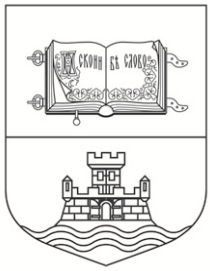
УНИВЕРЗИТЕТ У БЕОГРАДУ

МАТЕМАТИЧКИ ФАКУЛТЕТ



АЛГОРИТМИ ЕЛЕМЕНТАРНЕ ТЕОРИЈЕ БРОЈЕВА

-мастер рад-

СТУДЕНТ:
Дамјана Ђелмо 1068/2018

МЕНТОР:
проф. др Небојша Икодиновић

Београд, септембар 2020.

Садржај

Увод

Циљ овог рада јесте да пружи темељан увид у дизајн и анализу алгоритама из елементарне теорије бројева. Смисао алгоритамске теорије бројева јесте да ефикасно произведе објекте чије је постојање доказано у класичним теоремама у теорији бројева. На пример, Еуклидов[1] доказ нам говори да постоји прост број већи од било ког датог броја n; алгоритамска теорија бројева се пита: колико брзо можемо наћи такав прост број? Основна теорема аритметике каже да се сваки природан број n може записати као производ степена простих бројева; алгоритамска теорија бројева се пита: можемо ли брзо наћи просту факторизацију броја n? Да бисмо прецизно одговорили на питања неопходно је да се упознамо и са основама теоријског рачунарства. Развој криптологије и рачунарства заснован је на елементарној теорији бројева. Још пре 5000 година, древне цивилизације су развиле начине изражавања и извођења аритметике са целим бројевима. Током историје, различити начини су коришћени за означавање целих бројева. На пример, древни Вавилонци су користили 60 као основу за свој бројевни систем, а Маје су користиле 20. Наш начин изражавања целих бројева, декадни систем, развијен је у Индији пре око шест векова. Са појавом модерних рачунара, бинарни систем је постао широко распрострањен. Теорија бројева је коришћена на много начина да би се осмислили алгоритми за што ефикаснију рачунарску аритметику и за рачунске операције са великим целим бројевима. Стари Грци су у школи Питагоре[2], пре 2500 година, уочили разлику између простих и сложених бројева. Прост број је позитиван цео број који нема позитивних фактора сем јединице и самог себе. У записима, Еуклид је дао доказ да постоји бесконачно много простих бројева. Математичари су дуго тражили формуле које генеришу просте бројеве. Проблем разликовања простих бројева од сложених је опсежно проучаван. Древни грчки научник Ератостен[3] осмислио је методу, познатију као Ератостеново сито, која проналази све просте бројеве мање од одређене границе. Проблем ефикасног утврђивања да ли је цео број прост дуго је представљао изазов за математичаре. Данас је могуће за неколико минута, користећи рачунар, пронаћи просте бројеве са чак 200 децималних цифара. Ферма[4], Ојлер[5] и многи други математичари пронашли су маштовите технике факторизације природних бројева. Међутим, користећи најефикаснију

[1] Еуклид, старогрчки математичар који је живео око 300. године пре нове ере, аутор је првих сачуваних теоријских радова из математике. Оснивач је александријске математичке школе.
[2] Питагора је рођен 560. године пре нове ере. Умро око 500. године пре нове ере. Био је старогрчки филозоф и први истински математичар чије је име данас једно од најпознатијих у историји математике.
[3] Ератостен из Кирене (око 276-194. године пре нове ере) поставио је основе математичке географије; први нашао метод за мерење дужине лука меридијана.
[4] Пјер Ферма (1601-1665), француски математичар. По образовању је био правник и зарађивао је за живот бавећи се адвокатуром. Математиком се бавио из љубави, као „аматер“. Један је од оснивача теорије вероватноће.
[5] Леонард Ојлер (1707-1783), математичар, физичар, механичар и астроном. Сматра се за најпродуктивнијег математичара свих времена. Остварио је дубоке резултате у свим областима математике.

технику која је осмишљена, милијарде година би биле потребне рачунару да би се факторисао број са 200 децималних цифара.

Немачки математичар Карл Фридрих Гаус[6], који се сматра једним од највећих математичара свих времена, развио је језик конгруенција крајем осамнаестог века. Теорија конгруенција је коришћена у осмишљавању системских метода за чување рачунарских датотека, генерисању случајних бројева, производњи високо поузданих шифрирања и чак пројектовању брзих рачунара. Посебно је вредно истаћи да су рачунари у основи коначне машине; они имају ограничено складиштење, могу да се баве само бројевима неке коначне дужине и могу да обављају у суштини коначне кораке рачунања. Због таквих ограничења, конгруенцијска аритметика је посебно корисна у рачунарском хардверу и дизајну софтвера.

Алгоритми, који представљају прецизне поступке за решавање проблема на рачунару, у овом раду су илустровани путем кратких програма реализованих у програмском језику Пајтон (енгл. *Python*). Правила језика су једноставна. Коришћени су следећи елементи: операције `+`, `-`, `*`, `/`, `//`, `%` (редом, сабирање, одузимање, множење, дељење, целобројно дељење, остатак при дељењу); операције поређења `<`, `>`, `<=`, `>=`, `==`, `!=` (редом, мање од, веће од, мање или једнако од, веће или једнако од, није једнако); Пајтон бибилиотека `math` која нам даје математичке функције (нпр. `math.floor()` функција за налажење првог целог броја мањег од датог реалног броја); `decimal` за рад са реалним бројевима; `range(n, m)` функција која враћа низ бројева од n до m; `int()` цели бројеви; `input()` и `print()` функције које омогућавају унос и испис података; `for`, `while`, `if` петље.

[6] Карл Фридрих Гаус (1777-1855), немачки математичар кога су савременици називали „краљем математичара“ (*princeps matematicorum*). Гаус је, сећајући се свог детињства, говорио у шали: „Научио сам да рачунам пре него да говорим.“

1. О алгоритмима

Алгоритам[7] је један од основних појмова математике и рачунарства. Постоји грана математике која се бави алгоритмима и назива се теорија алгоритама. Алгоритам се може строго (формално) дефинисати, но за потребе програмирања довољна је следећа, описна дефиниција.

Алгоритам је коначан скуп строго формулисаних правила за решавање задатака одређене класе и то за коначно време. Свако овакво правило у алгоритму називаћемо алгоритамски корак.

Алгоритам се састоји од низа алгоритамских корака којима се врши трансформација улазних величина све док се не дође до излазних величина алгоритма. Скуп алгоритамских корака, заједно са везама између њих, зове се структура алгоритма. Приказивање алгоритма може да буде у обичном текстуалном облику, али се могу користити и алгоритамске шеме. Алгоритамске шеме су некада биле врло популарно средство за опис алгоритма. Појавом савремених програмских језика престала је потреба за коришћењем алгоритамских шема.

Псеудо програмски језик (краће, псеудојезик) јесте језик за запис алгоритама. Он је сличан програмском језику (па отуд назив псеудо програмски језик), али може садржати и конструкције говорног језика. То значи да није строг као програмски језик. Неформално, могло би се рећи да је то језик између говорног и програмског језика.

Рачунар је систем за обраду података (информација). Да би се инфорамције могле обрађивати, потребно их је на адекватан начин представити у меморији. Меморија је састављена од ћелија које се могу наћи у два електрична стања, па се стање сваке ћелије може означити бинарном цифром 0 или 1. Једним битом могу се представити највише две могућности (на пример, информација о положеном испиту: 0 - студент пао, 1 - студент положио). Пошто се реални системи и процеси који се моделирају програмом карактеришу великим бројем стања у којима се могу наћи, то се меморијске ћелије интерпретирају по групама од по 8, 16, 32, 64 и више ћелија (бита). Како се са n бита може представити највише 2^n различитих комбинација, то се једним бајтом (8 бита) може представити $2^8 = 256$ различитих могућности. Један бајт довољан је да се представе сви симболи на стандардној тастатури рачунара. Низови нула и јединица, складиштени у меморији, били би потпуно неразумљиви када не би било познато правило пресликавања (кодирање) које дефинише значење сваке комбинације.

[7] Реч „алгоритам“ настала је од имена Мохамеда ибн Мусе ел Хорезмија (око 783-око 850), персијског математичара, кога многи сматрају оцем алгебре, важне области математике. Данас се реч „алгоритам“ најчешће користи у рачунарству, јер рачунари користе алгоритме, задате у облику наредби. Први такав алгоритам написала је 1842. године Еjда Бајрон (1815-1852), први програмер у историји.

Потребно је најпре формулисати модел рачунарске машине на којој се алгоритми извршавају и дефинисати примитивне операције које се користе приликом извршавања. Основна претпоставка је да користимо рачунарски модел који представља идеализован „реалан“ рачунар са једним процесором. Не постављамо никакву горњу границу на величину меморије таквог рачунара, али претпостављамо да програми који се на њему извршавају не могу да модификују сами себе. Тачан репертоар инструкција овог модела рачунара није посебно битан, под условом да је скуп тих инструкција сличан оном који се може наћи код правих рачунара. Зато претпоставимо да постоје аритметичке операције (за сабирање, одузимање, множење, дељење, дељење са остатком, целобројни део броја), логичке операције (и , или, негација), улазно-излазне операције (читање, писање).

Програмски језик је средство за писање програма и саопштавање програма рачунару. Уопште, језик је средство за представљање и преношење информација, као и за комуникацију између два или више корисника. Корисници програмских језика су, пре свега, човек и рачунар. Међутим, због своје егзактности, програмски језици се користе и за комуникацију међу људима (на пример, за саопштавање научних информација). Програмски језици су вештачки језици намењени нотацији и преносу специфичних информација. Природни (говорни) језици су се показали недовољно строгим јер допуштају неједнозначност и непрецизност. Тако нешто није дозвољено у програмским језицима. Дакле, једнозначност сваке конструкције програмског језика је његова најбитнија карактеристика. Програм је добар ако је (поред тога што коректно решава постављени задатак) лак за читање, лак за разумевање и лак за модификовање. Настојећи да направе што удобнији језик за запис програма, људи су направили велики број програмских језика, од којих су многи у употреби и данас. Неки од њих су: FORTRAN, COBOL, LISP, PL/1, Pascal, PROLOG, Ada, Modula 2, C, C++, Java, итд.

