2020年ICPC集训---素数和约数

曾雪强

江西师范大学计算信息工程学院
xqzeng@jxnu.edu.cn

素数

- 素数（prime number）：在大于1的自然数中，除了1和它本身没有其他约数，这样的数称为素数；否则称为合数。
  - 素数又称质数，有无限个；2、3、5、7、11、13、17、19...
  - 素数p的约数只有两个：1 和 p
  - 0和1既不是素数也不是合数，最小的素数是2
- 素数判定：
  - 试除法：如果自然数n不能被 $[2, \sqrt{n}]$ 内的所有素数整除，那么n是素数
  - 加速技巧：大于4的素数总是等于 6x+1 或 6x-1
  - 如何证明？ 约数一定是成对出现的，d 和 n/d

更快的素数判定算法：Miller-Rabin算法，费马小定理 $x^p=x(\mathrm{mod}\ p)$

- 素数判定函数（试除法）

```
bool isPrime(int num) {
    if (num <= 3)  return num>1;

    // 不在6的倍数两侧的一定不是质数
    if (num % 6 != 1 && num % 6 != 5)
        return false;

    int s = (int) sqrt(num);
    for (int i = 5; i <= s; i += 6) {
        if (num % i == 0 || num % (i+2) == 0) {
            return false;
        }
    }
    return true;
}
```

素数的筛选

- 素数埃式筛法：求区间 [2, n] 内所有的素数
  - 初始数列：{2、3，4，5，6，7，8，9，10，11，12，13，…，n}
  - 输出最小的素数 {2}，筛掉2的倍数，剩下 {3，5，7，9，11，13，…}
  - 输出素数 {2,3}，筛掉3的倍数，剩下 {5，7，11，13，…}
  - 输出素数 {2,3,5}，筛掉5的倍数，剩下 {7，11、13，…}
  - 继续以上步骤，直到n。
- 埃式筛法的时间复杂度：O(n*log log n)
  - 筛选次数：n/2+n/3+…+n/n
  - 调和级数：$\sum_{i=1}^{n} {}^{1}/_{i} = \ln n + \gamma + \varepsilon_n$
  - 素数分布：π(x) ≈ x/lnx

- 素数线性筛法（欧拉筛法，Euler's Sieve）
  - 每个合数仅被它最小的质因数筛去，时间复杂度O(n)

```
const int N=1000001;
int primes[N], cnt;
bool st[N];
void getPrimes(int n){
    memset(st, false, sizeof st);
    cnt=0;
    for(int i=2; i<=n; i++) {
        if(!st[i])  primes[cnt++]=i;
        for(int j=0; j<cnt && i*primes[j]<=n; j++) {
            st[ i*primes[j] ] = true;
            if( i%primes[j]==0 )  break;
        }
    }
}
```

- 例题：素数距离 (POJ-2689) https://www.acwing.com/problem/content/198/
  - 给定两个整数L和U，需要在闭区间 [L,U] 内找到距离最接近的两个相邻素数C_1和C_2（即C_2-C_1是最小的），如果存在相同距离的其他相邻素数对，输出第一对。同时，还需要找到距离最远的两个相邻素数D_1和D_2（即D_2-D_1是最大的），如果存在相同距离的其他素数对，输出第一对。
  - 输入格式
    - 每行输入两个整数L和U，其中L和U的差值不会超过1000000。
  - 输出格式
    - 对于每个L和U，输出一个结果，结果占一行。
    - 结果包括距离最近的相邻素数对和距离最远的相邻素数对。（具体格式参照样例）
    - 如果L和U之间不存在素数对，则输出“There are no adjacent primes.”
  - 数据范围
    - $1 \le L < U \le 2^{31}-1$
  - 输入样例：
    - 2 17
    - 14 17
  - 输出样例：
    - 2,3 are closest, 7,11 are most distant.
    - There are no adjacent primes.

素数距离题解

- 据题意，要算出 [L,R] 之间的所有素数，那么要做素数筛选。困难是 L 和 R 特别大，无法直接求出 [1,R] 中的所有素数。
- 注意到：任何一个合数n，一定有小于等于$\sqrt{n}$的约数。
  - $\sqrt{2^{31}}$=46000+
- 所以，只需事先枚举出 2~$\sqrt{R}$ 之间的所有素数；基于这些素数，就可以筛选出 [L,R] 之间的所有素数。
- 然后，把 [L,R] 区间内的素数保存成一个数组。最后就可以通过一次遍历，找出满足条件的素数对。
- 易出的错误：
  - 给定素数p，[L,R] 之间第一个p的倍数是多少?
  - 个别变量会出整数越界错误，需要用 long long 类型
  - 要对 L=1 的情况做特殊处理

```
#include <iostream>
#include <cstring>
using namespace std;

typedef long long LL;
int p[1000001];
bool flag[1000001];
int cnt;

void getPrimes(int n) {...}

int main(){
    int L,U;
    while(cin>>L>>U){
        if(L<=1) L=2;

        getPrimes(1000001);

        memset(flag, false, sizeof flag);
```

```
        for(int i=0; i<cnt && p[i]<U; i++){
            for(LL j=(L+0ll+p[i]-1)/p[i]*p[i]; j<=U; j+=p[i]){
                if(j!=p[i])  flag[j-L]=true;
            }
        }

        cnt=0;
        for(int i=0;i<=U-L;i++){
            if(!flag[i])  p[cnt++]=L+i;
        }

        if(cnt<2) {
            cout<<"There are no adjacent primes."<<endl;
        }
        else {
            int minI=0,maxI=0;
            for(int i=1;i<cnt-1;i++) {...}
            printf("%d,%d are closest, %d,%d are most
distant.\n", p[minI],p[minI+1],p[maxI],p[maxI+1]);
        }} return 0; }
```

