

# Codebook

Pitoni++

Žiga Gosar, Maks Kolman, Jure Slak

verzija: 23. december 2014

# Kazalo

<b>1</b>	<b>Grafi</b>	<b>3</b>
1.1	Topološko sortiranje . . . . .	3
1.2	Mostovi in atriculation points grafa . . . . .	3
<b>2</b>	<b>Teorija števil</b>	<b>4</b>
2.1	Evklidov algoritem . . . . .	4
2.2	Razširjen Evklidov algoritem . . . . .	5
2.3	Kitajski izrek o ostankih . . . . .	5
2.4	Hitro potenciranje . . . . .	5
2.5	Številski sestavi . . . . .	6
<b>3</b>	<b>Eulerjeva funkcija <math>\phi</math></b>	<b>7</b>

# 1 Grafi

## 1.1 Topološko sortiranje

**Vhod:** Število vozlišč  $n$  in število povezav  $m$  ter seznam povezav  $E$  oblike  $u \rightarrow v$  dolžine  $m$ . Usmerjen graf  $G$  je tako sestavljen iz vozlišč z oznakami 0 do  $n - 1$  in povezavami iz  $E$ .  $G$  ne sme imeti zank, če pa jih ima, se jih lahko brez škode odstrani.

**Izhod:** Topološka ureditev usmerjenega grafa  $G$ , to je seznam vozlišč v takem vrstnem redu, da nobena povezava ne kaže nazaj. Če je vrnjeni seznam krajši od  $n$ , potem ima  $G$  cikle.

**Časovna zahtevnost:**  $O(V + E)$

**Prostorska zahtevnost:**  $O(V)$

**Testiranje na terenu:** UVa 10305

```
1  vector<int> topological_sort(int n, int m, const int E[][2]) {
2      vector<vector<int>> G(n);
3      vector<int> ingoing(n, 0);
4
5      for (int i = 0; i < m; ++i) {
6          int a = E[i][0], b = E[i][1];
7          G[a].push_back(b);
8          ingoing[b]++;
9      }
10
11     queue<int> q; // morda priority_queue, če je vrstni red pomemben
12     for (int i = 0; i < n; ++i)
13         if (ingoing[i] == 0)
14             q.push(i);
15
16     vector<int> res;
17     while (!q.empty()) {
18         int t = q.front();
19         q.pop();
20
21         res.push_back(t);
22
23         for (int v : G[t])
24             if (--ingoing[v] == 0)
25                 q.push(v);
26     }
27
28     return res; // če res.size() != n, ima graf cikle.
29 }
```

## 1.2 Mostovi in atriculation points grafa

**Vhod:** Število vozlišč  $n$  in število povezav  $m$  ter seznam povezav  $E$  oblike  $u \rightarrow v$  dolžine  $m$ . Neusmerjen graf  $G$  je tako sestavljen iz vozlišč z oznakami 0 do  $n - 1$  in povezavami iz  $E$ .

**Izhod:** Seznam articulation pointov: točk, pri katerih, če jih odstranimo, graf razpade na dve komponenti in seznam mostov grafa  $G$ : povezav, pri katerih, če jih odstranimo, graf razpade na dve komponenti.