Применом рачунара који постају неопходни у скоро свим аспектима нашег професионалног и личног живота, проучавање алгоритама постаје неопходност за све више и више људи. Други разлог за проучавање алгоритама је њихова корисност у развијању аналитичких способности. На крају крајева, алгоритми се могу посматрати као посебне врсте решења за проблеме - не само одговоре, већ и прецизно дефинисане процедуре за добијање одговора. Сходно томе, специфичне технике пројектовања алгоритама могу се тумачити као стратегије за решавање проблема које могу бити корисне без обзира на то да ли је рачунар укључен. Наравно, прецизност која је наметнута алгоритамским размишљањем ограничава проблеме који се могу решити алгоритмом. Нећете наћи, на пример, алгоритам за срећан живот или како постати богат и славан. С друге стране, ова потребна прецизност има важну образовну предност. *Donald Knuth*, један од најистакнутијих рачунарских научника у историји алгоритама, рекао је да особа добро обучена у рачунарству зна како се носити са алгоритмима (како их конструисати, манипулисати њима, разумети их, анализирати) и да је ово знање припрема за много више од писања добрих рачунарских програма (то је ментални алат опште намене који ће

дефинитивно помоћи у разумевању других предмета, било да су то хемија, лингвистика, музика, и др.)

Дефиницију алгоритма можемо илустровати и једноставним дијаграмом:

У рачунарству, временска сложеност алгоритма представља математичку функцију која величину уноса претвара у количину времена потребну да се алгоритам изврши. Она се најчешће изражава користећи се O нотацијом која занемарује константне факторе и сабирке нижег реда величине. Када се користи на претходно описан начин, каже се да се временска сложеност означава асимптотски када величина улаза тежи бесконачности. Временска сложеност се обично процењује тако што се одреди број елементарних операција које се извршавају у алгоритму, где се претпоставља да је за извршење сваке од тих операција потребна фиксно одређена количина времена. Стога се количина времена и број елементарних операција алгоритма разликују за највише константан фактор.

Класе функција на које се често наилази приликом анализирања сложености алгоритама су (c је произвољна константа, а функције су поређане тако да су прво наведене оне које спорије расту):

ознака	**назив**
$O(1)$	константна
$O(\log n)$	логаритамска
$O((\log n)^c)$	полилогаритамска
$O(n)$	линеарна
$O(n \log n)$	квазилинеарна
$O(n^2)$	квадратна
$O(n^c)$	полиномна
$O(c^n)$	експоненцијална

Обично за полиномне алгоритме кажемо да су „брзи“ или „ефикасни“, а за експоненцијалне да су „спори“ или „неефикасни“.

2. Дељивост целих бројева

У најужем смислу, задатак теорије бројева јесте изучавање структуре прстена целих бројева $(\mathbb{Z}, +, \cdot)$. Са $\mathbb{N}$ обележавамо скуп свих позитивних целих бројева (већих од 0). Позитивни цели бројеви називају се још и природни бројеви. Скуп ненегативних целих бројева, поред позитивних целих бројева садржи још и нулу.

Дефиниција 2.1. За цео број b $(b \neq 0)$ кажемо да је делилац броја $a \in \mathbb{Z}$, односно да дели a (у ознаци $b \mid a$), ако постоји $q \in \mathbb{Z}$ тако да је $a = bq$.

На основу дефиниције лако је доказати следећу теорему.

Теорема 2.1.

i. За све $a \in \mathbb{Z}$ важи $a \mid a$.
ii. За све $a, b \in \mathbb{Z}$, ако $a \mid b$ и $b \mid a$, тада $a = \pm b$.
iii. За све $a, b, c \in \mathbb{Z}$, ако $a \mid b$ и $b \mid c$, тада $a \mid c$.

Доказ.

i. Очигледно следи из претходне дефиниције.
ii. Ако је $b = 0$, тада је и $a = 0$, па је $a = b$. Нека је $b \neq 0$. Тада из $a \mid b$ и $b \mid a$ следи да постоје цели бројеви m и n такви да је $b = ma$ и $a = nb$, одакле је $a = nma$, па је $nm = 1$. Како су n и m цели бројеви, то је или $n = m = 1$ или $n = m = -1$, односно $a = b$ или $a = -b$.
iii. Ако $a \mid b$ и $b \mid c$ тада постоје цели бројеви m и n такви да је $b = ma$ и $c = nb$, па је $c = nma$, а како је $nm \in \mathbb{Z}$ следи да $a \mid c$. ■

Према томе, ако релацију дељивости $\mid$ ограничимо на скуп позитивних целих бројева, односно скуп природних бројева $\mathbb{N}$, добијамо релацију поретка, тј. парцијално уређење овог скупа.

Теорема 2.2. Ако $d \mid a$ и $d \mid b$, онда $d \mid \alpha a + \beta b$ за све $\alpha, \beta \in \mathbb{Z}$.

Доказ. Ако $d \mid a$ и $d \mid b$ тада постоје цели бројеви m и n такви да је $a = md$ и $b = nd$. Тада је за произвољне целе бројеве α и β

$$\alpha a + \beta b = \alpha md + \beta nd = d(\alpha m + \beta n),$$

па како је $\alpha m + \beta n \in \mathbb{Z}$ то важи $d \mid \alpha a + \beta b$. ■

У теорији бројева важну улогу има следећа теорема о дељењу са остатком.

Теорема 2.3.*(Алгоритам дељења)* Сваки цео број a може се на јединствен начин помоћу датог природног броја b приказати у облику $a = bq + r$, $0 \leq r < b$, где су q и r цели бројеви. Број q се назива количником, а r остатком при дељењу броја a бројем b.

Доказ. Посматрајмо скуп целих бројева $\{\dots, a-2b, a-b, a, a+b, a+2b, \dots\}$ и изаберимо у њему најмањи број који је природан или једнак 0 (он постоји). Нека је то $a - qb$ - обележимо га са r. Тада је

(1) $$a = bq + r, \qquad 0 \leq r < b,$$

јер би у случају $r \geq b$ и број $a - (q+1)b$, који је мањи од $a - qb$, био природан или једнак нули. Тиме је доказана егзистенција бројева q и r. Докажимо јединственост тих бројева. Претпоставимо да постоји још један пар (q_1, r_1), такав да је $a = bq_1 + r_1$ и $0 \leq r_1 < b$. Одузимањем ове једнакости од једнакости (1) добијамо

$$0 = b(q - q_1) + (r - r_1),$$

односно $b \mid r - r_1$. Због $|r - r_1| < b$ имамо да је $r - r_1 = 0$, тј. $r = r_1$, а због тога је и $q = q_1$. ■

У *Теореми 2.3.* претпоставка да је b природан број може да се замени са $b \neq 0$, али је онда $0 \leq r < |b|$.

Поступак дељења са остатком нам омогућава (између осталог) да природне бројеве изражавамо у бројевним системима са датом основом (бинарном, декадном, ...).

Теорема 2.4. Нека је b цео број већи од 1. Тада се сваки позитиван цео број m може на јединствен начин представити у облику

$$m = a_n b^n + a_{n-1} b^{n-1} + \cdots + a_0,$$

где је $a_n \neq 0$ и $0 \leq a_i < b$, за све $0 \leq i \leq n$.

Доказ. Теорему доказујемо индукцијом по m. Тврђење је јасно ако је $m \in \{1, \dots, b-1\}$. Зато претпоставимо да се сваки број мањи од m на јединствен начин записује у жељеном облику, при чему је $m \geq b$. Будући да дати услови захтевају да буде $0 \leq a_0 < b$ и $b \mid (m - a_0)$, следи да a_0 мора бити управо остатак броја m при дељењу са b, тј. a_0 је јединствено одређено. Посматрајмо сада број $m' = \frac{m - a_0}{b}$, целобројни количник m при дељењу са b. Пошто је $0 < m' < m$, по индуктивној претпоставци имамо да се m' на јединствен начин записује у траженом облику:

$$m' = a_l b^l + a_{l-1} b^{l-1} + \cdots + a_2 b + a_1.$$

Како је $m = m'b + a_0$, следи да је $m = a_{l+1}b^{l+1} + a_l b^l + \cdots + a_1 b + a_0$, па добијамо жељени запис за m. Он је јединствен, јер ако би било $m = a'_k b^k + \cdots + a'_1 b + a'_0$, где је $a'_k \neq 0$ и $0 \leq a'_j < b$ за све $0 \leq j \leq k$, тада на основу ранијег закључка имамо $a'_0 = a_0$, па је

$$a'_k b^{k-1} + \cdots + a'_1 = a_l b^l + \cdots + a_1.$$

Део индуктивне претпоставке који се односи на јединственост повлачи да мора бити $k = l$ и $a'_i = a_i$ за све $1 \leq i \leq l$, што се и тражило. ■

Ако је

$$m = a_n b^n + a_{n-1} b^{n-1} + \cdots + a_0,$$

при чему важе сви услови *Теореме 2.4.*, тада кажемо да је m представљен у бројевном систему са основом b. Број b је, дакле, основа или база датог бројевног система, а број m записујемо у облику $\overline{a_n a_{n-1} \ldots a_1 a_0}_{[b]}$.

Ако је из контекста јасно у ком бројевном систему је записан неки број, онда изостављамо индекс b у запису броја. Тако, на пример, у свакодневној пракси подразумевамо да је запис броја дат у декадном систему уколико није друкчије наглашено.

У декадном систему бројеви се записују само помоћу десет симбола 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 за првих десет бројева и за описивање већих бројева користе се степени броја десет, па тако за 12 340 важи $1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10$. Моћ и једноставност овог система засигурно заслужују посебну пажњу на почетку приче о теорији бројева. На пример, овај систем омогућава ученицима да изврше сабирање као $12\,340 + 567\,890 = 580\,230$ са мало потешкоћа.

У наставку ће бити приказани познати алгоритми повезани са основним рачунским операцијама и њихови описи на псеудојезику.

Множење бројева је много софистицираније операција од сабирања и ученицима ју је теже научити. Ако су a и b неки бројеви, њихов производ је број ab објеката у правоугаоној области објеката који садржи a редова и b колона. Како је бројање ab објеката исто што и додавање b самом себи a пута, проблем израчунавања производа a и b (проблем израчунавања ab) може се свести на сабирање алгоритмом:

```
a = int(input())
b = int(input())

p=0
t=a

while t > 0:
  t = t - 1
```

```
  p = p + b

print(p)
```

„While“ петља се извршава a пута у току смањења t од a до 0 (ако је $a = 0$, петља се никада не извршава и p остаје на нули) и свако извршавање петље додаје b у p, тако да је коначна вредност p производ - број b се додаје сам себи a пута.

Овај алгоритам је неупотребљив за ручно рачунање ако је a велики број. Модерни рачунари су тако мућевити да могу брзо да помноже бројеве са 4 или 5 цифара на овај примитиван начин, али такво рачунање је бесмислено трошење њихове моћи.

