西安电子科技大学 XIDIAN UNIVERSITY | 程序设计竞赛实训基地 Programming Contest Training Base

数论基础

ACM 中的数学小知识

陈德创

Codeforces: dcac

July 10, 2021

简介

主要讲什么？

- ACM 中的基础数学小知识

简介

主要讲什么？

- ACM 中的基础数学小知识

有啥用？

- 单独的题目考察
- 一些题目解答的必备前提
- 玩

符号约定

1. $x|y$：x 整除 y，即 x 是 y 的余数
2. $x \mod y$：x 除以 y 的得到的余数
3. $x \equiv y \pmod M$：x 同余 y（在模 M 的意义下）
4. $\gcd(x,y)$：x 和 y 的最大公约数，简写作 (x,y)
5. $\operatorname{lcm}(x,y)$：x 和 y 的最小公倍数，简写作 $[x,y]$
6. $\lfloor x \rfloor$：下取整
7. $\lceil x \rceil$：上取整
8. $\sum$：求和，如 $\sum\limits_{i=1}^{n} f(i)$，$\sum\limits_{1 \le i \le n, i \in P} f(i)$

整除基本性质

整除的定义：若整数 a 除以非零整数 b，商为整数且余数为零，即 a 能被 b 整除，记做 $b|a$，读作：b 整除 a 或 a 能被 b 整除。a 叫做 b 的倍数，b 叫做 a 的约数，或称为因数。

整除基本性质

整除的定义：若整数 a 除以非零整数 b，商为整数且余数为零，即 a 能被 b 整除，记做 $b|a$，读作：b 整除 a 或 a 能被 b 整除。a 叫做 b 的倍数，b 叫做 a 的约数，或称为因数。

- 若 $a|c, b|c$，则 $a|(b\pm c)$
- 若 $a|b$，则对任意的 $c(c\neq 0)$，有 $a|bc$
- 若 $a|b$，且 $b|c$，则 $a|c$
- 若 $a|bc$，且 $(a,c)=1$，则 $a|b$
- 若 $c|a$，且 $c|b$，则对于任意整数 m,n，有 $c|(ma+nb)$
- 若 $a|c, b|c$，且 $(a,b)=1$，则 $ab|c$
- **带余除法定理**：$\forall a,b>0, a,b\in\mathbb{N}$，$\exists$ 唯一的数对 q,r，使 $a=bq+r,(0\leq r<b)$。

同余基本性质

- 若 $a \equiv b \pmod{m}, c \equiv d \pmod{m}$，则 $a \pm b \equiv c \pm d \pmod{m}$
- 若 $a \equiv b \pmod{m}, c \equiv d \pmod{m}$，则 $a \times b \equiv c \times d \pmod{m}$

同余基本性质

- 若 $a \equiv b \pmod m, c \equiv d \pmod m$，
  则 $a \pm b \equiv c \pm d \pmod m$
- 若 $a \equiv b \pmod m, c \equiv d \pmod m$，
  则 $a \times b \equiv c \times d \pmod m$
- 证明基本思路：$a = k_1 m + b, c = k_2 m + b$

同余基本性质

- 若 $a \equiv b \pmod{m}, c \equiv d \pmod{m}$,
  则 $a \pm b \equiv c \pm d \pmod{m}$
- 若 $a \equiv b \pmod{m}, c \equiv d \pmod{m}$,
  则 $a \times b \equiv c \times d \pmod{m}$
- 证明基本思路：$a = k_1 m + b, c = k_2 m + b$
- 基本用法：求模运算下求解问题（避免大数运算）、快速幂

同余基本性质

- 若 $a \equiv b \pmod{m}, c \equiv d \pmod{m}$，
  则 $a \pm b \equiv c \pm d \pmod{m}$
- 若 $a \equiv b \pmod{m}, c \equiv d \pmod{m}$，
  则 $a \times b \equiv c \times d \pmod{m}$
- **证明基本思路**：$a = k_1 m + b, c = k_2 m + b$
- **基本用法**：求模运算下求解问题（避免大数运算）、快速幂
- **注意**：除法不满足以上的性质

快速幂

求解：$a^b \equiv x \pmod{m}, (a, b \le 1e9)$

快速幂

求解：$a^b \equiv x \pmod{m}, (a, b \leq 1e9)$

- 朴素求法：

```
inline ll mpow(ll a, ll b, ll m) {
  ll res = 1;
  for (int i = 1; i <= b; ++i)
    res = res * a % m;
  return res;
}
```

- 时间复杂度：$O(b)$

快速幂

求解：$a^b \equiv x \pmod{m}, (a, b \leq 1e9)$

- 快速幂

```
inline ll mpow(ll a, ll b, ll m) {
  ll res = 1;
  while(b) {
    if (b & 1) res = res * a % m;
    a = a * a % m;
    b >>= 1;
  }
  return res;
}
```

