Teorija brojeva

Adrian Satja Kurdija, PMF-MO
askurdija@gmail.com
za kolegij Natjecateljsko programiranje
15. studenoga 2014.

Koji su nam ciljevi?

Low level: upoznavanje s nekim pojmovima i algoritmima koji se bave prirodnim brojevima.

Middle level: razvoj matematičko-algoritamskog načina razmišljanja, programiranja te problem-solvinga općenito.

High level: naći dobar posao.

Osnovni pojmovi

Djeljivost

Prost broj

Rastav na proste faktore

Najveći zajednički djelitelj

Dijeljenje s ostatkom

Matrice

Rekurzivne relacije

Uvodni zadatak

Zadan je prirodan broj **N** (1 ≤ **N** ≤ 1 000 000).

Koja je posljednja strogo pozitivna znamenka broja **N**! = 1 * 2 * 3 * … * (**N** - 1) * **N**?

Rješenje: na ploči.

Kako provjeriti je li broj prost?

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

   return true;
}
```

Možemo i mnogo brže!

```
bool je_li_prost(int n)
{
   for (int i = 2; i <= sqrt(n); ++i)
      if (n % i == 0)
         return false;

   return true;
}
```

Generiranje prostih brojeva

Ideja:

2 je prost, prekriži sve daljnje brojeve djeljive s 2

3 je prost, prekriži sve daljnje brojeve djeljive s 3

4 je prekrižen

5 je prost, prekriži sve daljnje brojeve djeljive s 5

6 je prekrižen

7 je prost, prekriži sve daljnje brojeve djeljive s 7

8 je prekrižen

...

Eratostenovo sito

```
// Na pocetku nitko nije prekrizen.
bool prekrizen[M];
for (int n = 2; n < M; ++n)
   prekrizen[n] = false;

for (int n = 2; n < M; ++n)
   if (!prekrizen[n]) {  // Znaci da je prost!
      for (int i = 2 * n; i < M; i += n)
         prekrizen[i] = true;
   }
```

Rastav broja na proste faktore

Ideja:

za svaki broj od 2 nadalje provjeravamo je li on prost faktor od **N**. Ako jest, **N** njime dijelimo dok možemo.

Rastav na proste faktore

```
for (int p = 2; p <= n; ++p)
   if (je_li_prost(p)) {
      while (n % p == 0) {
         cout << p << endl;
         n /= p;
      }
   }
```

Ovo možemo ubrzati čak na trima mjestima! Kojim?

Rastav na proste faktore

```
int korijen = (int)sqrt(n);
for (int p = 2; n > 1 && p <= korijen; ++p)
    // Ne treba provjeravati je li p prost.
    while (n % p == 0) {
        cout << p << endl;
        n /= p;
    }
if (n != 1)  // Preostali broj je prost.
    cout << n << endl;
```

Zadatak Potpuni (DMIH 2007.)

http://hsin.hr/dmih07/zadaci/pascal_c_cpp/dan1/druga/zadaci.pdf

Na ploči je na početku napisan broj 1.

Potom dolazi **K** ljudi (1 ≤ **K** ≤ 500 000), jedan po jedan.

Svaki od njih briše trenutni broj i mjesto njega zapisuje njegov umnožak nekim brojem iz intervala [1, 1 000 000].

Nakon svakog množenja valja ispisati je li novi broj potpun kvadrat.

Kako brzo faktorizirati mnogo brojeva?

Za svaki broj unaprijed zapišemo neki (bilo koji) njegov prosti faktor.

(Kako? S pomoću Eratostenova sita!)

Sada dani broj dijelimo (uvijek znamo neki faktor) dok ne dođemo do broja 1. Tako smo pronašli sve njegove faktore.