Много ефикаснији, али и даље основни, алгоритам за множење је:

```
a = int(input())
b = int(input())

p=0
t=a

while t > 0:
  k = 1
  while t >= 10*k:
    k = k*10
  t = t - k
  p = p + k*b

print(p)
```

Овај алгоритам је прилагођен декадном систему у којем је множење степенима броја 10 једноставно - само се помере цифре потребног броја места улево. Уместо додавања b више пута у p, овај алгоритам проналази највећи степен броја 10 који је мањи или једнак a, означимо га са $10^e = k$, додаје 10^e пута b у p све одједном (налажење $10^e b$ не захтева, наравно, множење, само дописивање e нула после b) и смањује за 10^e број пута b који мора да се дода у p.

Овај ефикаснији алгоритам је сличан алгоритму који се учи у школи, осим што се почиње са крајњом левом цифром броја a а не са крајњом десном, и не претпоставља да је множење једноцифреним бројем лако: на пример, ако је $a = 32$ онда генерише производ као $p = (10 \cdot b) + (10 \cdot b) + (10 \cdot b) + b + b$ уместо $p = (2 \cdot b) + (30 \cdot b)$ начина на који уобичајен алгоритам ради.

Алгоритми за извођење множења преко сабирања су суштински бескорисни у практичном рачунарству. Они су важно питање којим се бави теоријско рачунарство у области заснивања бројева.

Одузимање и дељење нису увек изводљиви у скупу природних (и ненегативних целих) бројева.

Ознака $b - a$ представља број који када се дода броју a добије се b, и очигледно не постоји такав број када $b < a$. Стога се ова ознака може користити легитимно након што је $b \geq a$ доказано. На пример, у последњем алгоритму горе могло се t заменити са $t - k$ јер је k одређено тако да је $t \geq k$.

Дељење захтева слично ограничење. Ознака $\frac{a}{b}$ представља број који помножен са b даје a. Нотација $\frac{a}{b}$ ће бити коришћена само онда када је доказано да је a дељиво са b.

Како год, дељење са остатком постоји у сваком случају у ком b није 0: за дате бројеве a и b, $b \neq 0$, постоје бројеви q и r за које важи $a = qb + r$ и $r < a$ (*Теорема 2.3*). Штавише, q и r се одређују помоћу a и b следећим алгоритмом:

```
a = int(input())
b = int(input())

q = 0
r = a

while r >= b:
    r = r - b
    q = q + 1

print(q)
print(r)
```

или, ефикасније:

```
a = int(input())
b = int(input())

q = 0
r = a

while r >= b:
    k = 1
    while r > 2*k*b:
        k = k*2
    r = r - k*b
    q = q + k

print(q)
print(r)
```

Ова два алгоритма почињу са решењем $(q, r) = (0, a)$ за $a = qb + r$; они модификују (q, r) на сваком кораку тако што $a = qb + r$ остаје истинито и r се смањује, и завршавају се када $r < b$. Ако је $b = 0$, било који алгоритам бесконачно одузима 0 од a. У првом алгоритму, b се одузима од r и 1 се додаје q док је r мање од b, али у другом алгоритму се ове операције извршавају у серијама - одузимајући $2^e b$ од r и додајући 2^e у q, где је e највеће могуће.

2.1 Степеновање

За $b > 0$ дефинишимо $a \mod b = r$, где је r остатак добијен алгоритмом дељења броја a бројем b, тј. $a = qb + r$ и $0 \leq r < b$.

На пример: $23 \mod 7 = 2$, јер је $23 = 3 \cdot 7 + 2$ и $-4 \mod 5 = 1$, јер је $-4 = (-1) \cdot 5 + 1$.

За извођење модуларних операција у рачунарству веома су важни идентитети:

1. $(x + y) \mod m = (x \mod m + y \mod m) \mod m$
2. $(x \cdot y) \mod m = (x \mod m \cdot y \mod m) \mod m$
3. $(x - y) \mod m = (x \mod m - y \mod m) \mod m$

У наставку ћемо доказати идентитет за сабирање и множење по модулу.

1. На основу *Теореме 2.3.* имамо $x = q_x m + r_x$ и $y = q_y m + r_y$, $0 \leq r_x, r_y < m$ где су $q_x, q_y, r_x, r_y \in \mathbb{Z}$. Тада важи да је
$$x + y = q_x m + r_x + q_y m + r_y = (q_x + q_y)m + r_x + r_y.$$
Ако важи $r_x + r_y = qm + r$ за $0 \leq r < m$, тада је $x + y = q_x m + r_x + q_y m + r_y = (q_x + q_y)m + qm + r = (q_x + q_y + q)m + r$, па је $(x + y) \mod m = r$. Важи да је $(x \mod m + y \mod m) \mod m = (r_x + r_y) \mod m = r$, чиме је тврђење доказано.
2. На основу *Теореме 2.3.* имамо $x = q_x m + r_x$ и $y = q_y m + r_y$, $0 \leq r_x, r_y < m$ где су $q_x, q_y, r_x, r_y \in \mathbb{Z}$. Тада важи да је
$$x \cdot y = (q_x m + r_x) \cdot (q_y m + r_y) = q_x q_y m^2 + q_x m r_y + r_x q_y m + r_x r_y =$$
$$= (q_x q_y m + q_x r_y + q_y r_x)m + r_x \cdot r_y.$$
Ако важи $r_x \cdot r_y = qm + r$ за $0 \leq r < m$, тада је $x \cdot y = (q_x q_y m + q_x r_y + q_y r_x)m + qm + r = (q_x q_y m + q_x r_y + q_y r_x + q)m + r$, па је $(x \cdot y) \mod m = r$. Важи да је $(x \mod m \cdot y \mod m) \mod m = (r_x \cdot r_y) \mod m = r$, чиме је тврђење доказано.

Када су у питању мањи бројеви обично ручно можемо израчунати вредност степена. Уколико се бројеви повећавају долазимо до проблема јер степеновање постаје све теже и резултат је веома велики број што изискује и више времена за рачунање.

Проблем 2.1. За дато x и ненегативан цео број n, израчунати x^n.

Претпоставићемо да је x елемент скупа са добро дефинисаном операцијом (асоцијативан са јединицом). Иако се x^{16} на очигледан начин може израчунати са 15 множења, брже је израчунати помоћу 4 квадрата. Уопште, из $n = \sum a_i 2^i$, где је $a_i \in \{0, 1\}$, следи:

(2) $$x^n = x^{a_0}(x^2)^{a_1}(x^4)^{a_2} \ldots$$

што сугерише паметан начин за комбиновање множења и квадрирања.

Здесна-налево степеновање

```
import math
import decimal

x = decimal.Decimal(input())
n = int(input())

y = 1
while n > 0:
    if n%2 == 0:
        y = x*y
    x = x*x
    n = math.floor(n/2)

print(y)
```

Исправност алгоритма је јасна из (2), будући да се x^{2^k} множи у y ако и само ако k-ти бит a_k бинарног записа од n је ненула. То се формално може доказати показујући индукцијом да на почетку у „While“ петљи, $X^N = x^n y$ остаје, где X и N означавају почетне вредности променљивих x и n. Када је n једнако 0, једначина каже да $X^N = y$, тако да је y тражено решење.

Уобичајена индуктивна дефиниција за $Exp(x, n) := x^n$ даје очигледан рекурзивни алгоритам:

$$Exp(x,n) = \begin{cases} 1, & \text{ако је } n = 0, \\ Exp\left(x^2, \frac{n}{2}\right), & \text{ако је } n > 0 \text{ парно}, \\ x \cdot Exp\left(x^2, \frac{n-1}{2}\right), & \text{ако је } n \text{ непарно}. \end{cases}$$

Искусни програмери често имплементирају рекурзивне верзије алгоритама због њихове елеганције и очигледне исправности, а по потреби их претварају у еквивалентне, а можда и брже, итеративне (нерекурзивне) алгоритме. Ако се то уради рекурзивном програму, резултат је горе наведени алгоритам Здесна-налево степеновање.

Занимљиво, ако је индуктивна дефиниција замењена математички еквивалентним алгоритмом у којем квадрирање прати рекурзивне позиве,

$$Exp(x,n) = \begin{cases} 1, & \text{ако је } n = 0, \\ Exp\left(x, \frac{n}{2}\right)^2, & \text{ако је } n > 0 \text{ парно}, \\ x \cdot Exp\left(x, \frac{n-1}{2}\right)^2, & \text{ако је } n \text{ непарно}. \end{cases}$$

тада је одговарајући итеративни алгоритам заиста другачији.

Слева-надесно степеновање

```
import math
import decimal

x = decimal.Decimal(input())
n = int(input())
m = int(input())

y = 1
while m > 1:
    m = math.floor(m/2)
    y = y*y
    if n >= m:
        y = x*y
        n = n - m

print(y)
```

Уносимо: x, ненегативан цео број n, степен двојке $m = 2^a$ тако да $\frac{m}{2} \leq n < m$.

Тачност следи индуктивно доказивањем да је на почетку „While“ петље, $n < m$ и $y^m x^n = x^N$. За разлику од претходног алгоритма, ова верзија користи битове a_i у бинарном запису од n почевши од крајњег левог (најзначајнијег) бита.

Сложеност било које верзије овог алгоритма је $O(\log n)$ јер је број операција ограничен са $2 \cdot size_2(n)$. Ова изузетна ефикасност има бројне примене у алгоритамској теорији бројева. Потребно је нагласити да идеја рачунања x^n поновљеним множењем са x захтева време $O(n)$.

У посебном, али важном практичном случају, верзија слева-надесно степеновања је боља од верзије здесна-налево. Претпоставимо да је наша операција „множење по модулу N“ и да је x мали у односу на N. Тада ће множење са оригиналним x вероватно трајати краће него модуларно множење произвољним целим бројем X у распону $0 \leq X < N$. Верзија слева-надесно чува оригинално x (иако квадрати укључују произвољне целе бројеве), док верзија здесна-налево модификује x и тако обавља готово све операције на произвољним елементима. Другим речима, с другачијим рачунарским моделом (битна сложеност, са специфичном основном операцијом „множи по модулу N“ и x мало) алгоритам слева-надесно, било рекурзивни или итеративни, значајно је бољи од здесна-налево степеновања.

Ова дискусија о рачунању вредности степена једва да загребе површину великог дела теоријског и практичног рада. Велики значај степеновања довео је до многих значајних практичних побољшања.