- 时间复杂度：$O(\log b)$

素数基础

定义：只能被 1 和自身整除的数称为素数，又称为质数

- 1 既不是素数，也不是合数
- 2 是最小的素数，也是唯一的偶素数

素数基础

定义：只能被 1 和自身整除的数称为素数，又称为质数

- 1 既不是素数，也不是合数
- 2 是最小的素数，也是唯一的偶素数

基本性质：

- 素数计数函数 $\pi(x)$：$\pi(x) \sim \frac{x}{\ln x}$

素数基础

定义：只能被 1 和自身整除的数称为素数，又称为质数

- 1 既不是素数，也不是合数
- 2 是最小的素数，也是唯一的偶素数

基本性质：

- 素数计数函数 $\pi(x)$：$\pi(x) \sim \frac{x}{\ln x}$
- 素数的分布：1e9 范围内，任意两个相邻的素数差不超过 400

素数基础

定义：只能被 1 和自身整除的数称为素数，又称为质数

- 1 既不是素数，也不是合数
- 2 是最小的素数，也是唯一的偶素数

基本性质：

- 素数计数函数 $\pi(x)$：$\pi(x) \sim \frac{x}{\ln x}$
- 素数的分布：1e9 范围内，任意两个相邻的素数差不超过 400
- **威尔逊定理**：$(p-1)! \equiv -1 \pmod p$

素数的判定

- 暴力枚举所有可能的因子即可
- 事实：如果 $x|a$，那么 $\frac{a}{x}|a$。
- 判定算法：

```
inline bool isPrime(ll n) {
  if (n < 2) return false;
  for (int i = 2; i <= sqrt(n) + 1; ++i) {
    if (!(n % i)) return false;
  }
  return true;
}
```

- 时间复杂度：$O(\sqrt{n})$

素数的判定

- 暴力枚举所有可能的因子即可
- 事实：如果 $x|a$，那么 $\frac{a}{x}|a$。
- 判定算法：

```
inline bool isPrime(ll n) {
  if (n < 2) return false;
  for (int i = 2; i <= sqrt(n) + 1; ++i) {
    if (!(n % i)) return false;
  }
  return true;
}
```

- 时间复杂度：$O(\sqrt{n})$
- Miller-Rabin 素性测试：$O(k\log^3 n)$
- ~~我不会~~

筛法

筛法用于求出 $1 \sim n$ 中所有的素数。

- 事实：一个数 x 的任意整数倍一定是素数

筛法

筛法用于求出 $1 \sim n$ 中所有的素数。

- 事实：一个数 x 的任意整数倍一定是素数
- 对每个数，将其倍速标记为**非素数**，那么没被标记的数，就是素数

筛法

筛法用于求出 $1 \sim n$ 中所有的素数。

- 事实：一个数 x 的任意整数倍一定是素数
- 对每个数，将其倍速标记为**非素数**，那么没被标记的数，就是素数
- 算法实现：

```
const int MAXN = 1e6 + 5;
bool inp[MAXN], prime[MAXN], cnt;
inline void getPrime(int n) {
    memset(inp, 0, sizeof(inp));
    for (int i = 2; i <= n; ++i) {
        if (!inp[i]) prime[++cnt] = i;
        for (int j = 2; i * j <= n; ++j)
        inp[i * j] = true;
    }
}
```

- 时间复杂度：$O(n \log n)$

埃拉托斯特尼筛法

- 只需要对素数的倍数进行标记就好了

埃拉托斯特尼筛法

- 只需要对素数的倍数进行标记就好了
- 算法实现：

```
inline void getPrime(int n) {
    memset(inp, 0, sizeof(inp));
    for (int i = 2; i <= n; ++i) {
        if (inp[i]) continue;
        prime[++cnt] = i;
        for (int j = 2; i * j <= n; ++j)
            inp[i * j] = true;
    }
}
```

- 时间复杂度： $O(n \log \log n)$

线性筛
欧拉筛

- 如果能让每个合数都只被标记一次就好了

线性筛

欧拉筛

- 如果能让每个合数都只被标记一次就好了
- 算法实现：

```
inline void getPrime(int n) {
    for (int i = 2; i <= n; ++i) {
        if (!inp[i]) prime[++cnt] = i;
        for (int j = 1; j <= cnt && i * prime[j] <= n; ++j) {
            inp[i * prime[j]] = true;
            if (!(i % prime[j])) break;
        }
    }
}
```