约数

- 若整数n除以整数d的余数为0，即d能整除n，则称d是n的约数，n是d的倍数，记为 d|n 。
- 唯一分解定理（算数基本定理）：任一大于1的自然数N，都可以唯一分解为有限个素数之积：$N = p_1^{c_1} p_2^{c_2} \dots p_r^{c_r}$
- N的正约数集合可写作：$\{p_1^{b_1} p_2^{b_2} \dots p_m^{b_m}\}$，其中$0 \le b_i \le c_i$
  - N的正约数个数为 $(c_1+1)*(c_2+1)*\cdots*(c_m+1) = \prod_{i=1}^{m}(c_i+1)$
  - N的所有的正约数之和为

$$\left(1+p_1+p_1^2+\cdots+p_1^{c_1}\right)*\left(1+p_2+p_2^2+\cdots+p_2^{c_2}\right)*\cdots*\left(1+p_m+p_m^2+\cdots+p_m^{c_m}\right)$$

$$=\prod_{i=1}^{m}\left(\sum_{j=0}^{c_i}(p_i)^j\right)$$

- 例题：Aladdin and the Flying Carpet (LightOJ-1341)
  - Given the area of the carpet and the length of the minimum possible side of the carpet, your task is to find how many types of carpets are possible. The carpet was rectangular shaped, but not square shaped.
    For example, the area of the carpet 12, and the minimum possible side of the carpet is 2, then there can be two types of carpets and their sides are: {2, 6} and {3, 4}.
  - Input
    - Input starts with an integer T (≤4000), denoting the number of test cases.
    - Each case starts with a line containing two integers: a b($1 \le b \le a \le 10^{12}$) where a denotes the area of the carpet and b denotes the minimum possible side of the carpet.
  - Output
    - For each case, print the case number and the number of possible carpets.

Aladdin and the Flying Carpet 题解

- 题意：有 t 组数据，每组给出 2 个数 a、b，求满足 c*d=a 且 c>=b、d>=b 的二元组的个数，其中 (c,d) 与 (d,c) 视为同一种情况
- 对每一组数据，将面积a分解为 $a = p_1^{c_1} p_2^{c_2} \ldots p_r^{c_r}$
- a的约数个数为$\prod_{i=1}^{m}(c_i + 1)$，因为约数是对称的，所以要除以2
- $\prod_{i=1}^{m}(c_i + 1)/2$ 减去 所有 <b 的约数的个数，就是答案

反素数

- 反素数
  - 对于正整数x，其约数的个数记作g(x)。例如 g(1)=1、g(6)=4。
  - 如果某个正整数x满足：对任意正整数 i(0<i<x)，都有g(x)>g(i)，那么称x为反素数。
- 考虑 x 是小于 n 的最大的反素数的情况：　　1.....x....n
  - x是 小于n的 约数最多的 数
  - 当约数个数相同时，x一定是最小的数
- 反素数是约数最多的最小数，$x = p_1^{e_1} p_2^{e_2} \ldots p_r^{e_r}$ 的两个性质
  - 质因子 $p_1, p_2, \ldots, p_r$ 是从2开始连续的素数.
  - 质因子的指数递减 $e_1 \geq e_2 \geq e_3 \ldots \geq e_r$

- 例题：反素数 (BZOJ-1053) https://www.acwing.com/problem/content/description/200/
  - 对于任何正整数x，其约数的个数记作g(x)，例如g(1)=1、g(6)=4。如果某个正整数x满足：对于任意的小于x的正整数 i，都有g(x)>g(i) ，则称x为反素数。例如，整数1，2，4，6等都是反素数。现在给定一个数N，请求出不超过N的最大的反素数。
  - 输入格式
    - 一个正整数N。
  - 输出格式
    - 一个整数，表示不超过N的最大反素数。
  - 数据范围
    - $1 \le N \le 2*10^9$
  - 输入样例：
    - 1000
  - 输出样例：
    - 840

- 反素数题解
  - 反素数的不同的质因子不会超过10个，因为 $2*3*5*7*11*13*17*19*23*29*31>2*10^9$
  - 反素数的所有质因数的指数总和不超过30，因为$2^{31}>2*10^9$
  - 所以，可以使用递归DFS，暴力搜索前十个质数的指数的所有可能组合
  - 搜索时需满足指数单调递减，总乘积不超过N，且同时记录当前约数的个数

```
#include <iostream>
using namespace std;
typedef long long LL;
int ps[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
int n; int sum=0, minx;

//u:待搜索的质因子序号, last:待搜索质因子的指数最大值
//p:约数之积, s:约数个数
void dfs(int u, int last, int p, int s){
    if (s > sum || s == sum && p < minx){ //更新反素数
        sum = s;
        minx = p;
    }

    for (int i = 1; i <= last; i++ ) {
        if ( (LL)p*ps[u] > n ) break;
        p *= ps[u];
        dfs( u+1, i, p, s*(i+1) );
    }
}
```

```
int main(){
    cin >> n;

    dfs(0, 30, 1, 1);

    cout << minx << endl;

    return 0;
}
```