3. Еуклидов алгоритам

Еуклид је аутор првих сачуваних теоријских радова из математике. Најзначајније Еуклидово дело су „Елементи“ који се састоје од тринаест књига величине поглавља. Иако су у „Елементима“ приказани већином резултати других математичара, Еуклидов допринос кроз логички след организације садржаја постао је врло битан за даљи развој математике.

Један од најважнијих Еуклидових резултата, наведених у књизи VII, је тзв. Еуклидов алгоритам за одређивање највећег заједничког делиоца два цела броја.

Еуклидов алгоритам је вероватно најстарији нетривијални алгоритам. Врло је интересантно да модерни рачунари користе 2 200 година стар алгоритам. Он има широку и практичну примену. Представља кључан елемент RSA алгоритма, методе асиметричне криптографије која се у значајној мери примењује у електронском пословању. Користи се при решавању линеарних Диофантових[8] једначина, на пример код одређивања бројева које задовољавају вишеструке конгруенције (кинеска теорема о остацима). Може се употребити за конструисање верижних разломака и неколико савремених алгоритама за факторизацију природних бројева.

Дефиниција 3.1. За $a, b \in \mathbb{Z}$ највећи заједнички делилац бројева a и b, у ознаци $НЗД(a, b)$, јесте број $d \in \mathbb{Z}$ за који важи:

i. $d \mid a, d \mid b$;
ii. за све $c \in \mathbb{Z}$ такве да $c \mid a, b$ важи $|c| \leq |d|$.

Приметимо да пар $(0,0)$ нема највећи заједнички делилац; међутим, за сваки други пар целих бројева постоје тачно два цела броја који задовољавају горње услове, и они су један другом супротни. Ако је d највећи заједнички делилац за a, b, то пишемо и $d = (a, b)$, при чему се ова нотација најчешће односи на позитиван НЗД за a и b.

[8] Диофант је био грчки математичар који је живео у Александрији у трећем веку наше ере. Диофантово велико дело је „Аритметика“. Састојала се од 13 књига, од којих је шест сачувано. Диофантова дела су представљала почетну тачку за истраживања Фермаа, Ојлера, Гауса и других математичара у области теорије бројева.

С обзиром на важност Еуклидовог алгоритма, из историјских и практичних разлога, размотрићемо како га је сам Еуклид третирао. Оригинална верзија заснована је на одузимању.

Посматрамо два позитивна цела броја a и b. Хоћемо да одредимо њихов НЗД. Ако је $a = b$, тада је њихов највећи заједнички делилац баш тај број. У супротном, проблем се своди на налажење највећег заједничког делиоца мањег од два посматрана броја и разлике та два броја. Ако је $a > b$, тада важи $\text{НЗД}(a, b) = \text{НЗД}(a - b, b)$. Заиста, ако неки број d дели бројеве a и b, онда он дели и њихову разлику. Дакле, $d = \text{НЗД}(a, b)$ сигурно дели и $a - b$ и b. Ако d не би био највећи заједнички делилац бројева $a - b$ и b, тада би постојао неки већи број d' који би делио и $a - b$ и b. У том случају, d' би делио и збир $a = (a - b) + b$, па би био делилац бројева a и b, који је већи од d. Долазимо до контрадикције са претпоставком да је d највећи заједнички делилац бројева a и b. Ако је $a < b$, тада се слично доказује да је $\text{НЗД}(a, b) = \text{НЗД}(a, b - a)$.

Алгоритам заснован на одузимању изгледа овако:

```
a = int(input())
b = int(input())

while a != b:
  if a > b:
    a = a - b
  else:
    b = b - a
print(a)
```

Најгори случај наступа када је разлика између два броја јако велика (ако је $a = 1$, он ће се одузимати од b све док он не постане 1, за шта је потребно $b - 1$ корака). Сложеност овог алгоритма је линеарна у односу на већи од два броја, што се може записати као $O(\max(a, b))$ или $O(a + b)$. Ако се сложеност рачуна у односу на број цифара броја (што је величина улаза), онда је она заправо експоненцијална.

Пример 3.1. Наћи највећи заједнички делилац за бројеве 803 и 154.

$(803,154) = (649,154)$,
$(649,154) = (495,154)$,
$(495,154) = (341,154)$,
$(341,154) = (187,154)$,
$(187,154) = (33,154)$,
$(33,154) = (33,121)$,
$(33,121) = (33,88)$,
$(33,88) = (33,55)$,
$(33,55) = (33,22)$,
$(33,22) = (11,22)$,
$(11,22) = (11,11)$.

Дакле, $(803,154) = 11$.

У модерним терминима користи се верзија Еуклидовог алгоритма са дељењем.

Теорема 3.1. *(Еуклидов алгоритам)* Нека су $r_0 = a$ и $r_1 = b$ ненегативни цели бројеви и $b \neq 0$. Узастопном применом алгоритма дељења добијамо $r_j = q_{j+1}r_{j+1} + r_{j+2}$, $0 < r_{j+2} < r_{j+1}$ за $j = 0,1,2,\dots,n-2$. Алгоритам се завршава када је остатак дошао до 0 и последњи ненула остатак је (a,b).

Из ове теореме увиђамо да је највећи заједнички делилац бројева a и b последњи ненула остатак у низу једначина генерисаних узастопном применом алгоритма дељења, где се у сваком кораку дељеник и делилац замењују мањим бројевима, делиоцем и остатком.

Да бисмо доказали да Еуклидов алгоритам даје НЗД, следећа лема ће бити од користи.

Лема 3.1. Ако су $c, d \in \mathbb{Z}$ и $c = qd + r$, онда $(c,d) = (d,r)$.

Доказ. Ако цео број e дели и c и d, онда из $r = c - qd$, према *Теореми 2.2.*, следи $e \mid r$. Пошто важи $c = qd + r$, опет према Теореми 2.2., добијамо да $e \mid c$. Како су заједнички делиоци за бројеве c и d, такође, заједнички делиоци за d и r, следи $(c,d) = (d,r)$. ■

Сада ћемо доказати да Еуклидов алгоритам ради.

Доказ. Нека су $r_0 = a$ и $r_1 = b$ позитивни цели бројеви, $a \geq b$. Узастопном применом алгоритма дељења добијамо:

$$
\begin{aligned}
r_0 &= q_1 r_1 + r_2, && \text{где је } 0 \leq r_2 < r_1,\\
r_1 &= q_2 r_2 + r_3, && \text{где је } 0 \leq r_3 < r_2,\\
&\dots && \dots\\
r_{n-3} &= q_{n-2} r_{n-2} + r_{n-1}, && \text{где је } 0 \leq r_n < r_{n-1},\\
r_{n-2} &= q_{n-1} r_{n-1} + r_n, && \text{где је } 0 \leq r_n < r_{n-1},\\
r_{n-1} &= q_n r_n.
\end{aligned}
$$

Можемо претпоставити да на крају добијамо остатак нула, јер низ остатака $a = r_0 > r_1 > r_2 > \dots \geq 0$ не може садржати више од a чланова. Према *Леми 3.1.* имамо $(a,b) = (r_0,r_1) = (r_1,r_2) = (r_2,r_3) = \dots = (r_{n-2},r_{n-1}) = (r_{n-1},r_n) = (r_n,0) = r_n$. Следи $(a,b) = r_n$ је последњи ненула остатак. ■

Можемо приметити да је при сваком дељењу остатак строго мањи од остатка при претходном дељењу. Због тога закључујемо да се Еуклидов алгоритам сигурно завршава, тј. након коначно много дељења с остатком доћи ћемо до дељивости, односно до дељења без остатка.

Алгоритам заснован на дељењу следи:

```
a = int(input())
b = int(input())

while b != 0:
  r = a % b
  a = b
  b = r
print(a)
```

Приметимо да се после највише једног корака осигурава да је $a > b$ (јер се у сваком кораку пар (a, b) мења паром $(b, a \ mod \ b)$, а увек важи да је $a \ mod \ b < b$). После било које две итерације се од пара (a, b) долази до пара $\big(a \ mod \ b, b \ mod (a \ mod \ b)\big)$ (наравно, под претпоставком да је $b \neq 0$ и да је $a \ mod \ b \neq 0$). Докажимо да је $a \ mod \ b < \frac{a}{2}$. Ако је $b \leq \frac{a}{2}$, тада је $a \ mod \ b < b \leq \frac{a}{2}$. У супротном, за $b > \frac{a}{2}$ важи да је $a \ mod \ b = a - b < \frac{a}{2}$. Зато се први аргумент после свака два корака смањи бар двоструко. До вредности 1 први аргумент стигне у логаритамском броју корака у односу на већи од полазна два броја и тада други број сигурно достиже 0 (јер је строго мањи од првог) и поступак се завршава. Дакле, сложеност је логаритамска у односу на већи од два броја, што може да се запише и као $O(\log(a + b))$. Ако се сложеност рачуна у односу на број цифара броја (што је величина улаза), онда је она заправо линеарна.

Пример 3.2. Наћи највећи заједнички делилац за бројеве 803 и 154.

$$\begin{aligned} (803{,}154) &= (154{,}33), && \text{јер је } 803 = 5 \cdot 154 + 33; \\ (154{,}33) &= (33{,}22), && \text{јер је } 154 = 4 \cdot 33 + 22; \\ (33{,}22) &= (22{,}11), && \text{јер је } 33 = 1 \cdot 22 + 11; \\ (22{,}11) &= (11{,}0), && \text{јер је } 22 = 2 \cdot 11 + 0; \\ (11{,}0) &= 11. \end{aligned}$$

Дакле, $(803{,}154) = 11$.

Закључујемо да је варијанта заснована на дељењу ефикаснија од варијанте засноване на одузимању.

Теорема 3.2. Нека су $a, b, c \in \mathbb{Z}$ такви да (a, b) постоји, $c \mid a$ и $c \mid b$. Тада $c \mid (a, b)$.

Доказ. Ово значајно тврђење доказујемо применом Еуклидовог алгоритма за налажење највећег заједничког делиоца два броја. Дакле, ако је, на пример, $b \neq 0$, тада смо извршили следећа целобројна дељења:

$$\begin{aligned} a &= q_1 b + r_1, && \text{где је } 0 \leq r_1 < |b|, \\ b &= q_2 r_1 + r_2, && \text{где је } 0 \leq r_2 < r_1, \\ r_1 &= q_3 r_2 + r_3, && \text{где је } 0 \leq r_3 < r_2, \end{aligned}$$

...	...
$r_{k-1} = q_{k+2}r_k + r_{k+1},$	где је $0 \leq r_{k+1} < r_k,$
...	...
$r_{n-2} = q_n r_{n-1} + r_n,$	где је $0 \leq r_n < r_{n-1},$
$r_{n-1} = q_{n+1}r_n$	$(r_{n+1} = 0).$

Приметимо да се поступак сигурно завршава у коначно много корака, будући да низ

$$|b| > r_1 > r_2 > \cdots > r_k > \cdots$$

мора бити коначан.

