Codebook

Pitoni++

Žiga Gosar, Maks Kolman, Jure Slak

verzija: 14. 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;
}
```

1.4 Hitro potenciranje

Vhod: Število g iz splošne grupe in $n \in \mathbb{N}_0$.
Izhod: Število g^n.
Časovna zahtevnost: $O(\log(n))$
Prostorska zahtevnost: $O(1)$
Testiranje na terenu: `http://putka.upm.si/tasks/2010/2010_3kolo/nicle`

```
int fast_power(int g, int n) {
    int r = 1;
    while (n > 0) {
        if ((n & 1) == 1) {
            r *= g;
        }
        g *= g;
        n >>= 1;
    }
    return r;
}
```

1.5 Številski sestavi

Vhod: Število $n \in \mathbb{N}_0$ ali $\frac{p}{q} \in Q$ ter $b \in [2, \inf) \cap \mathbb{N}$.
Izhod: Število n ali $\frac{p}{q}$ predstavljeno v izbranem sestavu z izbranimi števkami in označeno periodo.
Časovna zahtevnost: $O(\log(n))$ ali $O(q \log(q))$.
Prostorska zahtevnost: $O(n)$ ali $O(q)$.
Testiranje na terenu: `http://putka.upm.si/tasks/2010/2010_finale/ulomki`
Opomba: Zgornja meja za bazo b je dolžina niza `STEVILSKI_SESTAVI_ZNAKI`.

```
char STEVILSKI_SESTAVI_ZNAKI[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

string convert_int(int n, int baza) {
    if (n == 0) return "0";
    string result;
    while (n > 0) {
        result.push_back(STEVILSKI_SESTAVI_ZNAKI[n % baza]);
        n /= baza;
    }
    reverse(result.begin(), result.end());
    return result;
}
```

```
string convert_fraction(int stevec, int imenovalec, int base) {
    div_t d = div(stevec, imenovalec);
    string result = convert_int(d.quot, base);
    if (d.rem == 0) return result;

    string decimalke;  // decimalni del
    result.push_back('.');
    int mesto = 0;
    map<int, int> spomin;
    spomin[d.rem] = mesto;
    while (d.rem != 0) {  // pisno deljenje
        mesto++;
        d.rem *= base;
        decimalke += STEVILSKI_SESTAVI_ZNAKI[d.rem / imenovalec];
        d.rem %= imenovalec;
        if (spomin.count(d.rem) > 0) {  // periodicno
            result.append(decimalke.begin(), decimalke.begin() + spomin[d.rem]);
            result.push_back('(');
            result.append(decimalke.begin() + spomin[d.rem], decimalke.end());
            result.push_back(')');
            return result;
        }
        spomin[d.rem] = mesto;
    }
    result += decimalke;
    return result;  // koncno decimalno stevilo
}
```