```
while (n > 1) {
    int p = prost_faktor[n];
    ++zastupljenost[p];
    n /= p;
}
```

Najveći zajednički djelitelj

Oznaka GCD: greatest common divisor

Kako računati GCD(**a**, **b**)?

Prva ideja: za svaki djelitelj od **a** provjeri je li on ujedno i djelitelj od **b**.

Sve djelitelje broja **a** možemo naći u složenosti O(sqrt(**a**)). Zašto?

Možemo dakle postići O(sqrt(min{**a, b**})).

Najveći zajednički djelitelj

Mnogo brže: Euklidov algoritam.

Temelji se na formuli

GCD(**a**, **b**) = GCD(**b**, **a** % **b**), za **b** != 0.

Dokaz na ploči.

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

Primjeri porabe GCD-a

- kraćenje razlomaka
- računanje najmanjeg zajedničkog višekratnika (least common multiple):

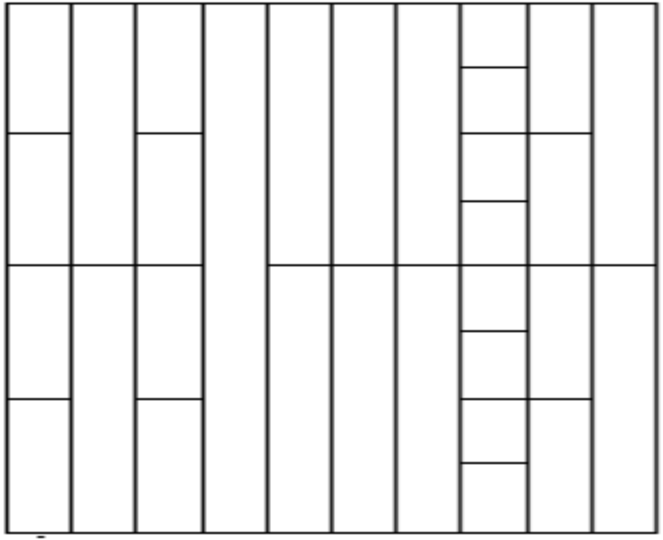
  LCM(**a**, **b**) = **a** * **b** / GCD(**a**, **b**)

  Dokaz na ploči.
- podjela pravokutnika na kvadrate
- koje se cjelobrojne točke vide iz ishodišta, tj. ne zaklanjaju ih druge cjelobrojne točke?
- sljedeća dva zadatka

Zadatak Guma

http://hsin.hr/honi/arhiva/2013_2014/kolo4_zadaci.pdf

Guma ima **N** vertikalnih dijelova od kojih svaki mora biti prerezan određenim brojem jednako udaljenih zareza. Koliko rezova je potrebno?



Zadatak Guma: rješenje (1/3)

Ako “traku” dijelimo na **K** dijelova, na njoj je **K**-1 rezova.

Neki od tih rezova produljeni su s prethodne trake, a neki tek započeti.

Rješenje će biti zbroj tek započetih rezova za sve trake.

Zadatak Guma: rješenje (2/3)

Pretpostavimo da je prethodna traka podijeljena na 30 dijelova, a trenutna na 70 dijelova.

Oba broja djeljiva su s 10. Što nam to znači?

30 dijelova trake možemo promatrati kao 10 velikih dijelova po 3.

70 dijelova trake možemo promatrati kao 10 velikih dijelova po 7.

Veliki se dijelovi podudaraju i rezove između njih možemo produljiti!

Tih rezova ima 10 - 1 = 9.

Preostalih 69 - 9 rezova valja nam započeti i pribrojiti rješenju.

Zadatak Guma: rješenje (3/3)

Iz prethodnoga primjera izvodimo općenit zaključak:

broj rezova koji možemo produljiti s trake od **a** dijelova na traku od **b** dijelova jednak je

GCD(**a**, **b**) - 1.

Ovo je točno i za relativno proste **a**, **b**!

Zadatak TB (Međuispit 2013.)

Za prirodan broj **n** definirajmo **f**(**n**) kao najmanji prirodan broj koji nije djelitelj broja **n**.

Na primjer: **f**(5) = 2, **f**(6) = 4.

Koliko ima prirodnih brojeva **n** manjih od zadanog broja **B** takvih da je **f**(**n**) = **K**?

Ograničenja:

$3 \leq \mathbf{B}, \mathbf{K} \leq 10^{17}$

Zadatak TB: rješenje (1/3)

Kakav mora biti broj **N** čiji je **f** jednak **K**?

- Mora biti djeljiv svim brojevima manjima od **K**.
- Ne smije biti djeljiv s **K**.

Zadatak TB: rješenje (2/3)

Prvi se uvjet svodi na:

N je djeljiv s NZV(1, 2, ..., **K** - 1).

Takvih brojeva ima

B / NZV(1, 2, ..., **K** - 1)

(cjelobrojno dijeljenje).

Zadatak TB: rješenje (3/3)

Od toga valja oduzeti brojeve djeljive s **K**.
Koliko ima takvih brojeva?

Ti su brojevi:

- djeljivi s NZV(1, 2, ..., **K** - 1),
- djeljivi s **K**.

Dakle, djeljivi su s NZV(1, 2, ..., **K** - 1, **K**),

pa ih ima **B** / NZV(1, 2, ..., **K** - 1, **K**).

Zadatak Bomboni (DZ)

Na zabavi je **m** djece i **N** vrećica bombona.

Ocjena zabave je

m * broj_otvorenih_vrećica,

a otvorene su one vrećice čiji je broj bombona djeljiv s **m**.

Valja pronaći **m** koji maksimizira ocjenu zabave.

Ograničenja:

N ≤ 200 000, broj bombona u svakoj vrećici ≤ 2 000 000.

Zadatak Bomboni: rješenje (1/2)

Za svaki **m** od 1 do **M** (maksimalni broj bombona u nekoj vrećici) računamo ocjenu.

Brute-force: prolazimo po svim vrećicama i brojimo one djeljive s **m**.

Složenost: O(**MN**).

Presporo!

Zadatak Bomboni: rješenje (2/2)

Prolazimo po **višekratnicima** od **m**,
brojeći vrećice na koje pritom naiđemo.
Treba nam pomoćni niz:

a[**k**] = broj vrećica s **k** bombona.

Broj koraka:

M / 1 + **M** / 2 + **M** / 3 … + **M** / (**M** - 1) + **M** / M.

Ovo je približno O(**M** log **M**)!

Za samostalno rješavanje:

1. Zadatak Kušač: pronaći najmanji broj rezova potreban da se **N** jednakih kobasica podijeli na **M** jednakih dijelova. *http://hsin.hr/honi/arhiva/2013_2014/kolo1_zadaci.pdf*
2. Zadatak Snaga, teža verzija zadatka TB: *http://hsin.hr/honi/arhiva/2012_2013/kolo1_zadaci.pdf*
3. Zadatak Guma (teža verzija, rezovi smiju “preskakati” dijelove): *http://hsin.hr/coci/archive/2013_2014/contest4_tasks.pdf*

Potenciranje

$\mathbf{a}^{\mathbf{n}}$ možemo računati uzastopnim množenjem **n** puta.
Složenost: O(**n**).

Ali možemo i mnogo brže!

Osnovna ideja:

$$\mathbf{a}^{2\mathbf{k}} = (\mathbf{a}^{\mathbf{k}})^2.$$

Broj množenja ovim je već prepolovljen.

A što ako eksponent nije paran?

$$\mathbf{a}^{2\mathbf{k}+1} = (\mathbf{a}^{\mathbf{k}})^2 * \mathbf{a}.$$

Brzo potenciranje: O(log n)

