = pM'$ 的最小质因子？

Code

- 显然如果 M' 没有小于 p 的因子就可以停下了。
- 我们修改一下原有的筛法，主循环枚举 M'。
  - 在次循环枚举 p，如果 $p|M'$，直接退出。
- 代码只是参考，不保证可以直接塞到题目里用

```
int[] work(int n, int prime[]) {
  int m;
  bool check[maxN]; // Assume starts all false
  for (int i = 2; i < N; i++) {
    if (!check[i]) primes[++m] = i;
    for (int k = 0; i * prime[k] < N; k++) {
      check[i * prime[k]] = true;
      if ((i % prime[k]) == 0) break;
    }
  }
}
```

适用于大数的辗转相除法

- 辗转相除法很好写。但如果输入的数很大（高精度或者类似），怎么保证不写炸？
  - 注意这时候取模操作非常麻烦
  - 另一方面，在计算机内部，除以 2 这个操作等价于右移 1 位
  - 只用右移和加减法完成辗转相除？
  - 使用 $gcd(x, y) = gcd(x, y - x)$ 是可能退化到 $O(n)$ 的

Code(Stein 算法)

- 代码只是参考，不保证可以直接塞到题目里用

```
bigint gcd(bigint x, bigint y) {
  // Make sure x > y
  if (x <= y) swap(x, y);
  if (y == 0) return x;
  if ((x % 2 == 0) && (y % 2 == 0))
    return gcd(x >> 1, y >> 1) << 1;
  if (x % 2 == 0) return gcd(x >> 1, y);
  if (y % 2 == 0) return gcd(x, y >> 1);
  return gcd(y, x - y); // !
}
```

- 为什么这个算法不退化?
  - 每两步至少有一步不会执行到最后一句

前言

- NOIP 级别的数论应用不多，简单看几个题目
- 比较神奇的东西最后有时间会讲一些

Problem 5

- 给定 a_0, a_1, b_0, b_1，求满足 $gcd(a_0, x) = a_1, lcm(b_0 x) = b_1$ 的 x 个数
- 多组数据，输入范围在 32 位整型内

Solution

- 每个质因子是独立计算的
- 每个质因子的子问题相当于 $min(k_{a0}, k_x) = k_{a1}, max(k_{b0}, k_x) = k_{b1}$，求 k_x 的取法种数
- 分几种情况分别讨论即可

Problem 6

- 给定 N，求 $x^2 = 1 (\mod N)$ 的解个数
- $N \leq 10^{10}$

Solution

- $N|(x-1)(x+1)$
- $N|gcd(x-1,N)gcd(x+1,N)$
- $gcd(x-1,N)$ 和 $gcd(x+1,N)$ 至少有一个超过 $\sqrt{N}$
- 暴力枚举这个数的所有倍数算即可

Problem 7

- 定义 $f(i)$ 为 i 的约数个数
- 输出 $f(1) + f(2) + \cdots + f(N)$
- Easy: $N \sim 10^5$
- Hard: $N \sim 10^{10}$

Solution

- 转化为计数问题：$\sum_{i=1}^{N}\sum_{j=1}^{i}1(j|i)$
- 常见 trick：调换求和顺序
- $\sum_{i=1}^{N}\sum_{j=1}^{i}f(i,j)=\sum_{j=1}^{N}\sum_{i=j}^{N}f(i,j)$
- 到本题中，我们发现问题变成了求 $N+(N/2)+(N/3)+\cdots+(N/N)$，除法为整除
- Easy：直接上
- Hard：$i<\sqrt{N}$ 时直接算，$i>\sqrt{N}$ 时取值只有 $\sqrt{N}$ 种，分段统计

Problem 8

- 输入 N，问满足 $1 \leq x, y \leq N, (x, y) = 1$ 有多少对。
  - Easy: $O(N\sqrt{N})$。Medium: $O(N \log N)$。

