

Codebook

Pitoni++

Žiga Gosar, Maks Kolman, Jure Slak

verzija: 14. december 2014

Kazalo

1	Teorija števil	3
1.1	Evklidov algoritem	3
1.2	Razširjen Evklidov algoritem	3
1.3	Kitajski izrek o ostankih	3
1.4	Hitro potenciranje	4

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: $O(1)$

```
1  int gcd(int a, int b) {
2      int t;
3      while (b != 0) {
4          t = a % b;
5          a = b;
6          b = t;
7      }
8      return a;
9  }
```

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 \mathbb{Z}_b^* .

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

Prostorska zahtevnost: $O(1)$

Testiranje na terenu: UVa 756

```
1  int ext_gcd(int a, int b, int& retx, int& rety) {
2      int x = 0, px = 1, y = 1, py = 0, r, q;
3      while (b != 0) {
4          r = a % b; q = a / b; // quotient and reminder
5          a = b; b = r;        // gcd swap
6          r = px - q * x;      // x swap
7          px = x; x = r;
8          r = py - q * y;      // y swap
9          py = y; y = r;
10     }
11     retx = px; rety = py;    // return
12     return a;
13 }
```

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

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

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.

```
1  int mul_inverse(int a, int m) {
2      int x, y;
3      ext_gcd(a, m, x, y);
4      return (x + m) % m;
5  }
6
7  int chinese_remainder_theorem(const vector<pair<int, int>>& cong) {
8      int M = 1;
9      for (size_t i = 0; i < cong.size(); ++i) {
10         M *= cong[i].second;
11     }
12     int x = 0, a, m;
13     for (const auto& p : cong) {
14         tie(a, m) = p;
15         x += a * M / m * mul_inverse(M/m, m);
16         x %= M;
17     }
18     return (x + M) % M;
19 }
```

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

```
1  int fast_power(int g, int n) {
2      int r = 1;
3      while (n > 0) {
4          if ((n & 1) == 1) {
5              r *= g;
6          }
7          g *= g;
8          n >>= 1;
9      }
10     return r;
11 }
```