线性筛

欧拉筛

- 如果能让每个合数都只被标记一次就好了
- 算法实现：

```cpp
inline void getPrime(int n) {
    for (int i = 2; i <= n; ++i) {
        if (!inp[i]) prime[++cnt] = i;
        for (int j = 1; j <= cnt && i * prime[j] <= n; ++j) {
            inp[i * prime[j]] = true;
            if (!(i % prime[j])) break;
        }
    }
}
```

- 时间复杂度：$O(n)$
- 非常重要，可以用来线性求解某些积性函数

因数

因数

定理（算术基本定理）

$\forall A \in \mathbb{N}, A > 1 \quad \exists p_1 < p_2 < p_3 < \cdots < p_n, a_i \in \mathbb{Z}^+ \text{ s.t. } \prod_{i=1}^{n} p_i^{a_i} = A$
其中 p_i 是一个质数。这种表示的方法存在，而且是唯一的。

因数

定理（算术基本定理）

$\forall A \in \mathbb{N}, A > 1 \quad \exists p_1 < p_2 < p_3 < \cdots < p_n, a_i \in \mathbb{Z}^+ \text{ s.t. } \prod_{i=1}^{n} p_i^{a_i} = A$
其中 p_i 是一个质数。这种表示的方法存在，而且是唯一的。

- $1 \sim 1e9$ 的数最多有 1344 个因子。（by 欢神）

因数

定理（算术基本定理）

$\forall A \in \mathbb{N}, A > 1 \quad \exists p_1 < p_2 < p_3 < \cdots < p_n, a_i \in \mathbb{Z}^+ \text{ s.t. } \prod_{i=1}^{n} p_i^{a_i} = A$
其中 p_i 是一个质数。这种表示的方法存在，而且是唯一的。

- $1 \sim 1e9$ 的数最多有 1344 个因子。（by 欢神）
- 求解一个数的所有因数：

```
vector<int> breakdown(int n) {
  vector<int> res;
  for (int i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      res.push_back(i);
      res.push_back(n / i);
    }
  }
  return res;
}
```

- 时间复杂度：$O(\sqrt{n})$

因数

- 求解一个数的所有**质因数**（即质因数分解）:

因数

- 求解一个数的所有**质因数**（即质因数分解）：

```
vector<int> breakdown(int n) {
  vector<int> res;
  for (int i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      while (n % i == 0) n /= i;
      res.push_back(i);
    }
  }
  if (n != 1) res.push_back(n);
  return res;
}
```

- 时间复杂度：$O(\sqrt{\frac{n}{\ln n}})$

最大公约数

GCD

最大公约数，是指一组数所有公约数里面最大的一个。

最大公约数

GCD

最大公约数，是指一组数所有公约数里面最大的一个。

- a 和 b 的最大公约数记为：$\gcd(a, b)$

最大公约数

GCD

最大公约数，是指一组数所有公约数里面最大的一个。

- a 和 b 的最大公约数记为：$\gcd(a, b)$
- **事实**：

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

最大公约数
GCD

最大公约数，是指一组数所有公约数里面最大的一个。

- a 和 b 的最大公约数记为：$\text{gcd}(a, b)$
- **事实**：

$$\text{gcd}(a, b) = \text{gcd}(b, a - b)$$
$$\text{gcd}(a, b) = \text{gcd}(a, a \bmod b)$$

- 集合 $S = \{\text{gcd}(a_i, \ldots, a_n) | 1 \le i \le n\}$ 至多有 $O(\text{max}(a_i))$ 个元素。

欧几里得算法

- 算法实现：

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

- 时间复杂度：$O(\log n)$

欧几里得算法

- 算法实现：

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

- 时间复杂度：$O(\log n)$
- 求解**斐波那契数列**相邻两项时达到最坏时间复杂度

欧几里得算法

- 算法实现:

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

- 时间复杂度: $O(\log n)$
- 求解**斐波那契数列**相邻两项时达到最坏时间复杂度
- <algorithm>库内置了__gcd()函数

欧几里得算法

- 算法实现：

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

- 时间复杂度：$O(\log n)$
- 求解**斐波那契数列**相邻两项时达到最坏时间复杂度
- <algorithm>库内置了__gcd()函数
- 本质上是将将大规模问题转化为小规模问题
- 将问题 (a, b) 转化为了 $(b, a \bmod b)$

多个数的最大公约数

求解多个数的最大公约数：

- 可以证明，$\gcd(a, \gcd(b, c)) = \gcd(a, b, c)$
- 即每次取出两个数求出 gcd 后再放 gcd 回去，不会对所需要的答案造成影响。

