Codebook

Pitoni++

Žiga Gosar, Maks Kolman, Jure Slak

verzija: 1. december 2014

Kazalo

1 Teorija števil

1.1 Evklidov algoritem

Vhod: $a, b \in \mathbb{Z}$

Izhod: Največji skupni delitelj a in b. Za pozitivna števila je pozitiven, če je eno število 0, je rezultat drugo število, pri negativnih je predznak odvisen od števila iteracij.

Časovna zahtevnost: $O(\log(\max\{a, b\}))$

Prostorska zahtevnost: $O(1)$

```
int gcd(int a, int b) {
    int t;
    while (b != 0) {
        t = a % b;
        a = b;
        b = t;
    }
    return a;
}
```

1.2 Razširjen Evklidov algoritem

Vhod: $a, b \in \mathbb{Z}$,. Števili $retx$, $rety$ sta parametra samo za vračanje vrednosti.

Izhod: Števila x, y, d, pri čemer $d = \gcd(a, b)$, ki rešijo Diofantsko enačbo $ax + by = d$. V posebnem primeru, da je b tuj a, je x inverz števila a v multiplikativni grupi Z_b^*.

Časovna zahtevnost: $O(\log(\max\{a, b\}))$

Prostorska zahtevnost: $O(1)$

Testiranje na terenu: UVa 756

```
int ext_gcd(int a, int b, int& retx, int& rety) {
    int x = 0, px = 1, y = 1, py = 0, r, q;
    while (b != 0) {
        r = a % b; q = a / b;  // quotient and reminder
        a = b; b = r;          // gcd swap
        r = px - q * x;        // x swap
        px = x; x = r;
        r = py - q * y;        // y swap
        py = y; y = r;
    }
    retx = px; rety = py;      // return
    return a;
}
```

1.3 Kitajski izrek o ostankih

Vhod: Sistem n kongruenc $x \equiv a_i \pmod{m_i}$, m_i so paroma tuji.

Izhod: Število x, ki reši ta sistem dobimo po formuli

$$x = \left[\sum_{i=1}^{n} a_i \frac{M}{m_i} \left[\left(\frac{M}{mi} \right)^{-1} \right]_{m_i} \right]_M, \qquad M = \prod_{i=1}^{n} m_i,$$

kjer $[x^{-1}]_m$ označuje inverz x po modulu m. Vrnjeni x je med 0 in M.

Časovna zahtevnost: $O(n \log(\max\{m_i, a_i\}))$

Prostorska zahtevnost: $O(n)$

Potrebuje: Evklidov algoritem (str. 3)

Testiranje na terenu: UVa 756

Opomba: Pogosto potrebujemo `unsigned long long` namesto `int`.

```
int mul_inverse(int a, int m) {
    int x, y;
    ext_gcd(a, m, x, y);
    return (x + m) % m;
}

int chinese_reminder_theorem(const vector<pair<int, int>>& cong) {
    int M = 1;
    for (size_t i = 0; i < cong.size(); ++i) {
        M *= cong[i].second;
    }
    int x = 0, a, m;
    for (const auto& p : cong) {
        tie(a, m) = p;
        x += a * M / m * mul_inverse(M/m, m);
        x %= M;
    }
    return (x + M) % M;
}
```