Тврдимо да је $r_n = (a, b)$, одакле одмах следи тврђење теореме, будући да се лако показује да за сваки заједнички делилац c бројева a, b мора бити $c \mid r_k$ за све k (па тако и за $k = n$); наиме, из $r_{k-2} = q_{k+1}r_{k-1} + r_k$ добијамо $r_k = r_{k-2} - q_{k+1}r_{k-1}$, па закључујемо да из (индуктивне) претпоставке $c \mid r_{k-2}$, $c \mid r_{k-1}$ следи $c \mid r_k$.

Пошто малопређашњи след закључака важи за сваки заједнички делилац c бројева a, b, следи да он важи и за $c = (a, b)$; због тога одмах имамо $(a, b) \mid r_n$, а самим тим и $(a, b) \leq r_n$. С друге стране, покажимо да r_n јесте заједнички делилац за a и b. Непосредно, имамо да $r_n \mid r_{n-1}$. Сада претпоставка да $r_n \mid r_{k+1}$ и $r_n \mid r_{k+2}$ повлачи, на основу једнакости $r_k = q_{k+2}r_{k+1} + r_{k+2}$, да $r_n \mid r_k$. Тако долазимо до закључка да $r_n \mid a$ и $r_n \mid b$. По дефиницији највећег заједничког делиоца, одавде следи $r_n \leq (a, b)$. Према томе, $r_n = (a, b)$, као што се и тражило. ■

Последица 3.1. За све $a, b \in \mathbb{Z}$, $c \in \mathbb{N}$ важи $(ca, cb) = c(a, b)$.

Доказ. Заправо, ово је више последица доказа него самог тврђења претходне теореме. Наиме, посматрајмо једнакости које смо добили током Еуклидовог алгоритма за израчунавање (a, b): овај алгоритам резултује последњим ненула остатком $r_n = (a, b)$. Помножимо сада све те једнакости са c; на тај начин добијамо управо једнакости које проистичу из инстанце Еуклидовог алгоритма за налажење (ca, cb). Последњи ненула остатак који тај алгоритам даје је баш $(ca, cb) = cr_n = c(a, b)$. ■

Следећа теорема даје једну веома интересантну и корисну особину највећег заједничког делиоца.

Теорема 3.3. *(Безуова теорема)* Највећи заједнички делилац бројева $a, b \in \mathbb{Z}$ се може изразити у облику

$$(a, b) = \alpha a + \beta b$$

за погодно одабране $\alpha, \beta \in \mathbb{Z}$.

Доказ. Индукцијом по k доказујемо да се сваки остатак r_k у Еуклидовом алгоритму за израчунавање (a, b) може изразити као $r_k = \alpha_k + \beta_k$ за неке $\alpha_k, \beta_k \in \mathbb{Z}$. Заиста, то је тачно за саме бројеве $a = 1 \cdot a + 0 \cdot b$ и $b = 0 \cdot a + 1 \cdot b$, као и за $r_1 = a - q_1 b = 1 \cdot a + (-q_1) \cdot b$. Зато претпоставимо да важи $r_{k-1} = \alpha_{k-1} a + \beta_{k-1} b$ и $r_k = \alpha_k a + \beta_k b$ за неке $\alpha_{k-1}, \alpha_k, \beta_{k-1}, \beta_k \in \mathbb{Z}$. Тада је

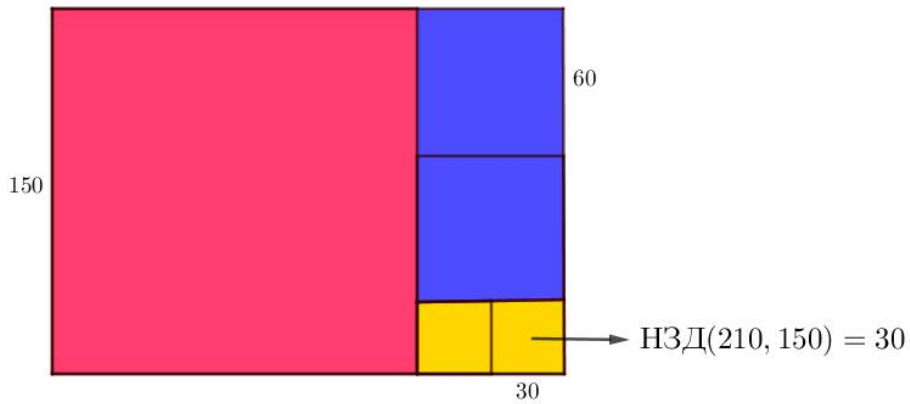
$$r_{k+1} = r_{k-1} - q_{k+1} r_k = (\alpha_{k-1} - q_{k+1}\alpha_k)a + (\beta_{k-1} - q_{k+1}\beta_k)b,$$

па је сада довољно дефинисати $\alpha_{k+1} = \alpha_{k-1} - q_{k+1}\alpha_k$ и $\beta_{k+1} = \beta_{k-1} - q_{k+1}\beta_k$. Специјално, следи $(a, b) = r_n = \alpha_n a + \beta_n b$, па $\alpha = \alpha_n$ и $\beta = \beta_n$ представљају адекватан избор тражених коефицијената. ■

Ово тврђење има значајну последицу у вези са решивошћу линеарне Диофантове једначине $ax + by = c$, о чему ће се говорити у једном од наредних поглавља.

Еуклидов алгоритам се може илустровати и геометријски. Посматрамо правоугаоник дужине a и ширине b. Потребно је одредити највећу дужину странице квадрата таква да се правоугаоник може поплочати квадратима те димензије.

Приказаћемо овај поступак на конкретном примеру. Посматрамо правоугаоник чије су странице 210 и 150. У првом кораку уочавамо да можемо исећи један квадрат димензија 150 пута 150 (на слици приказан црвеном бојом). Јасно је да ако неким мањим квадратима успемо да поплочамо преостали правоугаоник, да ћемо тим квадратима успети да поплочамо и овај квадрат 150 пута 150 (јер ће димензија тих малих квадрата делити број 150), па ћемо самим тим моћи поплочати и цео полазни правоугаоник димензија 210 пута 150. Од правоугаоника 150 пута 60 можемо исећи два квадрата димензије 60 пута 60 (на слици приказани плавом бојом) и преостаће нам правоугаоник димензије 60 пута 30. Њега можемо разложити на два квадрата димензије 30 пута 30 (на слици приказани жутом бојом) и то је највећа димензија квадрата којим се може поплочати полазни правоугаоник.



Дефиниција 3.2. За целе бројеве a и b кажемо да су узајамно прости ако је $(a, b) = 1$.

Нека су $a_1, a_2, \dots, a_n$ цели бројеви, при чему је бар један различит од нуле. Ако је $НЗД(a_i, a_j) = 1$, за $1 \leq i < j \leq n$, кажемо да су бројеви $a_1, a_2, \dots, a_n$ по паровима узајамно прости. Ако је $НЗД(a_1, a_2, \dots, a_n) = 1$, онда су бројеви $a_1, a_2, \dots, a_n$ узајамно прости. На пример, бројеви 6, 10 и 15 су узајамно прости а нису узајамно прости по паровима. То важи и за бројеве 6, 15 и 35. Бројеви 18, 25 и 77 су по паровима узајамно прости.

За цео број k кажемо да је заједнички садржалац целих бројева a и b ако је $a \mid k$ и $b \mid k$. Најмањи позитиван цео број m који је заједнички садржалац бројева a и b назива се најмањи заједнички садржалац бројева a и b. Обележаваћемо га са $НЗС(a, b)$. Јасно је да је

$$НЗС(a, b) = НЗС(-a, b) = НЗС(a, -b) = НЗС(-a, -b).$$

Прецизније, $m = НЗС(a, b)$ ако важи:

i. $a \mid m$ и $b \mid m$;
ii. ако је $a \mid k$ и $b \mid k$, онда је $m \mid k$.

Најмањи заједнички садржалац бројева a и b означавамо са $[a, b]$.

3.1 Проширени Еуклидов алгоритам

Теорема 3.3. нам омогућава да највећи заједнички делилац два природна броја a и b можемо изразити као њихову целобројну линеарну комбинацију. Овај поступак ћемо звати проширени Еуклидов алгоритам. Уместо α и β, означимо тражене целобројне коефицијенте са x и y, тј. (a, b) ћемо представити као $xa + yb$. Можемо написати одговарајућу итеративну верзију овог алгоритма. Применом идеје основног Еуклидовог алгоритма за одређивање највећег заједничког делиоца бројева a и b, добија се опадајући низ остатака

$$r_{i-1} = q_i r_i + r_{i+1}, \quad 0 \leq r_{i+1} < r_i, \text{за } \; i = 1,2, \dots, k.$$

За $r_0 = a$ важи да је $x = 1$ и $y = 0$, а за $r_1 = b$ важи $x = 0$ и $y = 1$. Желимо да r_{i+1} представимо као целобројну линеарну комбинацију бројева a и b, ако знамо да важи $r_{i-1} = x_{i-1}a + y_{i-1}b$ и $r_i = x_i a + y_i b$. На основу везе $r_{i+1} = r_{i-1} - q_i r_i$ добијамо:

$$r_{i+1} = x_{i-1}a + y_{i-1}b - q_i(x_i a + y_i b) = (x_{i-1} - q_i x_i)a + (y_{i-1} - q_i y_i)b.$$

Дакле, вредности x_{i+1} и y_{i+1} можемо израчунати на основу претходних вредности x_{i-1} и x_i, односно y_{i-1} и y_i.

Следећи псеудокод приказује поступак за налажење целобројних коефицијената x и y:

```
a = int(input())
b = int(input())

x_preth = 1
y_preth = 0

x_tren = 0
y_tren = 1

while b != 0:
    q = a//b
    r = a - q*b
    a = b
    b = r

    x_pom = x_preth - q*x_tren
    x_preth = x_tren
    x_tren = x_pom

    y_pom = y_preth - q*y_tren
    y_preth = y_tren
    y_tren = y_pom

x = x_preth
y = y_preth

print(x)
print(y)
```

Број операција је пропорционалан са бројем операција у Еуклидовом алгоритму, тј. једнак је $O(\log(a+b))$.

Пример 3.3.: Одредити целобројну линеарну комбинацију за бројеве 33 и 24.