多个数的最大公约数

求解多个数的最大公约数：

- 可以证明，$\gcd(a, \gcd(b, c)) = \gcd(a, b, c)$
- 即每次取出两个数求出 gcd 后再放 gcd 回去，不会对所需要的答案造成影响。

```
int gcd(int a[], int n) {
  int res = a[1];
  for (int i = 2; i <= n; ++i)
    res = __gcd(res, a[i]);
  return res;
}
```

例题

又是毕业季

题目链接：又是毕业季 I

题目描述

给定 n 和 k，问在 $1 \sim n$ 中选 k 个数使得它们 gcd 最大，求最大值是多少。
$1 \le k \le n \le 1e9$

例题

又是毕业季

题目链接：又是毕业季 I

题目描述

给定 n 和 k，问在 $1 \sim n$ 中选 k 个数使得它们 gcd 最大，求最大值是多少。
$1 \le k \le n \le 1e9$

题解

- **反向考虑**，即考虑一个数 x，$1 \sim n$ 中有多少数被它整除。

例题

又是毕业季

题目链接：又是毕业季 I

题目描述

给定 n 和 k，问在 $1 \sim n$ 中选 k 个数使得它们 gcd 最大，求最大值是多少。
$1 \leq k \leq n \leq 1e9$

题解

- **反向考虑**，即考虑一个数 x，$1 \sim n$ 中有多少数被它整除。
- 显然为 $\lfloor \frac{n}{x} \rfloor$ 个
- 即求 $\max_{x}\{[\lfloor \frac{n}{x} \rfloor \geq k]\}$，显然答案为 $\lfloor \frac{n}{k} \rfloor$

例题
又是毕业季

题目链接：又是毕业季 I

题目描述

给定 n 和 k，问在 $1 \sim n$ 中选 k 个数使得它们 gcd 最大，求最大值是多少。
$1 \leq k \leq n \leq 1e9$

题解

- **反向考虑**，即考虑一个数 x，$1 \sim n$ 中有多少数被它整除。
- 显然为 $\lfloor \frac{n}{x} \rfloor$ 个
- 即求 $\max_x\{[\lfloor \frac{n}{x} \rfloor \geq k]\}$，显然答案为 $\lfloor \frac{n}{k} \rfloor$
- 时间复杂度：$O(1)$

例题

又是毕业季

题目链接：又是毕业季 I

题目描述

给定 n 和 k，问在 $1 \sim n$ 中选 k 个数使得它们 gcd 最大，求最大值是多少。
$1 \leq k \leq n \leq 1e9$

题解

- 反向考虑，即考虑一个数 x，$1 \sim n$ 中有多少数被它整除。
- 显然为 $\lfloor \frac{n}{x} \rfloor$ 个
- 即求 $\max\limits_{x}\{[\lfloor \frac{n}{x} \rfloor \geq k]\}$，显然答案为 $\lfloor \frac{n}{k} \rfloor$
- 时间复杂度：$O(1)$

```python
n, k = map(int, input().split())
print(int(n / k))
```

例题

又是毕业季

题目链接：又是毕业季 I

题目描述

给定 n 和 k，问在 $1 \sim n$ 中选 k 个数使得它们 gcd 最大，求最大值是多少。
$1 \le k \le n \le 1e9$

题解

- **反向考虑**，即考虑一个数 x，$1 \sim n$ 中有多少数被它整除。
- 显然为 $\lfloor \frac{n}{x} \rfloor$ 个
- 即求 $\max_{x}\{[\lfloor \frac{n}{x} \rfloor \ge k]\}$，显然答案为 $\lfloor \frac{n}{k} \rfloor$
- 时间复杂度：$O(1)$

```python
n, k = map(int, input().split())
print(int(n / k))
```

同样思想解决的题：又是毕业季 II（双倍经验它不香吗 ?）。

例题

最大公约数

题目链接：最大公约数

题目描述

有一个 $n \times m$ 的矩阵 a。对此矩阵进行变换，定义为将这个矩阵内的所有元素变为其上下左右四个元素（不存在则忽略）及自身的最大公约数。
询问 $a_{x,y}$ 在进行最少多少次变换之后会变成 1。如果可以使 $a_{x,y}$ 经过若干次变换变成 1，输出其中最小的次数；否则输出 -1。

例题
最大公约数

题目链接：最大公约数

题目描述

有一个 $n \times m$ 的矩阵 a。对此矩阵进行变换，定义为将这个矩阵内的所有元素变为其上下左右四个元素（不存在则忽略）及自身的最大公约数。
询问 $a_{x,y}$ 在进行最少多少次变换之后会变成 1。如果可以使 $a_{x,y}$ 经过若干次变换变成 1，输出其中最小的次数；否则输出 -1。

数据范围：

$$1 \le n, m \le 2 \times 10^3, 1 \le a_{i,j} \le 10^{18}, 1 \le x \le n, 1 \le y \le m$$

