Lesson1

质数（从2开始定义）

定义：在大于1的整数中，如果只包含1和本身两个约数，即为质数或者素数

性质

- 从2开始的整数定义
- 所有小于2的数既不是质数也不是合数

质数的判定

- 试除法：时间复杂度O(sqrt(n))
- e1: 试除法判定质数

```java
static boolean isPrime(int x) {
    if (x < 2) return false;
    else
        for (int i=2; i<=x/i; i++)
            if (x % i == 0) return false;

    return true;
}


public static void main(String[] args) throws Exception {
    ins.nextToken(); int n = (int)ins.nval;

    while (n-- > 0) {
        ins.nextToken(); int a = (int)ins.nval;
        out.println((isPrime(a)? "Yes": "No"));
    }

    out.flush();
}
```

分解质因数

- 试除法

方法：从小到大枚举所有数

性质：对于一个数x，x中最多只包含一个大于sqrt(x)的质因子，以此优化时间复杂度

时间复杂度：O(logn)~O(sqrt(n))

- e2：分解质因数

```java
static void divide(int x) {
```

```
    for (int i=2; i<=x/i; i++)
        if (x % i == 0) {
            int s = 0;
            while (x % i == 0) {
                x /= i; s++;
            }
            out.println(i+" "+s);
        }

    if (x > 1) out.println(x+" 1");
    out.println();
}


public static void main(String[] args) throws Exception {
    ins.nextToken(); int n = (int)ins.nval;

    while (n-- > 0) {
        ins.nextToken(); int a = (int)ins.nval;
        divide(a);
    }

    out.flush();
}
```

筛质数

朴素筛法

- 思想：从2往后一直将所有数的倍数全部删掉
- 时间复杂度计算：调和级数

```
static int N = 1000010;

static int n;
static int[] primes = new int[N];
static int cnt;
static boolean[] st = new boolean[N];

// 朴素筛法，时间复杂度O(nlnn)
static void getPrimes(int n) {
    for (int i=2; i<=n; i++) {
        if (!st[i]) primes[cnt++] = i;

        for (int j=i+i; j<=n; j+=i) st[j] = true;
    }
}


public static void main(String[] args) throws Exception {
    ins.nextToken(); n = (int)ins.nval;

    getPrimes(n);
```

```
    out.println(cnt);

    out.flush();
}
```

埃氏筛法

- 质数定理：1~n中有n/lnn个质数
- 时间复杂度：O(nloglogn)，基本和O(n)一个级别

```
static int N = 1000010;

static int n;
static int[] primes = new int[N];
static int cnt;
static boolean[] st = new boolean[N];

// 埃式筛法，时间复杂度O(nloglogn)
static void getPrimes(int n) {
    for (int i=2; i<=n; i++) {
        if (!st[i]) {
            primes[cnt++] = i;

            for (int j=i+i; j<=n; j+=i) st[j] = true;
        }
    }
}
```

线性筛法

正确性证明：

- 每一个数x，只会被其最小质因子筛掉

```
从小到大枚举每一个质数
1. i % p[j] == 0;   //p[j]一定是i的最小质因子，且p[j]也一定是p[j]*i的最小质因子
2. i % p[j] != 0;   //p[j]一定小于i的所有质因子，所以p[j]也一定是p[j]*i的最小质因子
```

- 任意一个合数x，一定会被其最小质因数筛掉。

对于一个合数x，假设pj是x的最小质因子，当i枚举到x / pj时，其就会被筛掉

具体实现：时间复杂度O(n)

```
//线性筛法，时间复杂度O(n)
void get_primes(int n) {
    for (int i=2; i<=n; i++) {
        if (!st[i]) primes[cnt++] = i;

        //从小到大枚举所有质数
```

```
        //且primes[j]的最大值为min(n/i, i);
        //当i比n/i小时，i控制循环次数（循环i次时所有比i小的质数都已存入primes数组）
        //故循环次数不会超过cnt的范围
        //当i比n/i大时，n/i控制循环次数。primes[j]仍比i小，不会越界
        for (int j=0; primes[j]<=n/i; j++) {
            st[primes[j]*i] = true;

            if (i%primes[j] == 0) break;
        }
    }
}
```