Solution

- 我们注意到 x, y 对称，所以可以只考虑 $x < y$ 的情况。
- 这一半的答案就是 $\sum_{y=1}^{N} \phi(y)$。
- 总的答案是 $2 * Ans' - 1$（考虑 $(1, 1)$）。
- Easy：直接算。
- Medium：筛法期间对每个合数是可以（顺便）得到它的一个因子的。所以可以 $O(\log N)$ 分解质因数。

前言

- 这一部分包括一些组合数学的内容：组合数，递推式等等。
  - 重点是带大家理解矩阵乘法到底是怎么来的。
  - 如果你已经了解这些内容，也请不要私下讨论影响课堂秩序。
- 之后有时间会讲一些最基础的反演。

组合数

- $n! = 1 \times 2 \times 3 \times 4 \cdots \times n$。
  - 冷知识：阶乘是可以扩充到实数上的，比如 $0.5! = \sqrt{\pi}/2$。
- $C(n, m) = n!/(m!(n-m)!)$，称为组合数。
  - 其意义是在 n 个元素中不计顺序选取 m 个的方案数。

几个小定理

- 杨辉三角：$C(n,i) = C(n-1,i) + C(n-1,i-1)$
- 二项式定理：$(a+b)^n = \sum C(n,i)a^i b^{n-i}$
  - $C(n,0) + C(n,1) + \cdots + C(n,n) = 2^n$
  - $C(n,0) - C(n,1) + C(n,2) - \cdots + (-1)^n C(n,n) = 0$
- 隔板法
  - $N = n_1 + n_2 + \cdots + n_m$ 有 $C(n+m-1,n)$ 个非负整数解。
  - 有多少个正整数解？

著名问题

- 假设 $C(n,m) = p^a \cdot b$，且 $p \nmid b$，要求 a 和 $b \mod p$ 的值。
  - $p \leq 10^5$。

Solution(?)

威尔逊定理

- 若 P 是素数，$(P-1)! = -1(\mod P)$。

- 我们发现完全可以把问题放到阶乘上，于是考虑 $n!$。
- 把 $1 \sim n$ 每 p 位拆一段。先提出 p 的倍数，中间类似 $(kp+1)(kp+2)\cdots(kp+p-1)$ 的连乘积由威尔逊定理模 p 为-1。
  - 剩下最后 $n \mod p$ 位无法用威尔逊定理解决，但 p 不大可以暴力。
  - p 的倍数连乘为 $p^{n/p}(n/p)!$，故递归处理即可。
- 进一步扩展这个思路可以得到 Lucas 定理（证明略去）。
  - 若 A 的 p 进制位表示为 $a_k a_{k-1}\cdots a_0$，B 的表示为 $b_k b_{k-1}\cdots b_0$
  - 那么 $C(A,B) \mod P = C(a_k,b_k)C(a_{k-1},b_{k-1})\cdots C(a_0,b_0)$

线性递推式

- Fibonacci 数列
  - $F_0 = 1, F_1 = 1, F_n = F_{n-1} + F_{n-2}$
  - 求 $F_N \mod (10^9 + 7)$
  - $N \leq 10^{100}$

向量和矩阵

- $x = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$ 是一个向量，长度为 4，元素为 $1, 2, 3, 4$，第 i 个元素用 x_i 表示。
- 对两个长度相等的向量，定义它们的内积为对应元素乘起来之和。
  - 例如，$x^T x = 1^2 + 2^2 + 3^2 + 4^2 = 30$。
- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ 是一个矩阵，大小为 2x2，第 i 行第 j 个元素用 A_{ij} 表示。
- 在 NOIP 范畴内，矩阵是用来表示向量变换的工具。

