Number theory

Prime factors, sieve, GCD, extended Euclid

beCP Training

April 15, 2016

Table of Contents

Prime check

Sieve of Eratosthenes

Greatest Common Divisor

Extended Euclid

Prime numbers

$$2,\ 3,\ 5,\ 7,\ 11,\ 13,\ 17,\ 19,\ 23, \ldots$$

- Prime numbers have exactly two divisors: 1 and themselves
- Numbers with three or more are called *composite*
- We can decompose any number into a product of primes
- This decomposition is *unique*

Examples:

$$6 = 2 \cdot 3 \qquad 8 = 2^3 \qquad 45 = 3^2 \cdot 5 \qquad 13 = 13$$

Prime check: linear

- How to check if n is prime?
- By checking that only 1 and n divide n.

Solution 1: $O(n)$

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

Note: we will ignore 0 and 1 here.

Prime check: stop at square root

- Actually, if d divides n, then n/d too
- Since $d \times n/d = n$, at least one of them is $\leq \sqrt{n}$
- So we only need to check up to $\sqrt{n}$

Solution 2: $O\left(\sqrt{n}\right)$

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

Factorization

- We can also factor a number with this trick
- If there are factors missing when reaching the square root, the remainder is prime

```
vector<int> factors(int n)
{
    vector<int> f;
    for (int i = 2; i*i <= n; i++) {
        while (n % i == 0) {
            f.push_back(i);
            n /= i;
        }
    }
    if (n != 1) f.push_back(n); // n is prime now
    return f;
}
```

Table of Contents

Prime check

Sieve of Eratosthenes

Greatest Common Divisor

Extended Euclid

Sieve idea

- What if we want to factorize many numbers?
- Going up to $\sqrt{n}$ for all is slow

Idea: go through the numbers one by one

- If not prime, skip it
- If prime, mark its multiples as not prime

Sieve example

1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

Sieve example

1	**2**	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

Sieve example

1	2	**3**	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

Sieve example

1	2	3	4	**5**	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

Sieve example

1	2	3	4	5	6	**7**	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

Sieve implementation

```
typedef long long ll; // to avoid overflow on i*i
bitset<MAXN> bs; bs.set(); // set to true
vector<int> primes;

for (ll i = 2; i < MAXN; i++) {
    if (bs[i]) {
        // We can start it i*i
        for (ll j = i*i; j < MAXN; j += i)
            bs[j] = false;
        primes.push_back((int) i);
    }
}
```

- Complexity: $O(n \log \log n)$ (because math)
- Practical limit: 1M $\sim$ 10M

Prime check with sieve

- If $n <$ MAXN, use the table, $O(1)$
- If $n <$ MAXN^2, use the prime list
- Complexity is $O(\sqrt{n} \,/\, \log n)$ because the density of primes is proportional to $1 \,/\, \log n$

```
// Only called for n < MAXN*MAXN
bool isPrime(int n)
{
    if (n < MAXN)
        return bs[n];
    for (int i : primes)
        if (n % i == 0)
            return false;
    return true;
}
```

Sieve variants

We can get additional information from the sieve:

- A way to get the factorization very quickly, $O(\log n)$
- The number distinct prime factors

```
int numDiffPF[MAXN], largestPF[MAXN]; // init to 0

for (ll i = 2; i < MAXN; i++) {
    if (numDiffPF[i]) {
        // This time we must start at i
        for (ll j = i; j < MAXN; j += i) {
            numDiffPF[j]++;
            largestPF[j] = i;
        }
    }
}
```

Divide by largestPF recursively to find factorization.

Table of Contents

Prime check

Sieve of Eratosthenes

Greatest Common Divisor

Extended Euclid

GCD definition

The *greatest common divisor* of a and b is the largest number that divides a and b.

Examples:

$$\gcd(4,6) = 2 \quad \gcd(7,5) = 1 \quad \gcd(24,12) = 12$$

$$\gcd(n,n) = n \quad \gcd(n,0) = n \quad \gcd(n,1) = 1$$

When using the factorization, just take the minimum of the exponents (p^0 is implied):

$$\gcd(2^5 \cdot 3^2,\ 2^4 \cdot 3^7) = 2^4 \cdot 3^2 \qquad \gcd(5^2, 3^5) = 1$$

Euclid for GCD

- Interesting property: $\gcd(a, b) = \gcd(a + bn, b)$
- In particular: $\gcd(a, b) = \gcd(a \,\%\, b, b)$
- So we can always reduce the problem to a smaller pair!