Прво ћемо применити основни Еулидов алгоритам за налажење највећег заједничког делиоца:

$$\begin{aligned} 33 &= 1 \cdot 24 + 9, \\ 24 &= 2 \cdot 9 + 6, \\ 9 &= 1 \cdot 6 + 3, \\ 6 &= 2 \cdot 3. \end{aligned}$$

Дакле, $(33,24) = 3$. Потребно је још одредити одговарајуће коефицијенте x и y.

$x_0 = 1$
$x_1 = 0$
$x_2 = 1 - 1 \cdot 0 = 1$
$x_3 = 0 - 2 \cdot 1 = -2$
$x_4 = 1 - 1 \cdot (-2) = 3$

$y_0 = 0$
$y_1 = 1$
$y_2 = 0 - 1 \cdot 1 = -1$
$y_3 = 1 - 2 \cdot (-1) = 3$
$y_4 = -1 - 1 \cdot 3 = -4$

$x_5 = -2 - 2 \cdot 3 = -8$ $y_5 = 3 - 2 \cdot (-4) = 11$

Одавде следи да су тражени коефицијенти $x = 3$ и $y = -4$, тј. $33 \cdot 3 + 24 \cdot (-4) = 3$.

Тврђење 3.1.: Нека су $a, b, c \in \mathbb{Z}$ тако да је $a \neq 0$ или $b \neq 0$. Тада Диофантова једначина $ax + by = c$ има решење ако и само ако $(a, b) \mid c$.

Доказ. ($\Rightarrow$): Нека је (x_0, y_0) неко решење дате једначине. Пошто $(a, b) \mid a$ и $(a, b) \mid b$, важи

$$(a, b) \mid ax_0 + by_0 = c.$$

($\Leftarrow$): Претпоставимо да $(a, b) \mid c$, тј. да је $c = (a, b)c'$. Према *Теореми 3.3.*, постоје $\alpha, \beta \in \mathbb{Z}$ тако да је $(a, b) = \alpha a + \beta b$. То значи да је

$$c = a(\alpha c') + b(\beta c'),$$

односно, $x = \alpha c', y = \beta c'$ је једно решење дате једначине. ■

Значај проширеног Еуклидовог алгоритма је у томе што омогућава решавање једне класе Диофантових једначина, тзв. линераних Диофантових једначина које су поменутог облика $ax + by = c$, где су a, b, c дати цели бројеви, а x и y су цели бројеви које треба одредити.

Сада ћемо приказати проширени Еуклидов алгоритам заснован на одузимању.

Нека су a и b ненула бројеви и нека је d њихов највећи заједнички делилац. Другим речима, d је величина најмањег корака који неко може направити комбинујући величину корака a и величину корака b. Проширени Еуклидов алгоритам одређује како направити корак величине d користећи кораке величине a и кораке величине b. Прецизније, одређује два начина да се направи корак величине d користећи кораке величине a и кораке величине b.

Алгоритам налази решења u, v, x, y следеће две једначине:

(3)
$$d + ub = va$$
$$d + xa = yb$$

где су a и b дати ненула бројеви и d је њихов највећи заједнички делилац. Може се направити корак величине d удесно или направити v корака величине a удесно и u корака величине b улево, или направити y корака величине b удесно и x корака величине a улево.

Проширени Еуклидов алгоритам може бити формулисан и на следећи начин:

```
a = int(input())
b = int(input())

d = a
e = b

u = 0
v = 1

x = 0
y = 1

while d != e:
    if d > e:
        d = d - e
        u = u + y
        v = v + x
    else:
        e = e - d
        x = x + v
        y = y + u

print(u)
print(v)
print(x)
print(y)
```

Ако се занемаре u, v, x и y, алгоритам је обичан Еуклидов алгоритам „одузмите мање од већег“ и завршавају се оба и d и e са највећим заједничким делиоцем за a и b. Једначине $d + ub = va$ и $e + xa = yb$ важе на почетку јер $d + 0 \cdot b = 1 \cdot a$ и $e + 0 \cdot a = 1 \cdot b$ важе на почетку. Сваки корак чува истинитост ових једначина, као што се може видети у наставку. Ако $d + ub = va$ и $e + xa = yb$ важе, њихов збир је $d + (u + y)b = e + (v + x)a$. Ако је $d > e$, корак алгоритма оставља e, x и y непромењеним, па једначина $e + xa = yb$ остаје истинита, док $d + ub = va$ постаје $(d - e) + (u + y)b = (v + x)a$, а то је тачно, што резултира када је e одузето и од једне и од друге стране. На исти начин, ако је $d < e$, једначина која садржи d је непромењена и једначина која садржи e је промењена у једначину добијену одузимањем d од обе стране збира. Тако, обе једначине су истините у сваком кораку, укључујући последњи корак, у ком је $d = e$, који показује да су добијене једначине истините.

Како алгоритам фунционише може се јасно видети ако су кораци приказани у табеларној форми, са једном колоном за свако d, e, u, v, x и y и са по једним редом за сваки корак алгоритма. На пример, за $a = 23$ и $b = 14$, табела изгледа овако:

d	e	u	v	x	y
23	14	0	1	0	1
9	14	1	1	0	1
9	5	1	1	1	2
4	5	3	2	1	2

4	1	3	2	3	5
3	1	8	5	3	5
2	1	13	8	3	5
1	1	18	11	3	5

и завршава се једначинама $1 + 18 \cdot 14 = 11 \cdot 23$ и $1 + 3 \cdot 23 = 5 \cdot 14$.

Једна од главних чињеница примењене теорије бројева јесте да је проширени Еуклидов алгоритам прилично практичан, чак и када су унесени бројеви огромни. Тако, решења за (3) за било који дат пар ненула бројева a и b може се са лакоћом наћи. Како год, ово важи само ако се алгоритам модификује, тако да он одузима погодне множиоце мањег од већег:

```
a = int(input())
b = int(input())

d = a
e = b

u = 0
v = 1

x = 0
y = 1

while d != e:
    k = 1
    while d > 2*k*e or e > 2*k*d:
        k = k*2
    if d > e:
        d = d - k*e
        u = u + k*y
        v = v + k*x
    else:
        e = e - k*d
        x = x + k*v
        y = y + k*u

print(u)
print(v)
print(x)
print(y)
```

Ова „убрзана“ верзија основног алгоритма налази највећи степен броја 2, означимо га са $k = 2^e$, за који ће основни алгоритам понављати исти корак k пута у реду, и извршава ових k корака одједном. У примеру горе, убрзана верзија алгоритма производи исти рачун као и основни алгоритам, само што на крају, уместо да одузме 1 од 4 три пута заредом, убрзани алгоритам прво га одузме два пута у једном кораку, а затим га поново одузме. Ефекат у овом случају је једноставно прескакање трећег реда са дна табеле, што је једва побољшање у рачунању. С друге стране, убрзани алгоритам је огромно побољшање ако се

у једном тренутку алгоритма нађе корак у ком мањи од d и e је много мањи, као што је то случај, на пример, у првом кораку алгоритма када се примени на $a = 1002$ и $b = 5$.

4. Прости бројеви

Дефиниција 4.1. Цео број $p > 0$ је прост број ако не постоји делитељ d од p за који је $1 < d < p$ (тј. ако су 1 и p једини позитивни делитељи p). Ако цео број $a > 1$ није прост, каже се да је a сложен број.

Први кључни резултат теорије бројева који практично представља „одскочну даску“ за било каква озбиљнија испитивања структуре прстена $\mathbb{Z}$ јесте основна теорема аритметике.

Теорема 4.1.*(Основна теорема аритметике)* Сваки природан број $n > 1$ може се приказати као производ (позитивних) простих бројева и при томе је та факторизација јединствена до на поредак фактора: другим речима, ако важи

$$n = p_1 p_2 \dots p_r = q_1 q_2 \dots q_s,$$

где су p_i, q_j прости бројеви за све $1 \leq i \leq r, 1 \leq j \leq s$, тада је $r = s$ и постоји пермутација π скупа $\{1,2, \dots, r\}$ тако да је $p_i = q_{\pi(i)}$ за све $1 \leq i \leq r$.

Доказ:

Егзистенција: Тврђење да постоји разлагање броја $n > 1$ на просте факторе доказујемо (тоталном) индукцијом. Оно је евидентно за $n = 2$, пошто је посреди прост број. Зато претпоставимо да сви бројеви из $\{2, \dots, n-1\}$ имају бар по једно разлагање у производ простих бројева.

Ако је сам број n прост, тада нема шта да се доказује; у супротном, нека је $p > 1$ најмањи нетривијални делилац броја n. Очито, p мора бити прост број, јер би у супротном n имао делилац мањи од p, што је у супротности са избором p. Према томе, важи $n = pn'$, где је $1 < n' < n$; због тога је индуктивна претпоставка применљива на n', тј. n' је производ простих бројева: $n' = p_1 \dots p_m$. Но, тада је $n = pp_1 \dots p_m$, што окончава индуктивни доказ.

Јединственост: Претпоставимо да је $n = p_1 p_2 \dots p_r = q_1 q_2 \dots q_s$; без умањења општости, нека је $r \leq s$. Пошто $p_1 \mid q_1 q_2 \dots q_s$ и p_1 је прост број, закључујемо да $p_1 \mid q_{j_1}$ за неко j_1; међутим, и q_{j_1} је, као и p_1, (позитиван) прост број, па је $p_1 = q_{j_1}$. Исто закључивање се може поновити и за $p_2, \dots, p_r$, па за свако $1 \leq i \leq r$ постоји индекс j_i тако да је $p_i = q_{j_i}$. При томе су сви индекси $j_1, \dots, j_r$ међусобно различити. Уколико би било $r < s$, тада

бисмо, по означавању $\{k_1, \dots, k_{s-r}\} = \{1, \dots, s\} \setminus \{j_1, \dots, j_r\}$, добили да је $1 = q_{k_1} \dots q_{k_{s-r}}$, што је очито немогуће. Дакле, мора бити $r = s$; осим тога, пермутација π дефинисана са $\pi(i) = j_i$ $(1 \leq i \leq r)$ има све тражене особине. ■

Основна теорема аритметике, коју смо управо доказали, веома се често користи. Иако тврђење изгледа очигледно, то уопште није тако.

Теорема 4.2. Позитиван цео број n је сложен ако и само ако има прост фактор p, такав да је $p \leq \sqrt{n}$.

Доказ. Ако n има прост фактор $p \leq \sqrt{n}$, онда је n, очигледно сложен број.