向量和矩阵

- 一个矩阵（大小为 $m \times n$）作用在一个向量（n 维）上，得到的结果是一个 m 维向量。
- 结果的第 i 个元素是矩阵的第 i 行和向量的内积。
  - recall：对两个长度相等的向量，定义它们的内积为对应元素乘起来之和。
- 试着理解一下下面的等式。
- $$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a+b \\ a \end{pmatrix}$$
- 如果 $a = F_{i+1}, b = F_i$，以上等式可以这么理解：
- $$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{i+1} \\ F_i \end{pmatrix} = \begin{pmatrix} F_{i+2} \\ F_{i+1} \end{pmatrix}$$
- 于是我们重新发明了求 F_{i+2} 的方法。

造轮子：矩阵乘法

- $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{i+1} \\ F_i \end{pmatrix} = \begin{pmatrix} F_{i+2} \\ F_{i+1} \end{pmatrix}$
- 我们用 A 表示上面那个矩阵，用 f_i 表示向量 $(F_{i+1}, F_i)^T$。
- 现在我们想让右边成为 $\begin{pmatrix} F_{i+3} \\ F_{i+2} \end{pmatrix}$，也就是 f_{i+2}。
- 事实上，$Af_i = f_{i+1}$，$Af_{i+1} = f_{i+2}$，所以自然地应该有 $A^2 f_i = f_{i+2}$。
- 现在来定义矩阵乘法：
  - 两个矩阵 A 和 B 相乘之后得到的矩阵作用在一个向量上，应当和先作用 B，再作用 A 得到完全相同的效果。
  - 为什么是先 B 后 A 是因为接近向量的矩阵先作用。
  - 写成数学化的形式就是 $(AB)f = A(Bf)$。

造轮子：矩阵乘法

- 两个矩阵 A 和 B 相乘之后得到的矩阵作用在一个向量上，应当和先作用 B，再作用 A 得到完全相同的效果。
- 假设矩阵 A 的元素是 A_{ij}，矩阵 B 的元素是 B_{ij}。
- 那么 $(Bf)_i = \sum_j B_{ij} f_j$。
- 于是 $A(Bf)_i = \sum_j A_{ij}(\sum_k B_{jk} f_k) = (AB)f_i$。
- 另一方面，$(AB)f_i = \sum_k (AB)_{ik} f_k$。因为所有 f_j 都是变量，所以两式恒等，只有当每一项系数相等。
- 考虑 $A(Bf)_i$ 中 f_k 的系数，可以化简一下得到 $\sum_j A_{ij} B_{jk}$，根据前面的推导，这就是矩阵乘法的公式：
  - $(AB)_{i,j} = \sum_k A_{ik} B_{kj}$
  - 复杂度 $O(n^3)$，假设 A 和 B 都是方阵

矩阵快速幂

- $ABC = (AB)C = A(BC)$，即矩阵乘法有结合律。
  - 根据我们对矩阵乘法的定义不难理解。
- 现在可以正式解决一开始提到的问题了。
- 定义 $f_1 = (1, 0)^T = (F_1, F_0)^T$。
- 我们知道 $A^n f_1 = (F_{n+1}, F_n)^T$，于是问题变成求 A^n。
- 因为矩阵乘法有结合律，可以用快速幂思想求解。
  - $A^{2n} = A^n A^n$，$A^{2n+1} = A^{2n} A$
- 最后，向量可以看成有一维长度为 1 的矩阵，因此你也可以从矩阵开始推倒这套东西。

Code

- 需要特别小心指针重合的情况。
- *matrixMultiply*$(a[0], a[1], a[2])$?

```
void matrix_multiply(int n, int a[][], int b[][],
    int c[][]) {
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) {
        c[i][j] = 0;
      for (int k = 0; k < n; k++)
        c[i][j] += a[i][k] * b[k][j];
    }
}
```

一些类似的问题

- 对以下每个问题，尝试用矩阵转移的思路写出对应的矩阵和转移向量。
- 初始值无所谓，但不能让问题变得 trivial（比如第一问不能假设初始值都是 0）。

1. $f_i = f_{i-1} + f_{i-2} + f_{i-3} + 1$
2. $f_i = f_{i-1} + g_{i-2}, g_i = f_{i-1} + g_{i-2}$
3. 求 $\sum_{i=1}^{n} f_i$，f_i 是一个矩阵递推能搞定的递推式
4. 求 $\sum_{i=1}^{n} f_{2i}$
5. $f_i = f_{i-1} + f_{i-2} + i$
6. $f_i = f_{i-1} + i^2$