例题

最大公约数

题目链接：最大公约数

题目描述

有一个 $n \times m$ 的矩阵 a。对此矩阵进行变换，定义为将这个矩阵内的所有元素变为其上下左右四个元素（不存在则忽略）及自身的最大公约数。
询问 $a_{x,y}$ 在进行最少多少次变换之后会变成 1。如果可以使 $a_{x,y}$ 经过若干次变换变成 1，输出其中最小的次数；否则输出 -1。

数据范围：

$$1 \le n, m \le 2 \times 10^3, 1 \le a_{i,j} \le 10^{18}, 1 \le x \le n, 1 \le y \le m$$

hint

- **事实**:

$$\begin{aligned} \gcd(a, \gcd(b, c)) &= \gcd(a, b, c) \\ \gcd(\gcd(a, b), \gcd(b, c)) &= \gcd(a, b, c) \end{aligned}$$

- 分别考虑“判断可能”和“求解”。

例题
最大公约数

题解

显然，当且仅当所有元素的 gcd 不为 1 时，答案不存在。

例题
最大公约数

题解

显然，当且仅当所有元素的 $\gcd$ 不为 1 时，答案不存在。
当答案存在时，我们考虑每一次矩阵变换。

例题

最大公约数

题解

显然，当且仅当所有元素的 $\gcd$ 不为 1 时，答案不存在。
当答案存在时，我们考虑每一次矩阵变换。

- 第一次，$a_{x,y}$ 由 $a_{x,y}$ 上下左右四个点来更新。
  即 $a_{x,y} = \gcd(a_{x-1,y}, a_{x+1,y}, a_{x,y-1}, a_{x,y+2})$。
- 记 $a_{x,y}$ 周围的四个点为 b_1, b_2, b_3, b_4。
  同时 b_i 也由它们的上下左右四个点来更新

例题
最大公约数

题解

显然，当且仅当所有元素的 gcd 不为 1 时，答案不存在。
当答案存在时，我们考虑每一次矩阵变换。

- 第一次，$a_{x,y}$ 由 $a_{x,y}$ 上下左右四个点来更新。
  即 $a_{x,y} = \text{gcd}(a_{x-1,y}, a_{x+1,y}, a_{x,y-1}, a_{x,y+2})$。
- 记 $a_{x,y}$ 周围的四个点为 b_1, b_2, b_3, b_4。
  同时 b_i 也由它们的上下左右四个点来更新
- 第二次，相当于 $a_{x,y}$ 由每个 b_i 周围的四个点更新。
  即相当于 $a_{x,y}$ 由距离它小于 2 的点更新。
- 同时 b_i 也是由距离它小于 2 的点更新。

例题
最大公约数

题解

显然，当且仅当所有元素的 gcd 不为 1 时，答案不存在。
当答案存在时，我们考虑每一次矩阵变换。

- 第一次，$a_{x,y}$ 由 $a_{x,y}$ 上下左右四个点来更新。
  即 $a_{x,y} = \gcd(a_{x-1,y}, a_{x+1,y}, a_{x,y-1}, a_{x,y+2})$。
- 记 $a_{x,y}$ 周围的四个点为 b_1, b_2, b_3, b_4。
  同时 b_i 也由它们的上下左右四个点来更新
- 第二次，相当于 $a_{x,y}$ 由每个 b_i 周围的四个点更新。
  即相当于 $a_{x,y}$ 由距离它小于 2 的点更新。
- 同时 b_i 也是由距离它小于 2 的点更新。
- 以此类推，假设第 k 次，$a_{x,y}$ 由距离它小于 k 的点更新。
  那么 b_i 也是由距离它小于 k 的点更新。

例题
最大公约数

题解

显然，当且仅当所有元素的 gcd 不为 1 时，答案不存在。
当答案存在时，我们考虑每一次矩阵变换。

- 第一次，$a_{x,y}$ 由 $a_{x,y}$ 上下左右四个点来更新。
  即 $a_{x,y} = \text{gcd}(a_{x-1,y}, a_{x+1,y}, a_{x,y-1}, a_{x,y+2})$。
- 记 $a_{x,y}$ 周围的四个点为 b_1, b_2, b_3, b_4。
  同时 b_i 也由它们的上下左右四个点来更新
- 第二次，相当于 $a_{x,y}$ 由每个 b_i 周围的四个点更新。
  即相当于 $a_{x,y}$ 由距离它小于 2 的点更新。
- 同时 b_i 也是由距离它小于 2 的点更新。
- 以此类推，假设第 k 次，$a_{x,y}$ 由距离它小于 k 的点更新。
  那么 b_i 也是由距离它小于 k 的点更新。
- 那么显然第 $k+1$ 次，$a_{x,y}$ 由距离 b_i 小于 k 的点更新。
  即相当于距离它小于 $k+1$ 的点更新。

