部分数学基础

11D_Beyonder, CUG

1. 数论
2. 组合数学

数论基础

1. 素数
2. 分解质因数
3. 素数筛
4. 欧拉函数
5. 欧拉定理
6. 欧几里得算法
7. 扩展欧几里得算法
8. 乘法逆元

符号说明

1. 整除：$m \mid n$，m整除n。M|N表示M能整除N 即M是N的因数,N是M的倍数 如2|4,3|12
2. 互质：$m \perp n$，即$\gcd(m,n)=1$。
3. 取整：$\lfloor x \rfloor$、$\lceil x \rceil$。
4. 同余：$a \equiv b \pmod m$

素数

只有1和其本身两个因子的数称为**素数**，也称**质数**。

2, 3, 5, 7, 11, 13, 17…

素数检验

~~依次检测$i \in [2, x-1]$中是否满足$i \mid x$即可，复杂度$O(x)$。~~

若$\boldsymbol{i \mid x}$，则必有$\boldsymbol{\frac{x}{i} \mid x}$，因此是在$\boldsymbol{\sqrt{x}}$两侧成对出现的，因此只需要检测$\boldsymbol{i \in \left[2, \left\lfloor\sqrt{x}\right\rfloor\right]}$中是否满足$\boldsymbol{i \mid x}$即可，复杂度$\boldsymbol{O\left(\sqrt{x}\right)}$。

Miller-Rabin素数测试 略

```
bool is_prime(int x)
{
    for (int i = 2; i * i <= x; i++)
    {
        if (x % i == 0)
        {
            return false;
        }
    }
    return true;
}
```

分解质因数

算术基本定理：$\forall \boldsymbol{n} \in \boldsymbol{N}$，$\boldsymbol{A} > \mathbf{1}\ \exists \prod_{i=1}^{n} \boldsymbol{p}_i^{\boldsymbol{a}_i} = \boldsymbol{A}$，其中$\boldsymbol{p}_1 < \boldsymbol{p}_2 < \cdots < \boldsymbol{p}_n$，且$\boldsymbol{p}_i$是质数，$\boldsymbol{a}_i \in \boldsymbol{Z}^+$。

若$\boldsymbol{i} \mid \boldsymbol{x}$，则必有$\frac{x}{i} \mid \boldsymbol{x}$，因此是在$\sqrt{x}$两侧成对出现的，因此只需要检测$\boldsymbol{i} \in \left[2, \left\lfloor\sqrt{x}\right\rfloor\right]$中的质因子即可。

Pollard-Rho算法 略

```
void get_prime_factors(int x)
{
    for (int i = 2; i * i <= x; i++)
    {
        if (x % i == 0)
        {
            prime_factors.emplace_back(i);
            while (x % i == 0)
            {
                x /= i;
                ++cnt[i];
            }
        }
        if (x > 1)
        {
            prime_factors.emplace_back(x);
            ++cnt[x];
        }
    }
}
```

素数筛

想要知道$[1,n]$内的所有素数，逐个暴力检验显然不是最优复杂度，因此出现了诸多素数筛方法。

埃拉托斯特尼筛

从小到大考虑每个数$x \in [2, n]$，将每个x的倍数标记为合数，最后没有被标记的就是素数。

时间复杂度$O(n \log \log n)$。

```
void eratosthenes(int n)
{
    for (int i = 0; i <= n;i++)
    {
        is_prime[i] = true;
    }
    is_prime[0] = is_prime[1] = false;
    for (int i = 2; i <= n;i++)
    {
        if(is_prime[i])
        {
            for (int j = 2; i * j <= n;j++)
            {
                is_prime[i * j] = false;
            }
        }
    }
}
```

欧拉筛

埃氏筛将同一个合数重复多次标记，浪费时间。

欧拉筛只会将每个合数标记一次，时间复杂度为$O(n)$。

每个合数只被其最小质因子筛去。

```
void euler(int n)
{
    for (int i = 0; i <= n; i++)
    {
        is_prime[i] = true;
    }
    is_prime[0] = is_prime[1] = false;
    int cnt = 0;
    for (int i = 2; i <= n; i++)
    {
        if (is_prime[i])
        {
            prime[++cnt] = i;
        }
        for (int j = 1; j <= cnt && i * prime[j] <= n; j++)
        {
            is_prime[i * prime[j]] = false;
            if (i % prime[j] == 0)
            {
                break;
            }
        }
    }
}
```

欧拉函数

$\varphi(n)$表示$[1,n]$内与n互质的数的个数。

1. 若n为质数，则$\varphi(n) = n - 1$。
2. 若a,b互质，则$\varphi(ab) = \varphi(a)\varphi(b)$。
3. 若$n = p^k$，其中p为质数，则$\varphi(n) = p^k - p^{k-1}$。
4. 设$n = \prod_{i=1}^{s} p_i^{k_i}$，其中$p_i$为质数，则$\varphi(n) = n\prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right)$。

```
int euler_phi(int n)
{
    int ans = n;
    for (int i = 2; i * i <= n; i++)
    {
        if (n % i == 0)
        {
            ans = ans / i * (i - 1);
            while (n % i == 0)
            {
                n /= i;
            }
        }
    }
    if (n > 1)
    {
        ans = ans / n * (n - 1);
    }
    return ans;
}
```