Solution

1. 转移向量 $v_i = (f_i, f_{i-1}, f_{i-2}, 1)$，最后一维恒定不变
2. 转移向量 $(f_i, f_{i-1}, g_i, g_{i-1})$
3. 转移向量 $(f_i, \cdots, \sum f_i)$
4. 求一个转移两步的矩阵再套 3
5. 转移向量 $(f_i, f_{i-1}, 1, i)$
6. 转移向量 $(f_i, f_{i-1}, 1, i, i^2)$
   - $(i+1)^2 = i^2 + 2i + 1$

几个相关问题

- 矩阵乘法本身就是一个算法，所以单独考没什么好考的。
- 旋转平移这些操作都可以看作是对坐标的线性变换。给定 M 个平面上的这种操作，给 N 个点，问每个点经过同样操作后的坐标。
  - 转移向量是 $(x, y, 1)$。复杂度 $O(M+N)$
- 给定图 G，节点数很少 (~50)，问从 A 到 B 走恰好 k 步的方案数。需要 $O(\log K \cdot \text{poly}(N))$。
  - 向量 f_x^i 表示从 A 走 i 步的方案数，可以发现转移矩阵就是图的邻接矩阵
- 用 1x2 的多米诺骨牌覆盖 $n \times m$ 的棋盘，求方案数，n 很小但 m 很大
  - 状态压缩 dp，然后写成向量形式，转移就是一列到下一列的系数

相关问题 2

- 字符串相关
- 给定一个长度 20 的串，问所有长度 10^20 的串里有多少不包含给定串
  - 建 kmp 数组，模拟从前往后匹配的过程，转移向量 f_i 表示当前走到状态 i 有多少种可能
- 把上面换成一堆串，但总长度不会太长
  - 改成在 AC 自动机上跑
- 还有一些不好分类的脑洞题，比如下一页这个

Problem X

- 给你一个 n 位的数字，可以保持顺序的前提下随便切割
- 每种切割的分数是 Fib[切割各部分之和]
- 求所有切割分数之和
- 啥玩意啊.jpg
- 咋回事啊.png

Solution

- 你要求的是类似 $F^{1+2+3} + F^{12+3} + F^{1+23} + \cdots$ 的玩意
- 用 $dp[i]$ 表示前 i 位这些矩阵之和
- 枚举最后一段长度，从 j 开始，则这一部分贡献是 $F^{j+1..i} dp[j]$
- $O(n^2)$（大概）

概述

- 这东西几年前是严格限定在省选以上的。
- 但几年前网络流也是省选以上的，然而你们后天就要学习了。
- 而且 dalao 们可能很喜欢 (?)。
- 我们先大概从字面意义上理解一下反演这个词的意思，然后看几个经典反演。
- 例题都是省选级别，所以 tan90 了。

引入

- 如果你还记得的话：
- $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{i+1} \\ F_i \end{pmatrix} = \begin{pmatrix} F_{i+2} \\ F_{i+1} \end{pmatrix}$
- 假如我想搞事情，给了你右边的向量，怎么求左边的？
- $F_{i+2} = F_i + F_{i+1}$，则 $F_i = F_{i+2} - F_{i+1}$，事实上这可以引出下面的矩阵递推式
- $\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} F_{i+1} \\ F_i \end{pmatrix} = \begin{pmatrix} F_i \\ F_{i-1} \end{pmatrix}$
- 这样，你就完成了一次反演。
- 从线性变换的角度来说，这两个矩阵协同作用的效果是 nothing。
- $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2$，也称作单位矩阵。
- “反演就是求矩阵逆”