- Example:

  $$21 \quad 15 \quad 21 \,\%\, 15 = \boxed{6} \quad 15 \,\%\, 6 = \boxed{3} \quad 6 \,\%\, 3 = \boxed{0}$$

- The pairs are $(21, 15)$, $(15, 6)$, $(6, 3)$, $(3, 0)$.
- We know that $\gcd(3, 0) = 3$, so $\gcd(21, 15) = 3$.

GCD implementation

Solution: Euclid's algorithm

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

- Extremely simple
- Exercise: prove that after two iterations b is divided by at least 2 (hint: separate $a \% b \leq b/2$ and $a \% b > b/2$)
- Complexity: $O(\log b)$

Least Common Multiple

The *least common multiple* of a and b is the smallest number divisible by a and b.
It is the "opposite" of the GCD.

Examples:

$$\operatorname{lcm}(4,6) = 12 \quad \operatorname{lcm}(7,5) = 35 \quad \operatorname{lcm}(24,12) = 24$$

$$\operatorname{lcm}(n,n) = n \quad \operatorname{lcm}(n,0) = n \quad \operatorname{lcm}(n,1) = n$$

When using the factorization, just take the *maximum* of the exponents:

$$\operatorname{lcm}(2^5 \cdot 3^2,\ 2^4 \cdot 3^7) = 2^5 \cdot 3^7 \qquad \operatorname{lcm}(5^2, 3^5) = 3^5 \cdot 5^2$$

Getting LCM from GCD

- The GCD takes the minimum of the exponents and the LCM takes the maximum
- So they take both exponents!
- As a consequence, $\gcd(a, b) \times \text{lcm}(a, b) = a \times b$

Example:

$$\gcd(2^5 \cdot 3^2,\ 2^4 \cdot 3^7) = 2^4 \cdot 3^2 \qquad \text{lcm}(2^5 \cdot 3^2,\ 2^4 \cdot 3^7) = 2^5 \cdot 3^7$$

$$\gcd \times \text{lcm} = 2^{4+5} \cdot 3^{2+7} = (2^5 \cdot 3^2) \times (2^4 \cdot 3^7)$$

Solution: Just be careful with overflows!

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

Table of Contents

Prime check

Sieve of Eratosthenes

Greatest Common Divisor

Extended Euclid

Linear diophantine equation

- A diophantine equation is an equation where the solutions must be integers
- We will consider the equation $ax + by = c$ (x, y unknown)
- Note: if we had $x, y \in \mathbb{R}$ the solution would be a line
- Let $d = \gcd(a, b)$. Since d divides $ax + by$, it must also divide c.

Bézout's identity: There exist x, y such that $ax + by = d$

- So if d divides c, we can take $x(c/d)$ and $y(c/d)$
- Otherwise, no solution

The set of solutions

- Suppose we have an initial solution $ax_0 + by_0 = c$
- Then we can take $x = x_0 + (b/d)n$ and $y = y_0 - (a/d)n$ to generate all the solutions (proof in exercise, compare two solutions)

Example: $6x + 4y = 2$

- First solution: $x = 1, y = -1$
- Formula: $(x + 2n,\ y - 3n)$
- Positive n: $(3, -4), (5, -7), (7, -10) \ldots$
- Negative n: $(-1, 2), (-3, 5), (-5, 8) \ldots$

Finding the initial solution

- To find the initial solution to $ax + by = d$, we will extend Euclid's algorithm
- Base case: $b = 0$ so $x = 1, y = 0$ works
- Adapt it as we go, by reversing: $a - (a \,\%\, b) = b \cdot \lfloor a/b \rfloor$

Example: 21 15 6 3 0

- $3 = 3 \cdot 1 + 0 \cdot 0$
- $3 = 6 \cdot 0 + 3 \cdot (1 - \lfloor 6/3 \rfloor \cdot 0) = 6 \cdot 0 + 3 \cdot 1$
- $3 = 15 \cdot 1 + 6 \cdot (0 - \lfloor 15/6 \rfloor \cdot 1) = 15 \cdot 1 + 6 \cdot (-2)$
- $3 = 21 \cdot (-2) + 15 \cdot (1 - \lfloor 21/15 \rfloor \cdot (-2)) = 21 \cdot (-2) + 15 \cdot 3$

Extended Euclid: implementation

Solution: Extended Euclid's algorithm

```
int x,y,d; // global for convenience
void euclid(int a, int b) {
    // Base case
    if (b == 0) { x = 1; y = 0; d = a; return; }

    // Recurse and adapt coefficients
    euclid(b, a%b);
    int oldy = y;
    y = x - (a/b) * y;
    x = oldy;
}
```

- Complexity: $O(\log b)$, same as before
- Finds both the GCD and the coefficients

Bonus property