Теория чисел

Плотников Даниил Михайлович, Закарлюка Иван Владимирович

Санкт-Петербургский Государственный Университет

Оглавление

Делители

Представление чилса

Математическое определение $\forall x, y \ \exists a, b : y > b \geq 0 \Rightarrow x = ay + b$. В этом определении b — остаток от деления числа. По этому для $x = -10, y = 3$ коректным выражением является $-10 = -4 \times 3 + 2$

В c++ нет ограничения на знак остатка! При этом a является минимальным по модулю. Таким образом $-10\%3$ вернёт результат -1. Чтобы это обойти приходится писать $(a\%b + b)\%b$

Делители

Если $b = 0$, то говорят что a и y называют **делителями числа**. Пишется как $b \vdots a$ или $a|b$

Програмно проверку числа на делимость осуществить просто: $a\%b == 0$ либо $!(a\%b)$

Поскольку $a << 2$ и $a/2$, то можно писать $!(a\&1)$, что является более быстрой проверкой на чётность.

Нахождение делителей числа. Тривиальный алгоритм

```
int count_divisors(int n){
    int cnt = 0;
    for(int i = 1; i <= n; ++i){
        cnt += n%2==0;
    }
    return cnt;
}

```

Нахождение делителей числа. Более оптимальная версия

В действительности смотреть на все n чисел не имеет смысла. Рассмотрим делители числа 24. Оно делиться на $1, 2, 3, 4, 6, 8, 12, 24$. Когда мы доходим до числа 2, мы можем составить выражение $24 = 2 * 12$, то есть находя 2 мы уже можем сказать ещё один делитель. Рассмотрим все такие пары:
$(1, 24), (2, 12), (3, 8), (4, 6)$. Как понять сколько таких пар? Если мы смотрим на наименьшее число в паре, то оно не может превосходить корень числа. Если число является полным квадратом, то корень можно рассматривать как пару равных чисел, например для 36 это пара $(6, 6)$. Такие корни называются **чётными корнями**.

Время контеста

Нахождение делителей числа. Алгоритм за $O(\sqrt{n})$

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

Простые числа

Простые числа. Тривиальный алгоритм

Простое число — число которое делиться только на себя и на единицу. Единица и ноль не являются простыми числами.

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

```

Решето эратосфена

0	1	2	3	4	5	6	7	8	9	10	11	12	11	12	13	14	15	16	17
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1

Решето эратосфена

```
vector<bool> sieve(n, 1);
sieve[0] = sieve[1] = 0;
for(int i = 2; i < n; ++i){
    if(sieve[i]){
        for(int j = i*i; j < n; j+=i){
            sieve[j] = 0;
        }
    }
}

```

Время контеста

НОД и НОК

НОД и НОК

$gcd(x, y) = \min\{a \mid x \vdots a \text{ and } y \vdots a\}$

НОД - Наименьший общий делитесь. Простыми словами — наименьшее число, на которое делятся оба числа.

$lcm(x, y) = \min\{a \mid x \vdots a \text{ and } y \vdots a\}$

НОК - Наибольшее общее кратное. Простыми словами — наибольшее число, которое делиться на оба числа.

Нахождение НОД и НОК

Как найти НОД и НОК вручную?

1. Разложим оба числа в на простые делители
2.
   - Для нахождения НОК найдём объединений полученные множеств делителей
   - Для нахождения НОД найдём пересечение полученные множеств делителей
3. Перемножим все элементы множества из шага 2

НОД и НОК связаны между собой формулой $\text{НОК}(x, y) = \dfrac{x * y}{\text{НОД}(x, y)}$

Нахождение НОД и НОК

Посчитаем на примере чисел 132 и 270

132	2	270	2
66	2	135	5
33	3	27	3
11	11	9	3
1		3	3
		1	

Получили множества $\{2, 2, 3, 11\}$, $\{, 3, 3, 3, 5\}$

$gcd(132, 270) = 2 * 3 = 6$, $lcm(132, 270) = 2 * 2 * 3 * 11 * 3 * 3 * 5 = 5940$

$$lcm(132, 270) = \frac{132 * 270}{gcd(132, 270)} = \frac{35640}{6} = 5940$$

Алгоритм Евклида

```
1  int gcd(int x, int y){
2      if(y == 0){
3          return x;
4      }
5
6      if (x < y){
7          swap(x,y);
8      }
9
10     return gcd(y, x%y);
11 }
```

Работает за O(log(min(x,y)))

В стандартной библиотеке шаблонов уже реализована функция gcd(x,y);

Время контеста