其他反演

- 作为一个更一般的 idea，反演不仅仅用于数论中。
  - 计算几何的反演变换可以把直线变成圆，把圆变成直线。
  - 物理中反演是研究对称性的一种手段。
  - 在一些工程科学里也有这个名词。
  - (dota 里你也可以反演)
- 我们接下来看两个反演。
  - 二项式反演
  - Mobius 反演

二项式反演

- 这类反演的一般形式:
- 已知 $f(n) = \sum A_{n,m} g(m)$，求 $B_{m,n}$ 使得 $g(m) = \sum B_{m,n} f(n)$
- 在这个问题里，$A_{n,m} = C(n,m)$。
- 事实上，计算一下可以发现 $B_{n,m} = (-1)^{n+m} C(n,m)$。
- 证明反演成立相当于证明 $AB = I_n$
  - 主对角线上，$n = m$ 以外地区 A_{nm} 或者 B_{mn} 为 0
  - 其他地方，要证明 $0 = \sum_k (-1)^k C(n,k) C(k,m)$
  - 组合数的肮脏 trick: $C(n,k)C(k,m) = C(n,m)C(n-m,k-m)$
  - 然后加回去提出 $C(n,m)$，变成了 $\sum_k (-1)^k C(n-m,k-m) = (1-1)^{n-m} = 0$

Mobius 反演

- 假设有 $F(n) = \sum_{d|n} f(d)$，我们想知道如何用 $F(n)$ 表示 $f(d)$。
- 试着推倒一下前几项？

```
F(1) = f(1)                          f(1) = F(1)
F(2) = f(1) + f(2)                   f(2) = F(2) - F(1)
F(3) = f(1) + f(3)                   f(3) = F(3) - F(1)
F(4) = f(1) + f(2) + f(4)            f(4) = F(4) - F(2) // !!
F(5) = f(1) + f(5)                   f(5) = F(5) - F(1)
F(6) = f(1) + f(2) + f(3) + f(6)     f(6) = F(6) - F(3) - F(2) + F(1)
F(7) = f(1) + f(7)                   f(7) = F(7) - F(1)
F(8) = f(1) + f(2) + f(4) + f(8)     f(8) = F(8) - F(4)
```

Mobius 函数

- 我们先不进行推导给出 Mobius 函数。
- 若 x 有一个平方因子，则 $\mu(x) = 0$。否则 $\mu(x) = (-1)^d$，d 是 x 的素因子个数。
- 定理：$\sum_{d|n} \mu(n) = 1(n = 1)$。
  - 取 n 的最大无平方因子，在该因子上计算即可。
  - 思考：如何在筛法基础上在 $O(n)$ 时间内求 $\mu(n)$?

Mobius 反演

- 若 $F(n) = \sum_{d|n} f(d)$，那么有 $f(n) = \sum_{d|n} \mu(n/d)F(d)$。
- 证明需要注意到 $(AB)_{ij}$ 满足上述定理形式，只有 $i = j$ 时为 0。

Problem 8

- 输入 N，问满足 $1 \leq x, y \leq N, (x, y) = 1$ 有多少对。
- Easy: $O(N\sqrt{N})$。Medium: $O(N \log N)$。
- Hard: $O(N)$。

Solution

- 我们来尝试用 Mobius 反演解决这个问题。

Mobius 反演

- 若 $F(n) = \sum_{d|n} f(d)$，那么有 $f(n) = \sum_{d|n} \mu(n/d)F(d)$。
- 若 $F(n) = \sum_{n|d} f(d)$，那么有 $f(n) = \sum_{n|d} \mu(d/n)F(d)$。
  - 从矩阵逆的角度来说，我们就是把 i 和 j 的位置换了一下……

- 用 $f(d)$ 表示满足 $(x, y) = d$ 的对数。
- 利用第二个反演公式，我们构造 $F(d) = \sum_{n|d} f(d) = (N/d)^2$。
  - gcd 是 d 的倍数等价于两个是都是 d 的倍数。
- 反演就得到了 $f(1) = \sum_n \mu(n)(N/d)^2$，这个式子可以 $O(n)$ 算出来了。