欧拉函数

1. 当n为质数，则$\varphi(n) = n - 1$。
2. 当n为合数，设其最小质因子为p，且$m = \frac{n}{p}$；若$p \mid m$，则m与n的质因子种类相同，所以$\varphi(n) = n\prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right) = p \cdot m\prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right) = p\varphi(m)$；若$p \nmid m$，则$\varphi(n) = \varphi(pm) = (p-1)\varphi(m)$。

```
void euler_phi(int n)
{
        // 初始化（略）
    for (int i = 2; i <= n; i++)
    {
        if (is_prime[i])
        {
            prime[++cnt] = i;
            phi[i] = i - 1;
        }
        for (int j = 1; j <= cnt && i * prime[j] <= n; j++)
        {
            is_prime[i * prime[j]] = false;
            if (i % prime[j])
            {
                phi[i * prime[j]] = phi[i] * phi[prime[j]];
            }
            else
            {
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            }
        }
    }
}
```

费马小定理

若p为素数，且$\gcd(a,p)=1$，则$a^{p-1}\equiv 1 \pmod{p}$。

可使用数学归纳法证明 略

欧拉定理

若$\gcd(a,m)=1$，则$a^{\varphi(m)} \equiv 1 \pmod{m}$

扩展欧拉定理

$$a^b \equiv \begin{cases} a^{b \bmod \varphi(m)}, & \gcd(a,m) = 1 \\ a^b, & \gcd(a,m) \neq 1, b < \varphi(m) \\ a^{\left(b \bmod \varphi(m)\right)+\varphi(m)}, & \gcd(a,m) \neq 1, b \geq \varphi(m) \end{cases} \pmod m$$

欧拉降幂

洛谷P5091 【模板】扩展欧拉定理

给三个正整数，求$a^b \bmod m$。

$1 \le a \le 10^9$，$1 \le b \le 10^{20000000}$，$1 \le m \le 10^8$。

$$a^b \equiv \begin{cases} a^{b \bmod \varphi(m)}, & \gcd(a,m)=1 \\ a^b, & \gcd(a,m) \ne 1, b < \varphi(m) \\ a^{(b \bmod \varphi(m))+\varphi(m)}, & \gcd(a,m) \ne 1, b \ge \varphi(m) \end{cases} \pmod m$$

```
int read()
{
    char c;
    while (!isdigit(c = getchar()));
    for (; isdigit(c); c = getchar())
    {
        b = b * 10 + c - '0';
        if (b >= phi_m)
        {
            flag = true;
            b %= phi_m;
        }
    }
    if (flag)
    {
        b += phi_m
    }
}
```

欧几里得算法

$\gcd(a, b) = \gcd(b, a \bmod b)$

```
int gcd(int a, int b)
{
    if (b == 0)
    {
        return a;
    }
    return gcd(b, a % b);
}
```

C++ __gcd(a, b)

证明

不妨令$a = bq + r$，其中$q = \left\lfloor \frac{a}{b} \right\rfloor$，$r = a \bmod b$。

设d是a, b的一个公因数。$\frac{r}{d} = \frac{a}{d} - \frac{b}{d}q$，因为$d$是公因数，所以$d \mid r$。

设d是b, r的一个公因数。$\frac{r}{d} = \frac{a}{d} - \frac{b}{d}q$，因为$d$是公因数，所以$d \mid a$。

$\boldsymbol{a, b}$**的公因数也是**$\boldsymbol{b, a \bmod b}$**的公因数，最大公因数显然也是。**

扩展欧几里得算法

扩展欧几里得算法可得到$ax + by = \gcd(a, b)$ 的一组整数解。

1. 对于任意整数a, b，在欧几里得算法的最后一步，即$b = 0$时，显然有$x = 1, y = 0$使得$ax + by = \gcd(a, b)$ 。
2. 若$b > 0$，则$\gcd(a, b) = \gcd(b, a \bmod b)$。假设存在一对整数$x, y$，满足$bx + (a \bmod b)y = \gcd(a, a \bmod b)$ ，那么$ay + b\left(x - \left\lfloor\frac{a}{b}\right\rfloor\right) = \gcd(a, b)$。

此处为b

掉了y

```
int ex_gcd(int a, int b, int &x, int &y)
{
    if (!b)
    {
        x = 1;
        y = 0;
        return a;
    }
    else
    {
        int gcd = ex_gcd(b, a % b, x, y);
        //向上递推
        int temp = x;
        x = y;
        y = temp - a / b * y;
        return gcd;
    }
}
```

求解$ax + by = c$

当且仅当$\gcd(a, b) \mid c$，该方程有整数解。

可以先求出$ax + by = \gcd(a, b)$ 的一组特解(x^*, y^*)，然后就得到了$ax + by = c$ 的一组特解$\left(\frac{c}{\gcd(a,b)}x^*, \frac{c}{\gcd(a,b)}y^*\right)$。