例题

最大公约数

题解

- 即我们归纳证明了，第 k 次，$a_{x,y}$ 由距离它小于 k 的点更新。
- 即，第 k 次变换后，$a_{x,y}$ 的值变成了所有距它小于等于 k 的点的值的 gcd。

例题

最大公约数

题解

- 即我们归纳证明了，第 k 次，$a_{x,y}$ 由距离它小于 k 的点更新。
- 即，第 k 次变换后，$a_{x,y}$ 的值变成了所有距它小于等于 k 的点的值的 gcd。
- 题目转化为：从 (x,y) 开始一层层取 gcd，直到 $\gcd = 1$。

例题

最大公约数

题解

- 即我们归纳证明了，第 k 次，$a_{x,y}$ 由距离它小于 k 的点更新。
- 即，第 k 次变换后，$a_{x,y}$ 的值变成了所有距它小于等于 k 的点的值的 gcd。
- 题目转化为：从 (x,y) 开始一层层取 gcd，直到 $\gcd = 1$。
- 从 (x,y) 开始 bfs，同时对每个遍历到的元素取 gcd，直到 $\gcd = 1$。
- 遍历层数就是答案。

最小公倍数
LCM

- **事实**:

$$\forall a,b>0, a,b\in\mathbb{N}, a\times b=\mathsf{gcd}(a,b)\times \mathrm{lcm}(a,b)$$

- 所以求出俩数的 GCD 后 LCM 可在 $O(1)$ 的时间内求出

最小公倍数

LCM

- **事实**：

$$\forall a, b > 0, a, b \in \mathbb{N}, a \times b = \text{gcd}(a, b) \times \text{lcm}(a, b)$$

- **所以求出俩数的 GCD 后 LCM 可在 O(1) 的时间内求出**
- **算法实现**：

```
int lcm(int a, int b) {
  return a * b / __gcd(a, b);
}
```

最小公倍数
LCM

- **事实**:

$$\forall a, b > 0, a, b \in \mathbb{N}, a \times b = \mathrm{gcd}(a, b) \times \mathrm{lcm}(a, b)$$

- 所以求出俩数的 GCD 后 LCM 可在 O(1) 的时间内求出
- 算法实现:

```
int lcm(int a, int b) {
  return a * b / __gcd(a, b);
}
```

- 多个数的 LCM
  - 与多个数的 GCD 类似

最小公倍数

LCM

- **事实**：
  $$\forall a, b > 0, a, b \in \mathbb{N}, a \times b = \mathsf{gcd}(a, b) \times \mathrm{lcm}(a, b)$$
- 所以求出俩数的 GCD 后 LCM 可在 O(1) 的时间内求出
- 算法实现：

```
int lcm(int a, int b) {
  return a * b / __gcd(a, b);
}
```

- **多个数的** LCM
  - 与多个数的 GCD 类似
  - 每次取出两个数求 LCM 后再将 LCM 放回去，不会对所需要的答案造成影响

例题

最大公约数和最小公倍数问题

题目链接：[NOIP2001 普及组] 最大公约数和最小公倍数问题

题目描述

输入两个正整数 x_0，y_0，$(2 \leq x_0, y_0 \leq 10^5)$，求出满足下列条件的 P，Q 的个数：

- P，Q 是正整数。
- 要求 P，Q 以 x_0 为最大公约数，以 y_0 为最小公倍数。

试求：满足条件的所有可能的 P，Q 的个数。

例题

最大公约数和最小公倍数问题

题目链接：[NOIP2001 普及组] 最大公约数和最小公倍数问题

题目描述

输入两个正整数 x_0, y_0，$(2 \le x_0, y_0 \le 10^5)$，求出满足下列条件的 P，Q 的个数：

- P，Q 是正整数。
- 要求 P，Q 以 x_0 为最大公约数，以 y_0 为最小公倍数。

试求：满足条件的所有可能的 P，Q 的个数。

题解

- 根据 $P \times Q = \mathrm{gcd}(P, Q) \times \mathrm{lcm}(P, Q) = x_0 y_0$，我们可以得到 $P \times Q$ 的值
- 枚举 $i | (x_0 y_0)$，考虑 $i, \lfloor \frac{x_0 y_0}{i} \rfloor$ 的最大公约数和最小公倍数是不是分别为 P, Q 即可