Extension

- 这个题显然是可以接着扩张的。。
- 比如把 gcd 改成 lcm。。
  - 然后再多搞搞，你就会得到神题 JZPKIL
- Mobius 函数本身也有特别的性质：若 $(p, q) = 1$，$\mu(pq) = \mu(p)\mu(q)$。满足这类性质的函数成为积性函数。
  - 然后就问在台下的 dalao 吧，我啥都不懂。

高斯消元

- 解方程 $Ax = y$。
- $x = y/A$?
- A 是一个矩阵，x 和 y 是对应长度的向量，我们先假设这是个方阵
- 常规解读：给定 n 个未知元的 m 个线性方程，求一组解
  - $\sum A_{mi}x_i = b_m \leftrightarrow Ax = B$
- 类似的，可以用向量内积的形式写：$A_i^T x = b_i$
- 所以各行可以随意调换
  - 换 A 的列会发生什么？

解空间

- $\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a+b \\ a+b \end{pmatrix}$
- 如果右边是 $(0,1)^T$?

等价性

- 如果 $A_i^T x = b_i$, $A_j^T x = b_j$
- 那么：
  - (1) $kA_i^T x = kb_i$
  - (2) $(A_i + A_j)^T x = (b_i + b_j)$
- 结合以上两条：
  - $(A_i + kA_j)^T x = (b_i + kb_j)$

三角形式

- 如果方阵 A 满足 $\forall i, j, i > j \to A_{ij} = 0$，称 A 是上三角阵
- 这时候 $Ax = b$ 很好解
- 高斯消元：$Ax = b$ 等价于 $A'x = b'$，且 A' 是上三角阵

造轮子：高斯消元

- 由于我们在做行变换，先试着保证 A 的最后一列为 0。
- 找到一组 $A_i^T x = b_i$，使得 $A_{in} \neq 0$
- 对其他的 $A_j^T x = b$：$(A_j + kA_i)^T x = (b_j + kb_i)$
  - $k = -A_{jn}/A_{in}$
- 回代：$A_{ii}x_i = b_i - \sum_{j>i} A_{ij}x_j$

嘴炮 Code(?)

- 注意：这个 code 几乎一定不 work

```
void gaussian(int n, double A[], double x[],
    double y[]) {
  for (int d = n - 1; d > 0; d--) {
    // Assume A[d][d] != 0
    for (int i = d + 1; i < n; i++)
      A[i] = A[i] - (A[i][d]/A[d][d]) * A[d],
      b[i] = b[i] - (A[i][d]/A[d][d]) * b[d]
  }
  for (int d = n - 1; d > 0; d--) {
    double t = b[d];
    for (int i = d + 1; i < n; i++)
      t -= A[d][i] * x[i];
    x[d] = t / A[d][d];
  }
}
```

高斯消元

- 还没有解决的问题：
  - $A_{dd} = 0$ 怎么办？找一组换上
    - 所有 $A_{nd} = 0$ 怎么办？
  - 浮点数误差问题？
    - 根据数值分析原理，取绝对值大的做除数
    - 针对模 P 剩余类的高斯消元
- 一般情况的高斯消元解取决于 A 的 rank 和 $[A, b]$ 的 rank
  - $rank(A) < rank([A, b])$：无解
  - 否则有 $n - rank(A)$ 个自由元，固定这些自由元后有唯一解
  - 讲讲线性代数??