通解

$$\left(\frac{\boldsymbol{c}}{\boldsymbol{gcd(a,b)}}\boldsymbol{x}^* + \boldsymbol{k}\frac{\boldsymbol{b}}{\mathbf{gcd}(\boldsymbol{a},\boldsymbol{b})}, \frac{\boldsymbol{c}}{\boldsymbol{gcd(a,b)}}\boldsymbol{y}^* - \boldsymbol{k}\frac{\boldsymbol{a}}{\mathbf{gcd}(\boldsymbol{a},\boldsymbol{b})}\right)$$

乘法逆元

若$ax \equiv 1 \pmod{b}$，则x称为a在模b意义下的逆元。

方法	数学形式
费马小定理	$x \equiv a^{b-2} \pmod{b}$
扩展欧几里得	$ax + kb = 1$
线性递推	$\mathrm{inv}_i \equiv \left(p - \left\lfloor \frac{b}{i} \right\rfloor \mathrm{inv}_{b \bmod i}\right) \pmod{b}$

组合数学基础

1. 计数问题
2. 容斥原理

计数问题

1. 乘法原理
2. 加法原理
3. 排列数$A_n^m = \frac{n!}{(n-m)!}$
4. 组合数$C_n^m = \frac{n!}{m!(n-m)!}$
5. 隔板法、统计贡献、枚举
6. 卡特兰数、斯特林数……

洛谷P2415 集合求和

给定集合s，求出此集合所有子集元素之和。

举例：$s=\{2,3\}$，子集为$\emptyset,\{2\},\{3\},\{2,3\}$，和为$2+3+2+3=10$。

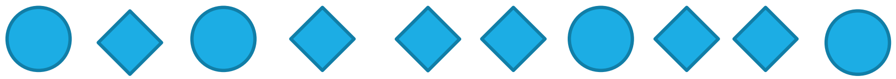
共有$2^{|s|}$个子集。

每个元素的贡献是等价的，只会在$2^{|s|-1}$个子集中出现。

答案 $2^{|s|-1}\sum_{x\in s}x$。

洛谷P5520 [YLOI2019] 青原樱

在n个位置可以种下樱花，准备了m棵幼苗。由于樱花盛放时对左右空间需求非常大，所以樱花不能紧挨着种植，也就是任意两支幼苗之间必须至少存在一个不种花的空位置。一共有多少合法的方案让他把这 m 支幼苗都种下去？两种方案不同当且仅当被选择种花的位置不同或从左向右数花的编号序列不同。



共$n-m$个空位，m棵幼苗，幼苗将空位隔开。

答案 A_{n-m+1}^{m}

洛谷P3799 妖梦拼木棒

有n根木棒，现在从中选出4根，拼成正三角形，问有几种选法?

第i根木棒长度用a_i表示。

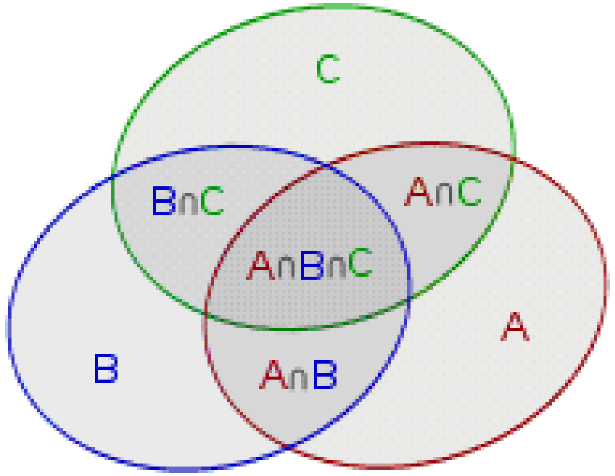
$1 \le n \le 10^5$，$0 \le a_i \le 5 \times 10^3$。

选出的4根木棒，必然是其中两根的长度和为l，而另两根的长度各为l。

可以枚举l的长度，并统计两根木棒长度和为l的可选方案，累加得到答案。

容斥原理

$$\left|\bigcup_{i=1}^{n} S_i\right| = \sum_{k=1}^{n} (-1)^{k-1} \sum_{1 \le i_1 < i_2 < \cdots < i_k \le n} \left|\bigcap_{t=1}^{k} S_t\right|$$



$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$

UVA11806 CHEERLEADERS

$n \times m$的网格图，将k个石头放在网格里，满足如下条件：

1. 网格图的四个边上都至少有一个石头，
2. 每个格子至多有一个石头，
3. 每个石头都必须有位置。

四个角的石头可以同时算作在两个边上。

mn个空格放k个石子，总方案数C_{mn}^{k}，已满足条件2、3。

要求第一行、第一列、第m行、第n列必须有石子。

设A_1为第一行没有石子的方案数，A_2为第一例没有石子的方案书，A_3为第m行没有石子的方案书，A_4为第n列没有石子的方案数。

答案 $\boldsymbol{C_{mn}^{k}} - \left|\bigcup_{i=1}^{4} \boldsymbol{A_i}\right|$ （容斥原理计算）