Hello, World!

Алгоритми теорије бројева - Део 1

Слободан Митровић, Марко Савић

Теорија бројева је веома лепа област математике, али многима није познато да је она изузетно коришћена и у пракси. Алгоритми теорије бројева играју веома значајну улогу у криптографији, прецизније криптографским шемама базираним на великим простим бројевима.

У овом и наредном броју писаћемо о неколико алгоритма теорије бројева, а након тога ћемо представити и неке основне технике криптографије.

Провера да ли је број прост

У раду са бројевима врло често се јавља потреба да се дати број $n \in N$ провери да ли је прост или не. Ради подсећања, n је прост број ако и само ако једини природни бројеви са којим је дељив су 1 и n. Додајући на то да позитивно n није дељиво ни једним бројем већим од n, следећи једноставан алгоритам тестира да ли је n прост број или није.

Алгоритам Prost
Улаз: Природан позитиван број n.
Излаз: Одговор на питање да ли је n прост број.

```
for i ← 2 ... n − 1
        if i дели n
                return false
return true
```

Временска сложеност алгоритма **Prost** је $O(n)$.

Можемо добити квадратно убрзање датог алгоритма на следећи начин. Претпоставимо да n није прост број. То значи, да $n = a \cdot b$, где је $a, b \geq 2$. То даље значи да је $a \leq \sqrt{n}$ или $b \leq \sqrt{n}$. Односно, ако n није прост број, тада постоји делилац који није већи од $\sqrt{n}$. Овај закључак можемо да применимо путем следећег алгоритма и тиме побољшамо временску сложеност алгоритма за утврђивање да ли је број прост на $O(\sqrt{n})$.

Алгоритам ProstBrz
Улаз: Природан број n.
Излаз: Одговор на питање да ли је n прост број.

```
for i ← 2 ... √n
        if i дели n
                return false
return true
```

Ератостеново сито

Понекад се у задацима од нас тражи да пронађемо све просте бројеве који нису већи од n. Један начин је да једноставно за сваки позитиван број $i \leq n$ позовемо **ProstBrz** са параметром i. За такав алгоритам можемо да гарантујемо сложеност $O(n\sqrt{n})$. У наставку ћемо видети да постоји алгоритам, који се зове Ератостеново сито, за који можемо да покажемо да је знатно бржи.

Неформално, Ератостеново сито је следећа процедура. На папир запишемо редом бројеве од 2 до n. На почетку, сви бројеви су непрецртани. Изаберемо најмањи непрецртан број. Нека је то број p. Затим прецртамо бројеве облика $k \cdot p$, за $k \geq 1$, и наставимо процес докле год има непрецртаних бројева.

Лако је видети да сваки прецртани број за неко p и $k \geq 2$ није прост. С друге стране, можемо математичком индукцијом показати да сваки изабрани број p јесте прост. Као базу индукције, довољно је да приметимо да ће број 2 бити изабран као прво p. У индукцијском кораку можемо да покажемо да је $p > 2$ прост број контрадикцијом. Наиме, претпоставимо да је неко изабрано p дељиво са простим бројем p_1. Тада по индукцијској хипотези знамо да је у неком моменту p_1 било изабрано као најмањи непрецртан број. Из тога следи да ће $k \cdot p_1 = p$, за $k = p/p_1$, бити прецртано. То је у контрадикцији с претпоставком да је p дељиво са p_1, те p јесте прост број.

Дакле, број p, $2 \leq p \leq n$, је прост ако и само ако је изабран у Ератостеновом ситу као најмањи непрецртан број. Следи псеудокод алгоритма.

Алгоритам EratostenovoSito
Улаз: Природан број n.
Излаз: Сви прости бројеви из интервала $2 \ldots n$.
Обележи све бројеве као непрецртане.
for $i \leftarrow 2 \ldots n$
if i није прецртано
означи да је i прост број.
for $k \leftarrow 2 \ldots \lfloor n/i \rfloor$
прецртај $k \cdot i$
return сви бројеви означени као прости.

Остаје нам још да анализирамо временску сложеност алгоритма. Да бисмо одредили време извршавања потребно је избројати укупан број итерација у петљи по k. Број итерација по k, означимо тај број са T, можемо записати на следећи начин.

$$T = \sum_{p \text{ је прост и } p \leq n} \frac{n}{p} \leq \sum_{p=2}^{n} \frac{n}{p} = n \sum_{p=2}^{n} \frac{1}{p} = O(n \log n)$$

Овим смо показали да је сложеност алгоритма $O(n \log n)$. Детаљнијом анализом се може добити мало прецизнији резултат, $T = O(n \log \log n)$.

Додатно, ако се алгоритам **EratostenovoSito** промени, тако да петља по k почиње од i уместо од 2, алгоритам постаје још ефикаснији. Ова модификација не утиче на исправност алгоритма, јер када наиђемо на i које није прецртано, знамо да су сви бројеви облика $k \cdot i$ за $k \in \{2, 3, \ldots, i-1\}$ већ прецртани када је претходно откривен неки прост фактор броја k, што се сигурно десило, јер је $k < i$.