前言

- ……好像还有一些时间？
- 我们接下来讲抽象代数。
- 这些东西应该不会让你写出更多算法，但是能帮助你理解一些东西。
- 有人想学 Miller-Rabin 和 Pho-Rollard？
- 自己 google……

群

- 一个群 G 是一个集合 S，以及定义在 S 上的二元运算 $+$ 组成的结构。
  - $+$ 运算满足：封闭、结合、有幺元（0 元），有逆元
- 比如，G 是模 p 同余群，那么 $(G, +)$ 是一个群，但是 $(G, *)$ 就不是。
  - 有的元素没有逆元，但我们可以不包含模 p 同余的每个元素，这样可以得到一个群。
- 事实上，所有同样大小的矩阵，定义在矩阵加法上也是一个群（比较显然）。
- 不太显然的事情：所有可逆方阵，定义在矩阵乘法上，是一个群。
- 另外，模 P 群都是交换群（阿贝尔群）。模 P 加法群是最简单的群，记为 $Z\backslash Zp$，它的性质也很简单。

乘法群

- 但是模 P 的乘法群就比较麻烦了。
- 首先，我们必须指定模 P 乘法群的元素为和 P 互质的剩余类，否则它们将没有逆元。
  - 这样，乘法群的大小恰好是欧拉函数 $\phi(P)$。这是欧拉定理证明的开始。
- 我们再引入一个概念：直积。两个群的直积结果是一个群，元素是两个群的笛卡尔积，运算是对两部分分别运算。
  - C_3 和 C_5 的直积是 C_{15}（中国剩余定理）。
  - 但 C_2 和 C_2 的直积就不知道是什么了（Klein 四元群，四元数群）。
  - C_2 和 C_4 的直积是什么？

模 P 乘法群

- 给定一个 P，你可以造出模 P 的乘法群。但是和加法群不同，乘法群不一定是循环的。
- 首先，如果 P 是素数，那么 $(Z/p)*$ 循环（这有很简单的证明）。
- 其次，如果 $P = p^k$，p 是一个素数，那么这个群也循环。（我忘记怎么证了。）
- 最后，你会发现，除了以上提到的，以及 $2p^k$ 和 $n = 1, 2, 4$ 以外，其他的模 P 乘法群居然都不是循环的。
- 那他们可以是什么样的呢?
- 注意到这个群必须是交换的，而且有限，因此体现为若干循环群的直积。
- 小测验：算一下 $(Z/12)*$ 的形态。