Обрнуто, ако је p најмањи прост фактор сложеног броја n, тада је $n = pm$, за неки цео број m и при томе је $m \geq p$. Следи да је $p \leq \sqrt{n}$. ■

Горње тврђење може се формулисати и на следећи начин:

Позитиван цео број $n > 1$ је прост ако и само ако не садржи прост фактор $p \leq \sqrt{n}$.

Проблем 4.1. Дат је прост број n. Раставити га на просте чиниоце.

Проблем може да се реши индуктивно-рекурзивним приступом на следећи начин. Претпоставимо да умемо да одредимо просте чиниоце за све бројеве мање од n. Ако n није прост број, онда се он може представити као производ $n = dn_1$, где је d најмањи од свих делилаца броја n и важи $d \leq \sqrt{n}$. Чинилац d се онда може пронаћи проласком кроз скуп природних бројева у редоследу од 1 до $\sqrt{n}$ (редослед нам гарантује проналажење најмањег чиниоца), а остале чиниоце броја n_1 по индуктивној хипотези знамо да одредимо. Итеративна варијанта овог реурзивног алгоритма би се састојала из наредних корака: све док је n дељиво са 2, одштампати 2 и вредност n поставити на $\frac{n}{2}$. Када n више није дељиво са 2, онда су сви чиниоци непарни, те идемо редом од $i = 3$ до $\sqrt{n}$ по скупу непарних бројева и ако је n дељиво са i, све док је n дељиво са i, штампамо i и постављамо вредност n на $\frac{n}{i}$. Специјалан случај чини ситуација када је n прост број, у ком случају треба одштамшати сам тај број.

Алгоритам факторизације следи:

```
n = int(input())

while n%2 == 0:
    print(2)
    n = n//2

i = 3
while i*i < n:
    while n%i == 0:
```

```
            print(i)
            n = n//i
        i = i + 2

if n > 2:
    print(n)
```

Треба поменути да се нигде у алгоритму посебно не одређују прости бројеви. На први поглед ово може деловати збуњујуће, међутим редослед којим тражимо просте факторе нам гарантује да никада нећемо одштампати неки сложени број $f = i \cdot j, i, j < f$ јер ћемо пре испитивања дељивости са f проверавати дељивост са i и j, с обзиром на то да је њихова вредност мања од f.

Сложеност приказаног алгоритма је у најгорем случају $O(\sqrt{n})$ - он се дешава када је $n = p^2$, где је p неки прост број или када је сам број n прост.

У разлагању $n = p_1 \dots p_r$ се један дати прост број може појавити више пута као фактор. Због тога је уобичајено да у разлагању броја на просте чиниоце идентичне факторе „окупимо“ у степене различитих простих бројева:

$$n = {p_1}^{\alpha_1} {p_2}^{\alpha_2} \dots {p_k}^{\alpha_k},\ \alpha_i > 0.$$

Овакво разлагање броја n се зове канонички облик за $n > 1$. Из основне теореме аритметике непосредно следи да је он јединствен до на поредак степени простих бројева ${p_i}^{\alpha_i}$ (и заправо је јединствен ако, на пример, захтевамо да је $p_1 < p_2 < \dots < p_k$). Међутим, из практичних разлога ми узимамо да је $\alpha_i \geq 0$, јер онда било који прост број може формално да фигурише у каноничком представљању неког броја.

Канонички облик природног броја нам омогућава веома добру „контролу“ над његовим делиоцима, као што то наредно тврђење показује.

Теорема 4.3. Нека је $n > 1$ природан број чији је канонички облик дат са $n = {p_1}^{\alpha_1} {p_2}^{\alpha_2} \dots {p_k}^{\alpha_k}$. Тада $d \mid n$ ако и само ако је $d = {p_1}^{\beta_1} \dots {p_k}^{\beta_k}$, где је $0 \leq \beta_i \leq \alpha_i$ за све $1 \leq i \leq k$.

Доказ. ($\Rightarrow$): Ако $d \mid n$, тада је $n = dq$ за неко $q \in \mathbb{N}$; стога се канонички облик броја n добија множењем каноничких облика бројева d и q. То значи, између осталог, да је сваки прост фактор p који се појављује у каноничком облику броја d са ненула експонентом присутан и у n са ненула експонентом, и при томе се p појављује у n са најмање оноликим степеном као у d. Отуда мора бити $0 \leq \beta_i \leq \alpha_i$ за све i.

($\Leftarrow$): Ако је d облика као у формулацији тврђења, тада за

$$q = {p_1}^{\alpha_1 - \beta_1} \dots {p_k}^{\alpha_k - \beta_k} \in \mathbb{N}$$

важи $n = dq$, тј. $d \mid n$. ■

Број делилаца природног броја $n > 0$ означавамо са $\sigma_0(n)$. Приметимо да је број n прост ако и само ако је $\sigma_0(n) = 2$. Функција $\sigma_0(n)$ се веома лако израчунава на основу каноничког облика броја n.

Последица 4.1. Број делилаца броја n, израженог у каноничком облику $n = {p_1}^{\alpha_1}{p_2}^{\alpha_2} \dots {p_k}^{\alpha_k}$, једнак је

$$\sigma_0(n) = (\alpha_1 + 1)(\alpha_2 + 1) \dots (\alpha_k + 1).$$

Доказ. По претходном тврђењу, d је делилац броја n ако и само ако је $d = {p_1}^{\beta_1} \dots {p_k}^{\beta_k}$ за неке $0 \le \beta_i \le \alpha_i$, $1 \le i \le k$. Према томе, сваки низ бројева $(\beta_1, \dots, \beta_k)$ са датим ограничењима описује један делилац броја n; основна теорема аритметике обезбеђује да различити низови експонената дају различите делиоце. Број β_i се у том низу може изабрати на $\alpha_i + 1$ начина; како су сви ти избори независни, резултат следи. ■

Дефиниција 4.2. За комплексну функцију $f: \mathbb{Z} \longrightarrow \mathbb{C}$ једне целобројне променљиве (овакве функције се понекад зову и аритметичке) кажемо да је мултипликативна ако за све $a, b \in \mathbb{Z}$ такве да је $(a, b) = 1$ важи

$$f(ab) = f(a)f(b).$$

Осим у случају када је функција f константна нула-функција, имамо да је $f(1) = 1$ и $f(-1) \in \{1, -1\}$ (из тог разлога су занимљиви једино позитивни аргументи, будући да за $a > 0$ важи $f(-a) = f(-1)f(a)$).

Теорема 4.4. Функција σ_0 је мултипликативна.

Доказ. Ако су бројеви m и n узајамно прости, тада је

$$m = {p_1}^{\alpha_1}{p_2}^{\alpha_2} \dots {p_r}^{\alpha_r}, \qquad n = {q_1}^{\beta_1}{q_2}^{\beta_2} \dots {q_s}^{\beta_s},$$

при чему се ниједан од бројева p_i не поклапа си са једним од бројева q_j. Зато је

$$mn = {p_1}^{\alpha_1}{p_2}^{\alpha_2} \dots {p_r}^{\alpha_r}{q_1}^{\beta_1}{q_2}^{\beta_2} \dots {q_s}^{\beta_s}$$

канонička факторизација броја mn, па је на основу *Теореме 4.2.* и *Последице 4.1.*

$$\sigma_0(mn) = (\alpha_1 + 1) \dots (\alpha_r + 1)(\beta_1 + 1) \dots (\beta_s + 1) = \sigma_0(m)\sigma_0(n),$$

одакле следи да је σ_0 мултипликативна функција. ■

Функција $\sigma_0(n)$ је један представник фамилије $\sigma_k(n)$. Прецизније, $\sigma_k(n)$ је функција делилаца и означава суму k-тих степена свих позитивних делилаца броја n и за њу важи да

је мултипликативна. Специјално, $\sigma_0(n)$ означава број делилаца броја n, а $\sigma_1(n)$ збир свих делилаца броја n, $\sigma_2(n)$ збир квадрата делилаца броја n, ...

Највећи заједнички делилац два броја може се наћи и на следећи начин.

Теорема 4.5. Нека су природни бројеви $a, b > 0$ дати у својим „проширеним“ каноничким облицима

$$a = {p_1}^{\alpha_1} \dots {p_k}^{\alpha_k}, \qquad b = {p_1}^{\beta_1} \dots {p_k}^{\beta_k},$$

што значи да је $\alpha_i, \beta_j \geq 0$ за све $1 \leq i, j \leq k$. Тада је:

$$(a, b) = {p_1}^{min\{\alpha_1, \beta_1\}} \dots {p_k}^{min\{\alpha_k, \beta_k\}}.$$

Доказ. Нека је

$$d = \prod_{i=1}^{k} {p_i}^{min\{\alpha_i, \beta_i\}}.$$

Пошто је $min\{\alpha_i, \beta_i\} \leq \alpha_i$ и $min\{\alpha_i, \beta_i\} \leq \beta_i$, следи да $d \mid a$ и $d \mid b$, тј. d је заједнички делилац за a и b. С друге стране, нека је c заједнички делилац за a и b. По *Теореми 4.2.*, тада је

$$c = {p_1}^{\gamma_1} \dots {p_k}^{\gamma_k},$$

при чему је $\gamma_i \leq \alpha_i$ и $\gamma_i \leq \beta_i$ за све $1 \leq i \leq k$. Према томе, $\gamma_i \leq min\{\alpha_i, \beta_i\}$, одакле $c \mid d$. Стога је $d = (a, b)$. ■

Међутим, треба скренути пажњу да горње тврђење (иако делује веома једноставно и интуитивно) заправо није погодно за практичне, рачунске примене, и да са становишта теорије алгоритамске сложености даје лоше резултате. Наиме, Еуклидов алгоритам налази НЗД два броја за линеарно време у односу на број цифара и за логаритамско у односу на величину броја. С друге стране, налажење НЗД-а на основу горњег тврђења претпоставља да су дати бројеви већ факторисани на просте чиниоце, тј, дати у својим каноничким облицима. Управо ту се налази проблем: није познато да ли уопште постоји (брз) алгоритам у поменутом времену, а који разлаже дати број на просте факторе. Заправо значајан део криптографије, заштите комуникација и, специјално, добар део безбедности банкарских система заснива се на (слепој) претпоставци да такав алгоритам за факторизацију не постоји.

Из основне теореме аритметике такође следи закључак да су два природна броја узајамно проста ако и само ако немају заједнички прост делилац. То одмах даје следећи резултат.

Лема 4.1. За све природне бројеве $a, b, c > 0$ важи $(c, ab) = 1$ ако и само ако је $(c, a) = 1$ и $(c, b) = 1$.