Еуклидов алгоритам

За проналажење *највећег заједничког делиоца* (скраћено *нзд*) два позитивна броја a и b, у ознаци $\text{nzd}(a, b)$, можемо користити *Еуклидов алгоритам*. Идеја Еуклидовог алгоритма је следећа. Ако је $a = 0$ или $b = 0$, тада је, очигледно, $\text{nzd}(a, b) = \max\{a, b\}$. Идеја алгоритма је да ако су оба броја позитивна, можемо смањити један од бројева тако да остане ненегативан, а да се *нзд* не промени. Одговарајући алгоритам је приказан испод.

Алгоритам Euklid
Улаз: Природни бројеви a и b.
Излаз: $\text{nzd}(a, b)$.

```
while a ≠ 0 и b ≠ 0
    if a < b
        b ← b − a
    else
        a ← a − b
return max{a, b}
```

Означимо са a_i и b_i вредности променљивих a и b након итерације i. Специјално, a_0 и b_0 су вредности променљивих датих на улазу. Лако је видети да су $a_i \geq 0$ и $b_i \geq 0$ за свако i, као и да се алгоритам завршава јер је $a_i + b_i < a_{i-1} + b_{i-1}$, односно у свакој итерацији збир променљивих a и b се смањује. Приметимо још да постоје бројеви a и b тако да је сложеност алгоритма $O(\max\{a, b\})$. На пример, за било коју вредност a и $b = 2$.

Остаје да се покаже да алгоритам враћа $\text{nzd}(a_0, b_0)$. Дефинишимо $g := \text{nzd}(a_0, b_0)$. Најпре ћемо показати да g дели и a_i и b_i за свако i. То се лако види из чињенице да ако g дели a_i и g дели b_i, тада g дели $a_i - b_i$ и g дели $b_i - a_i$. Дакле, инваријанта да g дели и a_i и b_i се одржава из итерације у итерацију, а за a_0 и b_0 тривијално важи.

Приметимо да на крају алгоритма не могу и a_t и b_t истовремено бити једнаки нули, где је t последња итерација алгоритма. Пошто g дели и a_t и b_t, важи да је излаз алгоритма неки број облика $k \cdot g$, $k \in N$. Остаје нам још да покажемо да $k = 1$. Претпоставимо да је $k > 1$. Изаберимо најмање j, тако да $k \cdot g$ дели a_j и b_j. По нашој претпоставци такво j постоји (знамо да је j највише t), и такође важи $j > 0$. Са друге стране, ако $k \cdot g$ дели a_j и b_j тада $k \cdot g$ дели a_{j-1} и b_{j-1} јер $k \cdot g$ дели $a_j + b_j$. Ово је у контрадикцији са нашим избором најмање вредности j за коју је $k > 1$, а самим тим контрадикција са претпоставком да је $k > 1$, што повлачи да је $k = 1$.

Еуклидов алгоритам може да се имплементира, тако да је знатно бржи од описаног алгоритма изнад. Узмимо пример где су на улазу дати $a = 100$ и $b = 3$. Тада ће описан алгоритам одузимати b од a све док не добије број 1. За то су потребне 33 итерације. Са друге стране, исти резултат се добија ако се понављано одузимање замени са $a \bmod b$. Ово запажање води следећем алгоритму.

Алгоритам EuklidBrz
Улаз: Природни бројеви a и b.
Излаз: $\text{nzd}(a, b)$.

```
while a ≠ 0 и b ≠ 0
        if a < b
                b ← b mod a
        else
                a ← a mod b
return max{a, b}
```

Приметимо да ако је $a < b$, број b након итерације постаје бар двоструко мањи. У случају да је $a \leq b/2$ важи $b \bmod a < a \leq b/2$, док ако је $a > b/2$, важи $b \bmod a = b - a < b - b/2 = b/2$. Аналогно се показује да број a постаје бар двоструко мањи након итерације у случају када је $a \geq b$. С обзиром на то да у свакој итерацији један од бројева постаје бар двоструко мањи, укупан број итерација је $O(\log a + \log b)$. Самим тим, толика је и сложеност алгоритма,.

За крај, наведимо мало једноставнију имплементацију последњег алгоритма, у којој је петља замењена рекурзијом.

Алгоритам EuklidBrzRec
Улаз: Природни бројеви a и b.
Излаз: $\text{nzd}(a, b)$.

```
if b = 0
        return a
else
        return EuklidBrzRec(b, a mod b)
```

Задаци

Дефинишимо функцију нзд за више од два параметра рекурентном формулом $\text{nzd}(a_1, a_2, \ldots, a_n) := \text{nzd}(\text{nzd}(a_1, a_2, \ldots, a_{n-1}), a_n)$. Доказати да вредност ове функције не зависи од редоследа навођења параметара. Показати да је време потребно да се израчуна вредност те функције $O(n + \log(\max\{a_1, a_2, \ldots, a_n\}))$.

Нека је p пермутација бројева од 1 до n. Низ пермутација a се добија, тако што је прва пермутација у том низу идентичка пермутација, а свака наредна се добија применом пермутације p на претходну. Прецизније,

$a_0 = I$, где је $I(i) = i$, а за свако $k \in N$ важи $a_k(i) = p(a_{k-1}(i))$. Конструисати алгоритам који налази најмање $k \in N$ за које је $a_k = I$.

У равни је дато n тачака са целобројним координатама. Конструисати алгоритам сложености боље од $O(n^3)$ који утврђује колико се највише прави може повући кроз дате тачке тако да никоје две нису паралелне.

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 2016/17 година