例题

最大公约数和最小公倍数问题

题目链接：[NOIP2001 普及组] 最大公约数和最小公倍数问题

题目描述

输入两个正整数 x_0, y_0，$(2 \le x_0, y_0 \le 10^5)$，求出满足下列条件的 P，Q 的个数：

- P，Q 是正整数。
- 要求 P，Q 以 x_0 为最大公约数，以 y_0 为最小公倍数。

试求：满足条件的所有可能的 P，Q 的个数。

题解

- 根据 $P \times Q = \gcd(P, Q) \times \operatorname{lcm}(P, Q) = x_0 y_0$，我们可以得到 $P \times Q$ 的值
- 枚举 $i | (x_0 y_0)$，考虑 $i, \lfloor \frac{x_0 y_0}{i} \rfloor$ 的最大公约数和最小公倍数是不是分别为 P, Q 即可
- 时间复杂度：$O(n \log n)$

拓展欧几里得算法

简介

- 拓展欧几里得算法用于求解 $\mathrm{ax}+\mathrm{by}=\mathrm{gcd}(\mathrm{a},\mathrm{b})$ 的一组可行解

拓展欧几里得算法
简介

- 拓展欧几里得算法用于求解 $\mathrm{ax} + \mathrm{by} = \mathrm{gcd}(\mathrm{a}, \mathrm{b})$ 的一组可行解
- 常用于求解**乘法逆元**

拓展欧几里得算法

简介

- 拓展欧几里得算法用于求解 $ax + by = \gcd(a, b)$ 的一组可行解
- 常用于求解**乘法逆元**
- **裴蜀定理**：该方程组的解一定存在

定理（裴蜀定理）

设 a, b 是不全为零的整数，则存在整数 x, y，使得 $ax + by = \gcd(a, b)$。

拓展欧几里得算法
简介

- 拓展欧几里得算法用于求解 $ax+by=\gcd(a,b)$ 的一组可行解
- 常用于求解**乘法逆元**
- **裴蜀定理**：该方程组的解一定存在

定理（裴蜀定理）

设 a,b 是不全为零的整数，则存在整数 x,y，使得 $ax+by=\gcd(a,b)$。

- **算法思想**：将问题 (a,b) 转化为 $(b, a \bmod b)$

拓展欧几里得算法

简介

- 拓展欧几里得算法用于求解 $ax + by = \gcd(a, b)$ 的一组可行解
- 常用于求解**乘法逆元**
- **裴蜀定理**：该方程组的解一定存在

定理（裴蜀定理）

设 a, b 是不全为零的整数，则存在整数 x, y，使得 $ax + by = \gcd(a, b)$。

- **算法思想**：将问题 (a, b) 转化为 $(b, a \bmod b)$
- 要注意负值

拓展欧几里得算法

证明

Proof.

拓展欧几里得算法

证明

Proof.

$$
\begin{aligned}
ax_1 + by_1 &= \gcd(a, b) \\
bx_2 + (a \bmod b)y_2 &= \gcd(b, a \bmod b)
\end{aligned}
$$

拓展欧几里得算法
证明

Proof.

$$
\begin{aligned}
ax_1 + by_1 &= \gcd(a, b) \\
bx_2 + (a \bmod b)y_2 &= \gcd(b, a \bmod b)
\end{aligned}
$$

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

拓展欧几里得算法
证明

Proof.

$$
\begin{aligned}
ax_1 + by_1 &= \gcd(a, b) \\
bx_2 + (a \bmod b)y_2 &= \gcd(b, a \bmod b)
\end{aligned}
$$

$$
\begin{aligned}
\because \gcd(a, b) &= \gcd(b, a \bmod b) \\
\therefore ax_1 + by_1 &= bx_2 + (a \bmod b)y_2
\end{aligned}
$$

拓展欧几里得算法

证明

Proof.

$$
\begin{aligned}
ax_1 + by_1 &= \gcd(a, b) \\
bx_2 + (a \bmod b)y_2 &= \gcd(b, a \bmod b)
\end{aligned}
$$

$$
\begin{aligned}
\because \gcd(a, b) &= \gcd(b, a \bmod b) \\
\therefore ax_1 + by_1 &= bx_2 + (a \bmod b)y_2 \\
\because a \bmod b &= a - (\lfloor \frac{a}{b} \rfloor \times b)
\end{aligned}
$$

拓展欧几里得算法

证明

Proof.

$$\begin{aligned} ax_1 + by_1 &= \gcd(a, b) \\ bx_2 + (a \bmod b)y_2 &= \gcd(b, a \bmod b) \end{aligned}$$

$$\begin{aligned} \because \gcd(a, b) &= \gcd(b, a \bmod b) \\ \therefore ax_1 + by_1 &= bx_2 + (a \bmod b)y_2 \\ \because a \bmod b &= a - (\lfloor \frac{a}{b} \rfloor \times b) \\ \therefore ax_1 + by_1 &= bx_2 + (a - (\lfloor \frac{a}{b} \rfloor \cdot b))y_2 \\ &= ay_2 + bx_2 - \lfloor \frac{a}{b} \rfloor \cdot by_2 \end{aligned}$$