约数

试除法求一个数的所有约数

性质

- 在1~n中，所有的约数个数为nlnn即nlogn级别，故每个数平均约数个数为logn个
- int范围内，约数个数最多的数含有1500个左右约数

具体实现：时间复杂度O(sqrt(n))

```
static List<Integer> getDivisors(int x) {
    List<Integer> res = new ArrayList<>();

    for (int i=1; i<=x/i; i++) {
        if (x % i == 0) {
            res.add(i);

            if (i != x/i) res.add(x/i);
        }
    }

    res.sort((o1, o2) -> o1-o2);
    return res;
}

public static void main(String[] args) throws Exception {
    ins.nextToken(); int n = (int)ins.nval;

    while (n-- > 0) {
        ins.nextToken(); int x = (int)ins.nval;

        List<Integer> res = getDivisors(x);

        for (int p: res) out.print(p+" ");
        out.println();
    }

    out.flush();
```

```
}
```

约数个数

计算公式：(a1+1)(a2+1)...(an+1)，其中a1, a2...an为原数分解质因数后每一个质因子的指数，可用算数基本定理进行证明

```
static int mod = (int)1e9+7;

static Map<Integer, Integer> primes = new HashMap<>();

public static void main(String[] args) throws Exception {
    ins.nextToken(); int n = (int)ins.nval;

    while (n-- > 0) {
        ins.nextToken(); int x = (int)ins.nval;

        for (int i=2; i<=x/i; i++)
            if (x % i == 0)
                while (x % i == 0) {
                    primes.put(i, primes.getOrDefault(i, 0)+1);
                    x /= i;
                }

        if (x > 1) primes.put(x, primes.getOrDefault(x, 0)+1);
    }

    long res = 1;    //注意数据范围
    for (int v: primes.values())
        res = res*(v+1) % mod;

    out.println(res);

    out.flush();
}
```

约数之和

计算公式：(p1^0+p1^1+...+p1^a1)(p2^0+p2^1+...+p2^a2)...(pn^0+pn^1+...+pn^an)

```
static int mod = (int)1e9+7;

static Map<Integer, Integer> primes = new HashMap<>();

public static void main(String[] args) throws Exception {
    ins.nextToken(); int n = (int)ins.nval;

    while (n-- > 0) {
        ins.nextToken(); int x = (int)ins.nval;

```

```java
        for (int i=2; i<=x/i; i++)
            if (x % i == 0)
                while (x % i == 0) {
                    primes.put(i, primes.getOrDefault(i, 0)+1);
                    x /= i;
                }

        if (x > 1) primes.put(x, primes.getOrDefault(x, 0)+1);
    }

    long res = 1;
    for (int p: primes.keySet()) {
        int v = primes.get(p);

        long t = 1;
        while (v-- > 0) t = (t*p+1) % mod;

        res = res*t % mod;
    }

    out.println(res);

    out.flush();
}
```

最大公约数（欧几里得算法，辗转相除法）

预备知识

- 任何数 | 0
- d|a, d|b -> d|(a+b), d|(ax+by)
- (a, b) = (b, a mod b)

  证明：

  a mod b = a - a/b*b = a - c * b

  (a, b) = (b, a-c*b)

  从左推右，因为d|a, d|b，所以d|b, d|(a-c*b)

  从右推坐，d|b, d|(a-c*b)，所以d|b, d | (a - c * b + c * b) = d|a

  所以综上，(a, b) = (b, a mod b)

具体实现：时间复杂度O(logn)

```java
static int gcd(int a, int b) {
    return b!=0? gcd(b, a % b): a;
}


public static void main(String[] args) throws Exception {
    ins.nextToken(); int n = (int)ins.nval;

    while (n-- > 0) {
```

```
        ins.nextToken(); int a = (int)ins.nval;
        ins.nextToken(); int b = (int)ins.nval;

        out.println(gcd(a, b));
    }

    out.flush();
}
```