

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

```
1  #include <iostream>
2
3  using namespace std;
4
5  bool is_prime(int x) {
6      if (x < 2) return false;
7      else
8          //不推荐写成i*i<=x，当x接近int最大值时存在溢出风险
9          //也不建议写成i <= sqrt(x)，sqrt函数运算较慢
10         for (int i=2; i <= x/i; i++)
11             if (x%i == 0) return false;
12
13     return true;
14 }
15
16 int main(void) {
17     int n;
18     scanf("%d", &n);
19
20     while (n--) {
21         int a;
22         scanf("%d", &a);
23
24         if (is_prime(a)) puts("Yes");
25         else puts("No");
26     }
27
28     return 0;
29 }
```

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

```
1  #include <iostream>
2
3  using namespace std;
4
5  void divide(int x) {
6      for (int i=2; i<=x/i; i++) {
7          if (x%i == 0) {
8              int s = 0;
9              while (x%i == 0) x /= i, s++;
10             printf("%d %d\n", i, s);
11         }
12     }
13
14     //x中最多只包含一个大于sqrt(x)的质因子
15     if (x > 1) printf("%d 1\n", x);
16     puts("");
17 }
18
19 int main(void) {
```

```
20     int n;
21     scanf("%d", &n);
22
23     while (n--) {
24         int a;
25         scanf("%d", &a);
26         divide(a);
27     }
28
29     return 0;
30 }
```

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

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 1000010;
6
7  int n;
8  int primes[N], cnt;
9  bool st[N];
10
11  //朴素筛法，时间复杂度O(nlnn)，可视为O(nlogn)
12  void get_primes(int n) {
13      for (int i=2; i<=n; i++) {
14          if (!st[i]) primes[cnt++] = i;
15
16          for (int j=i+i; j<=n; j+=i) st[j] = true;
17      }
18  }
19
20  int main(void) {
21      scanf("%d", &n);
22
23      get_primes(n);
24
25      printf("%d", cnt);
26
27      return 0;
28  }
```

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

```
1  int primes[N], cnt;
2  bool st[N];
3
4  //埃氏筛法，时间复杂度O(nloglogn)，可视为O(n)
5  void get_primes(int n) {
6      for (int i=2; i<=n; i++) {
7          if (!st[i]) {
8              primes[cnt++] = i;
9
10             for (int j=i+i; j<=n; j+=i)
11                 st[j] = true;
12         }
13     }
14 }
15
```

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

```
1  从小到大枚举每一个质数
2  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
3  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)

```
1 //线性筛法，时间复杂度O(n)
2 void get_primes(int n) {
3     for (int i=2; i<=n; i++) {
4         if (!st[i]) primes[cnt++] = i;
5
6         //从小到大枚举所有质数
7         //且primes[j]的最大值为min(n/i, i);
8         //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）
9         //故循环次数不会超过cnt的范围
10        //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
11        for (int j=0; primes[j]<=n/i; j++) {
12            st[primes[j]*i] = true;
13
14            if (i%primes[j] == 0) break;
15        }
16    }
17 }
```

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))

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 using namespace std;
6
7 vector<int> get_divisors(int x) {
8     vector<int> res;
9
10    for (int i=1; i<=x/i; i++) {
11        if (x%i == 0) {
12            res.push_back(i);
13            //特判
14            if (i != x/i) res.push_back(x/i);
15        }
16    }
17
18    sort(res.begin(), res.end());
19
20    return res;
21 }
22
23 int main(void) {
24     int n;
25     scanf("%d", &n);
26
27     while (n--) {
28         int a;
29         scanf("%d", &a);
30
31         auto res = get_divisors(a);
32
33         for (auto t: res) cout << t << " ";
34         puts("");
35     }
36
37     return 0;
38 }
```

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

```

1  #include <iostream>
2  #include <unordered_map>
3  #include <algorithm>
4
5  using namespace std;
6
7  typedef long long LL;
8
9  const int mod = 1e9+7;
10
11 int main(void) {
12     int n;
13     scanf("%d", &n);
14
15     //哈希表
16     unordered_map<int, int> primes;
17
18     //质因数分解
19     while (n--) {
20         int x;
21         scanf("%d", &x);
22
23         for (int i=2; i<=x/i; i++)
24             while (x%i == 0)
25                 x /= i, primes[i]++;
26
27         if (x > 1) primes[x]++;
28     }
29
30     LL res = 1;
31     for (auto p: primes) res = res*(p.second+1)%mod;
32
33     cout << res << endl;
34
35     return 0;
36 }

```

约数之和

计算公式: $(p_1^0+p_1^1+\dots+p_1^{a_1})(p_2^0+p_2^1+\dots+p_2^{a_2})\dots(p_n^0+p_n^1+\dots+p_n^{a_n})$

```

1  #include <iostream>
2  #include <algorithm>
3  #include <unordered_map>
4
5  using namespace std;
6
7  typedef long long LL;
8
9  const int mod = 1e9+7;
10
11 int main(void) {
12     int n;
13     scanf("%d", &n);
14
15     unordered_map<int, int> primes;
16
17     while (n--) {
18         int x;
19         scanf("%d", &x);
20
21         for (int i=2; i<=x/i; i++)
22             while (x%i == 0)
23                 x /= i, primes[i]++;
24
25         if (x > 1) primes[x]++;
26     }
27
28     LL res = 1;
29     for (auto prime: primes) {
30         LL p = prime.first, a = prime.second;
31
32         //快速计算p^1+p^2+...+p^ak
33         int t = 1;
34         while (a--) t = (t*p+1)%mod;
35
36         res = res*t%mod;
37     }

```

```
38
39     cout << res;
40
41     return 0;
42 }
```

最大公约数（欧几里得算法，辗转相除法）

预备知识

- 任何数 $| 0$
- $d|a, d|b \rightarrow d|(a+b), d|(ax+by)$
- $(a, b) = (b, a \bmod b)$

证明：

$$a \bmod b = a - a/b * b = a - c * b$$

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

从左推右，因为 $d|a, d|b$ ，所以 $d|b, d|(a - c * b)$

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

所以综上， $(a, b) = (b, a \bmod b)$

具体实现：时间复杂度 $O(\log n)$

```
1  #include <iostream>
2
3  using namespace std;
4
5  int gcd(int a, int b) {
6      return b ? gcd(b, a%b) : a;
7  }
8
9  int main(void) {
10     int n;
11     scanf("%d", &n);
12
13     while (n--) {
14         int a, b;
15         scanf("%d%d", &a, &b);
16         printf("%d\n", gcd(a, b));
17     }
18
19     return 0;
20 }
```