From Wikipedia

$(\mathbb{Z}/n\mathbb{Z})^\times$ 的群结构

n	$(\mathbb{Z}/n\mathbb{Z})^\times$	$\varphi(n)$	$\lambda(n)$	生成元
1	C_1	1	1	0
2	C_1	1	1	1
3	C_2	2	2	2
4	C_2	2	2	3
5	C_4	4	4	2
6	C_2	2	2	5
7	C_6	6	6	3
8	$C_2 \times C_2$	4	2	7, 3
9	C_6	6	6	2
10	C_4	4	4	3
11	C_{10}	10	10	2
12	$C_2 \times C_2$	4	2	5, 7
13	C_{12}	12	12	2
14	C_6	6	6	3
15	$C_2 \times C_4$	8	4	14, 2
16	$C_2 \times C_4$	8	4	15, 3
17	C_{16}	16	16	3
18	C_6	6	6	5
19	C_{18}	18	18	2
20	$C_2 \times C_4$	8	4	19, 3
21	$C_2 \times C_6$	12	6	13, 5
22	C_{10}	10	10	2
23	C_{22}	22	22	5
24	$C_2 \times C_4$	8	4	23, 3
25	C_{20}	20	20	2
26	C_{12}	12	12	5
27	C_{18}	18	18	2
28	$C_2 \times C_6$	12	6	27, 13
29	C_{28}	28	28	2
30	$C_2 \times C_4$	8	4	29, 7
31	C_{30}	30	30	3
32	$C_2 \times C_8$	16	8	31, 3
33	$C_2 \times C_{10}$	20	10	10, 2
34	C_{16}	16	16	3
35	$C_2 \times C_{12}$	24	12	6, 2
36	$C_2 \times C_6$	12	6	19, 5
37	C_{36}	36	36	2
38	C_{18}	18	18	3
39	$C_2 \times C_{12}$	24	12	38, 2
40	$C_2 \times C_2 \times C_4$	16	4	39, 11, 3
41	C_{40}	40	40	6
42	$C_2 \times C_6$	12	6	13, 5
43	C_{42}	42	42	3
44	$C_2 \times C_{10}$	20	10	43, 3
45	$C_2 \times C_{12}$	24	12	44, 2
46	C_{22}	22	22	5
47	C_{46}	46	46	5
48	$C_2 \times C_2 \times C_4$	16	4	47, 7, 5
49	C_{42}	42	42	3
50	C_{20}	20	20	3
51	$C_2 \times C_8$	16	8	7, 3
52	$C_2 \times C_{12}$	24	12	51, 5
53	C_{52}	52	52	2
54	C_{18}	18	18	5
55	$C_2 \times C_{20}$	40	20	2, 21
56	$C_2 \times C_{12}$	24	12	29, 3
57	C_{18}	18	18	2
58	C_{28}	28	28	3
59	C_{58}	58	58	2
60	$C_2 \times C_4$	8	4	31, 41
61	C_{60}	60	60	2
62	C_{30}	30	30	3
63	$C_6 \times C_6$	36	6	2, 5
64	$C_2 \times C_{16}$	32	16	3, 63
65	$C_4 \times C_{12}$	48	12	2, 12
66	$C_2 \times C_{10}$	20	10	5, 7
67	C_{66}	66	66	2
68	$C_2 \times C_{16}$	32	16	3, 67
69	$C_2 \times C_{22}$	44	22	2, 68
70	$C_2 \times C_{12}$	24	12	3, 11
71	C_{70}	70	70	7
72	$C_2 \times C_2 \times C_6$	24	12	5, 7, 11
73	C_{72}	72	72	5
74	C_{36}	36	36	5
75	$C_2 \times C_{20}$	40	20	2, 74
76	$C_2 \times C_{18}$	36	18	3, 75
77	$C_2 \times C_{30}$	60	30	2, 76
78	$C_2 \times C_{12}$	24	12	5, 7
79	C_{78}	78	78	3
80	$C_2 \times C_4 \times C_4$	32	4	3, 7, 11

原根和 DFT

- 你们可能听说过原根的概念。
- 对于素数 P，存在一个数 a，使得 $a^0, a^1, \cdots, a^{P-2}$ 遍历模 P 剩余类除 0 以外的每一类。a 称为 P 的原根，这一步是 DFT 的基础。
- 事实上，这里 a 就是模 P 乘法群的一个生成元。
  - 注意！不是每个元素都是生成元。比如 3 就不是 $(Z/6Z)$ 的生成元；你需要手动算元素周期。
- 但是，也像我们刚才看到的，如果 P 不是素数，乘法群不是循环群，原根就不一定存在。
- FFT 的核心思想是将多项式在 $\omega^1, \omega^2 \cdots, \omega^N$ 上插值。DFT 中，ω 被替换为 a。
  - 通过简单类比可以发现 a 也有 ω 的性质：$a^{P-1} = 1$，且小于 $P-1$ 时互不相同。
  - DFT 的优势在于避免了 FFT 大量三角函数的计算，也完全躲开了计算误差，但使用范围有限，且受 P 大小的限制。

CRT

- 我们（顺便）来 fancy 地讲讲中国剩余定理
- 如果 $n = p_1 p_2 \cdots p_n$，那么 $(Z/n)*$ 同构于 (Z/p_i) 这 n 个群的直积。

欧拉公式的证明

- 考虑模 P 乘法群。幺元是 1，大小是 $\phi(N)$。
- 群上元素 x 的周期定义为最小正整数 k，使得 $x + x + \cdots + x = 0$，0 是群幺元。
- 有一个经典定理：有限群每个元素的周期是群大小的约数。
- 说人话，对每个 x 使得 $(x, N) = 1$，存在 $d|\phi(N)$ 使得 $x^d = 1 (\mod N)$，那么显然 $x^{\phi(N)} = 1 (\mod N)$。