```
int potencija(int a, int n) {
   if (n == 0) return 1;
   if (n % 2 == 0)
      return kvadrat(potencija(a, n/2));
   else
      return kvadrat(potencija(a, n/2)) * a;
}
```

Može i iterativno, koristeći binarni zapis broja **n** (slajd u prezentaciji Luke Kalinovčića).

Modularna aritmetika

Naravno, u kodu s prethodne stranice vjerojatno se događa overflow.

Zato se u raznim zadacima gdje je rezultat veoma velik traži samo njegov ostatak pri dijeljenju nekim zadanim prostim brojem, npr. 1 000 007.

Modularna aritmetika

Pri zbrajanju, množenju, oduzimanju vrijedi:

$$((a \bmod M) + (b \bmod M)) \bmod M \equiv (a + b) \bmod M,$$
$$((a \bmod M) * (b \bmod M)) \bmod M \equiv (a * b) \bmod M,$$
$$((a \bmod M) - (b \bmod M)) \bmod M \equiv (a - b) \bmod M.$$

Dakle, smijemo modati međurezultate.

Modularno oduzimanje

Kod oduzimanja valja paziti.

Primjer: računamo 9 - 6 modulo 7.

Željeni rezultat je, naravno, 3.

Ali, u jeziku poput C++a može nam ispasti:

((9 % 7) - (6 % 7)) % 7 = (2 - 6) % 7 = -4 % 7 = -4.

Rješenje: rezultatu oduzimanja pribrojimo modul (da osiguramo pozitivnost) i tada ga modamo.

Overflow alert!

Čak i u liniji poput ove:

```
return (a % M) * (b % M) % M;
```

može se dogoditi overflow!

Naime, umnožak može ispasti prevelik, bez obzira što ga poslije modamo.

Rješenje: neki od faktora castati u long long.

A što je s dijeljenjem?

Ako je p prost te a, b relativno prosti sa p,

onda **(a / b) modulo p** valja računati kao

$$(a * b^{p-2}) \% p.$$

To slijedi iz malog Fermatova teorema: $b^{p-1} \equiv 1 \pmod{p}$.

Dakle, mjesto da dijelimo sa b, množimo s b^{p-2} (brzo potenciranje!).

To je ***modularni inverz*** broja b.

Što ako modul nije prost? Primjer: sljedeći zadatak.

Zadatak Vlakić

Imamo **R** crvenih, **G** zelenih i **B** plavih vagona.

Koliko različitih kompozicija možemo složiti od svih vagona?

Rješenje ispisati modulo **M**.

1 ≤ **R**, **G**, **B** ≤ 20 000

1 ≤ **M** ≤ 1 000 000 000

Prošireni Euklidov algoritam

Prošireni Euklidov algoritam nalazi koeficijente **x**, **y** takve da vrijedi **ax** + **by** = gcd(**a**, **b**).

Najvažnije primjene:

- nalaženje **x**, **y** takvih da je **ax** + **by = c** (linearna diofantska jednadžba), npr. 7**x** + 17**y** = 4
- rješavanje modularnih jednadžbi: 7**x** = 4 (mod 17)
- primjer: zadatak Skakavci (na ploči)

Ako vam zatreba, guglajte “extended euclid” ili pogledajte prezentaciju Luke Kalinovčića.

Bignum aritmetika

Rad s velikim brojevima (većima od long long).

Broj pamtimo kao niz znamenaka:

$4297 = 4 * 10^3 + 2 * 10^2 + 9 * 10^1 + 7 * 10^0$,

zapisujemo ga u vektor: <4 2 9 7], na početku (nultoj poziciji) nalazi se znamenka jedinica (7)!

Općenito, znamenka na poziciji p množi 10^p.

Sličnost s polinomima!

Zbrajanje bignumova

Slično “pismenom” zbrajanju koje smo učili u školi.

```
// Dodaj nule na kraj vektora a ili b
// tako da imaju jednak broj_znamenaka.
vector<int> zbroj(broj_znamenaka);
for (int i = 0; i < broj_znamenaka; ++i)
  zbroj[i] = a[i] + b[i];
rijesi_prijenose(zbroj);
```

Rješavanje prijenosa

Zbroj iz prethodne petlje može ispasti npr.

<12 9 17 15], tj.

$$12 * 10^3 + 9 * 10^2 + 17 * 10^1 + 15 * 10^0 = 13085.$$

Valja ga pretvoriti u <1 3 0 8 5].

Rješavanje prijenosa

```
void rijesi_prijenose(vector<int> &a)
{
   a.push_back(0); // znamenka vise, ako zatreba
   for (int i = 0; i + 1 < a.size(); ++i) {
      a[i + 1] += a[i] / 10;  // to je carry
      a[i] %= 10;
   }
   if (a.back() == 0) a.pop_back();
}
```

Množenje bignumova

Slično množenju polinoma.

Svaka znamenka prvoga broja množi se sa svakom znamenkom drugoga broja.

Primjer:

$$(7 * 10^2) * (9 * 10^5) = 63 * 10^{2 + 5}$$

Množenje bignumova

```
vector<int> umnozak(a.size() + b.size());
for (int i = 0; i < a.size(); ++i)
   for (int j = 0; j < b.size(); ++j)
      umnozak[i + j] += a[i] * b[j];

rijesi_prijenose(umnozak);
```

Dijeljenje bignumova

Nije lako!

Ideja: binarno pretraživanje.

Svodi se na zbrajanje, dijeljenje sa 2, množenje i uspoređivanje.

Matrice i linearne rekurzivne relacije

Prebaci se na prezentaciju Luke Kalinovčića.

Zadatak Rekurzija

Zadan je niz cijelih brojeva x koji se dobiva iz brojeva A, B, C, D i E na sljedeći način:

$$x[1] = A \bmod 31337,$$

$$x[2] = B \bmod 31337,$$

$$x[n] = (C * x[n-2] + D * x[n-1] + E) \bmod 31337,$$

$$\text{za } n > 2.$$

Napišite program koji nalazi K-ti član tog niza.

Rješenje: na ploči.

I gotovi smo!