拓展欧几里得算法

证明

Proof.

$$\begin{aligned} ax_1 + by_1 &= \gcd(a,b) \\ bx_2 + (a \bmod b)y_2 &= \gcd(b, a \bmod b) \end{aligned}$$

$$\begin{aligned} \because \gcd(a,b) &= \gcd(b, a \bmod b) \\ \therefore ax_1 + by_1 &= bx_2 + (a \bmod b)y_2 \\ \because a \bmod b &= a - (\lfloor \frac{a}{b} \rfloor \times b) \\ \therefore ax_1 + by_1 &= bx_2 + (a - (\lfloor \frac{a}{b} \rfloor \cdot b))y_2 \\ &= ay_2 + bx_2 - \lfloor \frac{a}{b} \rfloor \cdot by_2 \\ \therefore ax_1 + by_1 &= ay_2 + b(x_2 - \lfloor \frac{a}{b} \rfloor y_2) \end{aligned}$$

拓展欧几里得算法

证明

Proof.

$$
\begin{aligned}
ax_1+by_1 &= \gcd(a,b) \\
bx_2+(a \bmod b)y_2 &= \gcd(b, a \bmod b)
\end{aligned}
$$

$$
\begin{aligned}
\because \gcd(a,b) &= \gcd(b, a \bmod b) \\
\therefore ax_1+by_1 &= bx_2+(a \bmod b)y_2 \\
\because a \bmod b &= a-(\lfloor \frac{a}{b} \rfloor \times b) \\
\therefore ax_1+by_1 &= bx_2+(a-(\lfloor \frac{a}{b} \rfloor \cdot b))y_2 \\
&= ay_2+bx_2-\lfloor \frac{a}{b} \rfloor \cdot by_2 \\
\therefore ax_1+by_1 &= ay_2+b(x_2-\lfloor \frac{a}{b} \rfloor y_2) \\
\because a=a, &\quad b=b \\
\therefore x_1=y_2, &\quad y_1=x_2-\lfloor \frac{a}{b} \rfloor y_2
\end{aligned}
$$

□

拓展欧几里得算法
实现

- $x_1 = y_2, y_1 = x_2 - \lfloor \frac{a}{b} \rfloor y_2$
- 其中 x_1, y_1 是问题 (a, b) 的答案
  x_2, y_2 是问题 $(b, a \bmod b)$ 的答案

拓展欧几里得算法

实现

- $x_1 = y_2, y_1 = x_2 - \lfloor \frac{a}{b} \rfloor y_2$
- 其中 x_1, y_1 是问题 (a, b) 的答案
  x_2, y_2 是问题 $(b, a \bmod b)$ 的答案
- 算法实现：

```
ll exgcd(ll a, ll b, ll& x, ll& y){
  if (!b){
      x = 1, y = 0;
      return a;
  }
  ll gcd = exgcd(b, a % b, y, x);
  y -= a / b * x;
  return gcd;
}
```

例题
同余方程

题目链接：[NOIP2012 提高组] 同余方程

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**。
数据保证一定有解。
$2 \le a, b \le 2e9$

例题

同余方程

题目链接：[NOIP2012 提高组] 同余方程

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**。
数据保证一定有解。
$2 \leq a, b \leq 2e9$

题解

- $ax \equiv 1 \pmod b$ 可以转化为 $ax = (-b)y + 1$，即 $ax + by = 1$。

例题
同余方程

题目链接：[NOIP2012 提高组] 同余方程

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**。
数据保证一定有解。
$2 \le a, b \le 2e9$

题解

- $ax \equiv 1 \pmod b$ 可以转化为 $ax = (-b)y + 1$，即 $ax + by = 1$。
- 拓展欧几里得裸题。

例题
同余方程

题目链接：[NOIP2012 提高组] 同余方程

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**。
数据保证一定有解。
$2 \le a, b \le 2e9$

题解

- $ax \equiv 1 \pmod b$ 可以转化为 $ax = (-b)y + 1$，即 $ax + by = 1$。
- 拓展欧几里得裸题。
- 时间复杂度：$O(\log a)$

例题

同余方程

题目链接：[NOIP2012 提高组] 同余方程

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**。
数据保证一定有解。
$2 \le a, b \le 2e9$

题解

- $ax \equiv 1 \pmod b$ 可以转化为 $ax = (-b)y + 1$，即 $ax + by = 1$。
- 拓展欧几里得裸题。
- 时间复杂度：$O(\log a)$

拓展欧几里得习题推荐：

- 青蛙的约会

Questions?