**Časovna zahtevnost:**  $O(V + E)$

**Prostorska zahtevnost:**  $O(V + E)$

**Testiranje na terenu:** UVa 315

```
1  namespace {
2      vector<int> low;
3      vector<int> dfs_num;
4      vector<int> parent;
5  }
```

```

6
7 void articulation_points_and_bridges_internal(int u, const vector<vector<int>>& G,
8 vector<bool>& articulation_points_map, vector<pair<int, int>>& bridges) {
9     static int dfs_num_counter = 0;
10    low[u] = dfs_num[u] = ++dfs_num_counter;
11    int children = 0;
12    for (int v : G[u]) {
13        if (dfs_num[v] == -1) { // unvisited
14            parent[v] = u;
15            children++;
16
17            articulation_points_and_bridges_internal(v, G, articulation_points_map, bridges);
18            low[u] = min(low[u], low[v]); // update low[u]
19
20            if (parent[u] == -1 && children > 1) // special root case
21                articulation_points_map[u] = true;
22            else if (parent[u] != -1 && low[v] >= dfs_num[u]) // articulation point
23                articulation_points_map[u] = true; // assigned more than once
24            if (low[v] > dfs_num[u]) // bridge
25                bridges.push_back({u, v});
26        } else if (v != parent[u]) {
27            low[u] = min(low[u], dfs_num[v]); // update low[u]
28        }
29    }
30 }
31
32 void articulation_points_and_bridges(int n, int m, const int E[][2],
33 vector<int>& articulation_points, vector<pair<int, int>>& bridges) {
34     vector<vector<int>> G(n);
35     for (int i = 0; i < m; ++i) {
36         int a = E[i][0], b = E[i][1];
37         G[a].push_back(b);
38         G[b].push_back(a);
39     }
40
41     low.assign(n, -1);
42     dfs_num.assign(n, -1);
43     parent.assign(n, -1);
44
45     vector<bool> articulation_points_map(n, false);
46     for (int i = 0; i < n; ++i) {
47         if (dfs_num[i] == -1) {
48             articulation_points_and_bridges_internal(i, G, articulation_points_map, bridges);
49         }
50     }
51     for (int i = 0; i < n; ++i) {
52         if (articulation_points_map[i]) {
53             articulation_points.push_back(i); // actually return only articulation points
54         }
55     }
56 }

```

## 2 Teorija števil

### 2.1 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:**  $O(\log(a) + \log(b))$

**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 }

```

## 2.2 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:**  $O(\log(a) + \log(b))$

**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 remainder
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 }
```

## 2.3 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. 4)

**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 }
```

## 2.4 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](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) {
5              r *= g;
6          }
7          g *= g;
8          n >>= 1;
9      }
10     return r;
11 }
```

## 2.5 Številski sestavi

**Vhod:** Število  $n \in \mathbb{N}_0$  ali  $\frac{p}{q} \in \mathbb{Q}$  ter  $b \in [2, \infty) \cap \mathbb{N}$ .

**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](http://putka.upm.si/tasks/2010/2010_finale/ulomki)

**Opomba:** Zgornja meja za bazo  $b$  je dolžina niza STEVILSKI\_SESTAVI\_ZNAKI.

```
1  char STEVILSKI_SESTAVI_ZNAKI[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
2
3  string convert_int(int n, int baza) {
4      if (n == 0) return "0";
5      string result;
6      while (n > 0) {
7          result.push_back(STEVILSKI_SESTAVI_ZNAKI[n % baza]);
8          n /= baza;
9      }
10     reverse(result.begin(), result.end());
11     return result;
12 }
13
14 string convert_fraction(int stevec, int imenovalec, int base) {
15     div_t d = div(stevec, imenovalec);
16     string result = convert_int(d.quot, base);
17     if (d.rem == 0) return result;
18
19     string decimalke; // decimalni del
20     result.push_back('.');
21     int mesto = 0;
22     map<int, int> spomin;
23     spomin[d.rem] = mesto;
24     while (d.rem != 0) { // pisno deljenje
25         mesto++;
26         d.rem *= base;
27         decimalke += STEVILSKI_SESTAVI_ZNAKI[d.rem / imenovalec];
28         d.rem %= imenovalec;
29         if (spomin.count(d.rem) > 0) { // periodično
30             result.append(decimalke.begin(), decimalke.begin() + spomin[d.rem]);
31             result.push_back('(');
32             result.append(decimalke.begin() + spomin[d.rem], decimalke.end());
33             result.push_back(')');
34             return result;
35         }
36         spomin[d.rem] = mesto;
37     }
38     result += decimalke;
39     return result; // končno decimalno stevilo
40 }
```

### 3 Eulerjeva funkcija $\phi$

**Vhod:** Število  $n \in \mathbb{N}$ .

**Izhod:** Število  $\phi(n)$ , to je število števil manjših ali enakih  $n$  in tujih  $n$ . Direktna formula:

$$\phi(n) = n \cdot \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

**Časovna zahtevnost:**  $O(\sqrt{n})$ .

**Prostorska zahtevnost:**  $O(1)$ .

**Testiranje na terenu:** <https://projecteuler.net/problem=69>

```
1  int euler_phi(int n) {
2      int res = n;
3      for (int i = 2; i*i <= n; ++i) {
4          if (n % i == 0) {
5              while (n % i == 0) {
6                  n /= i;
7              }
8              res -= res / i;
9          }
10     }
11     if (n > 1) res -= res / n;
12     return res;
13 }
```