5. Ератостеново сито

Пре него што опишемо алгоритам Ератостеновог сита, решићемо следећи проблем.

Проблем 5.1. Саставити алгоритам који за унети природан број n испитује да ли је број прост.

Природан број је прост ако је већи од 1 и ако није дељив ни са једним бројем осим са 1 и са самим собом. По дефиницији број 1 није прост. Дакле, број већи од 1 је прост ако нема ниједног правог делиоца. Потребно је извршити претрагу скупа делилаца и проверити да ли неки од њих стварно дели број n.

Следећи алгоритам проверава да ли је унети број n прост.

```
def prost(n):
    if n == 1:
        return False

    for i in range(2, n):
        if n % i == 0:
            return False
    return True

n = int(input())

if prost(n):
    print('Broj je prost!')
else:
    print('Broj nije prost!')
```

Пошто се провера сваког делиоца извршава израчунавањем једног остатка при дељењу, сложеност овог приступа одговара броју делилаца и једнака је $O(n)$.

Да би се решио проблем добијања потпуне листе простих бројева мањих од датог броја N, увешћемо једноставан алгоритам. Претпоставља се да је био откривен у древној Грчкој и познат је као Ератостеново сито (око 200.године п.н.е.).

Сада ћемо мало детаљније објаснити овај једноставан алгоритам за генерисање узастопних простих бројева који не прелазе задати цео број $N > 1$. Алгоритам започиње иницијализацијом листе главних кандидата са узастопним целим бројевима од 2 до N. Затим, приликом прве итерације, алгоритам елиминише са листе све умношке броја 2, тј. 4, 6, итд. Онда прелази на следећу ставку на листи, а то је 3 и елиминише њене умношке. У овој директној верзији један корак је занемарен, јер се неки бројеви, попут 6, елиминишу више пута. Није потребно разматрати број 4: будући да је и сама 4 и сви њени умношци такође умношци од 2, они су већ елиминисани у претходним корацима. Следећи број на листи је 5. Алгоритам се наставља на овај начин све док се више бројева не може елиминисати са листе. Преостали цели бројеви листе су тражени прости бројеви.

Као пример, размотримо примену алгоритма за проналажење листе простих бројева која не прелази $N = 25$.

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3	x	5	x	7	x	9	x	11	x	13	x	15	x	17	x	19	x	21	x	23	x	25
2	3	x	5	x	7	x	x	x	11	x	13	x	x	x	17	x	19	x	x	x	23	x	25
2	**3**	x	**5**	x	**7**	x	x	x	**11**	x	**13**	x	x	x	**17**	x	**19**	x	x	x	**23**	x	x

За овај пример више нису потребни пролази јер бисмо елиминисали бројеве који су већ елиминисани у претходним понављањима алгоритма.

Теорема 5.1. Сви бројеви који нису елиминисани представљају тачно скуп свих простих бројева од 2 до N.

Доказ. Нека је p прост број, $p \leq N$. Њега очигледно нећемо елиминисати јер су сви бројеви које елиминишемо облика qk за неке $q, k \geq 2$. Уочимо сада сложен број $n \leq N$. Нека важи да је $n = ab$, где су $a, b \geq 2$. Имамо $a^2 \leq N$ или $b^2 \leq N$. Заиста, у супротном би следило $a^2b^2 > N \cdot N = N^2$, тј. $n > N$. Контрадикција.

Без умањења општости, нека је $a^2 \leq N$. Тада a има прост чинилац p, $p^2 \leq N$. Пошто је p прост број, он неће бити елиминисан, а из $p^2 \leq N$ следи да се алгоритам неће завршити пре него што p дође на ред. У оном кораку када будемо заокружили p, број n ће бити елиминисан. ■

Следећи псеудокод узима све бројеве од 0 до n. Уместо да сложене бројеве елиминише, замениће их са 0 и на крају издвојити листу бројева који нису 0. То ће бити управо сви прости бројеви мањи од n.

```
n = int(input())

prost = list(range(0, n + 1))
prost[1] = 0
```

```
i = 2
while i*i <= n:
    if prost[i]:
        for j in range(i*i, n + 1, i):
            prost[j] = 0
    i = i + 1

rezultat = []
for x in prost:
    if x != 0:
        rezultat.append(x)

print(rezultat)
```

Сложеност Ератостеновог сита је $O(n \log\log n)$.

Скуп простих бројева $\{2, 3, 5, 7, 11, \dots\}$ је један од најједноставније задатих, али истовремено и један од „најмистериознијих“ скупова у математици уопште. Природно се поставља питање: да ли су почев од неког природног броја сви природни бројеви сложени, тј. да ли је број простих бројева коначан? Одговор на то питање дао је Еуклид. Следећа теорема, заједно са доказом, укључена је у његову књигу „Елементи“.

Теорема 5.2. Скуп простих бројева је бесконачан.

Доказ. Претпоставимо супротно: да постоји само коначно много простих бројева $p_1, \dots, p_k$. Посматрајмо број

$$A = p_1 \cdots p_k + 1.$$

Број A није дељив ни са једним од простих бројева $p_1, \dots, p_k$, јер при дељењу са сваким од њих даје остатак 1. Међутим, по основној теореми аритметике, A је производ простих бројева; специјално, A има неки прост делилац p, који је при томе различит од свих бројева $p_1 \dots p_k$. Контрадикција. ■

Нека p_n означава n-ти прост број, $n \geq 1$, при чему је $p_1 = 2, p_2 = 3, p_3 = 5$, итд.

Уобичајено је да се за (у начелу реалан број) $x > 0$ са $\pi(x)$ означи број свих простих бројева не већих од x (тј. број свих простих бројева у интервалу $[0, x]$). Фундаментални резултат аналитичке теорије бројева, теорема о простим бројевима, даје асимптотско понашање функције $\pi(x)$ када $x \to \infty$; у извесном смислу, ова теорема описује „густину“ простих бројева у скупу $\mathbb{N}$.

Теорема 5.3. *(Теорма о простим бројевима)* Важи $\pi(x) \sim \frac{x}{\ln x}$ када $x \to \infty$. Другим речима,

$$\lim_{x\to\infty} \frac{\pi(x)}{\frac{x}{\ln x}} = 1.$$

Према томе, вероватноћа да је случајно изабран број из скупа $\{1, \dots, N\}$ прост је (уз униформну расподелу и довољно велико N) приближно једнака $\frac{1}{\ln N}$.

Последица 5.1. Важи

$$\lim_{n\to\infty} \frac{p_n}{n \ln n} = 1.$$

Према томе, постоје константе $c_1, c_2 > 0$ тако да за довољно велико n важи

$$c_1 n \ln n < p_n < c_2 n \ln n.$$

6. Верижни разломци

Дефиниција 6.1. Коначним верижним разломком назива се израз облика

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots \\ a_{n-2} + \cfrac{1}{a_{n-1} + \cfrac{1}{a_n}}}}}$$

где је a_0 цео ненегативан број, а $a_1, a_2, \dots, a_n$ су природни бројеви при чему је $a_n \neq 1$.

Бројеви $a_0, a_1, a_2, \dots, a_n$ су елементи верижног разломка. Верижни разломак кратко ћемо записивати на следећи начин: $[a_0;\ a_1, a_2, \dots, a_n]$.

Иначе, назив „верижни“ долази од речи вериге, а вериге су ланац направљен од већих карика (алки). Вериге су коришћене за вешање котла за кухиње изнад огњишта.

Очигледно, сваки коначни верижни разломак је једнак неком рационалном броју.

Теорема 6.1. Сваки ненегативан рационалан број може бити представљен на јединствен начин у облику коначног верижног разломка.

Доказ. Нека је $\frac{a}{b}$ ненегативан рационалан број. Применом Еуклидовог алгоритма налазимо:

$$
\begin{aligned}
a &= q_0 b + r_1, && 0 < r_1 < b,\\
b &= q_1 r_1 + r_2, && 0 < r_2 < r_1,\\
r_1 &= q_2 r_2 + r_3, && 0 < r_3 < r_2,\\
&\dots && \dots\\
r_{k-2} &= q_{k-1} r_{k-1} + r_k, && 0 < r_k < r_{k-1},\\
r_{k-1} &= q_k r_k, && (r_{k+1} = 0).
\end{aligned}
$$

Сваку од ових једнакости запишимо на следећи начин:

$$\frac{a}{b} = q_0 + \frac{r_1}{b} = q_0 + \frac{1}{\frac{b}{r_1}},$$

$$\frac{b}{r_1} = q_1 + \frac{r_2}{r_1} = q_1 + \frac{1}{\frac{r_1}{r_2}},$$

$$\frac{r_1}{r_2} = q_2 + \frac{r_3}{r_2} = q_2 + \frac{1}{\frac{r_2}{r_3}},$$

…

$$\frac{r_{k-2}}{r_{k-1}} = q_{k-1} + \frac{r_k}{r_{k-1}} = q_{k-1} + \frac{1}{\frac{r_{k-1}}{r_k}},$$

$$\frac{r_{k-1}}{r_k} = q_k.$$

Сада једноставно налазимо

$$\frac{a}{b} = q_0 + \frac{1}{\frac{b}{r_1}} = q_0 + \frac{1}{q_1 + \frac{1}{\frac{r_1}{r_2}}} = q_0 + \frac{1}{q_1 + \frac{1}{q_2 + \frac{1}{\frac{r_2}{r_3}}}} = \cdots$$

$$= q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots \\ q_{k-2} + \cfrac{1}{q_{k-1} + \cfrac{1}{q_k}}}}},$$

односно $\frac{a}{b} = [q_0;\ q_1, q_2, \dots, q_k]$, што значи да су елементи верижног разломка количници $q_0, q_1, q_2, \dots, q_k$ који се добијају Еуклидовим алгоритмом.

Докажимо да је представљање једнозначно. Претпоставимо супротно, тј. претпоставимо да постоји бар још једно представљање рационалног броја $\frac{a}{b}$ у облику верижног разломка. Дакле, нека је

$$\frac{a}{b} = [q_0;\ q_1, q_2, \dots, q_k] \quad \text{и} \quad \frac{a}{b} = [Q_0;\ Q_1, Q_2, \dots, Q_n]$$

и при томе ова два предстваљања нису једнака. Међутим, на основу транзитивности је

$$[q_0;\ q_1, q_2, \dots, q_k] = [Q_0;\ Q_1, Q_2, \dots, Q_n].$$

Како је $\mathrm{E}\left(\frac{a}{b}\right) = q_0$ и $\mathrm{E}\left(\frac{a}{b}\right) = Q_0$, где $\mathrm{E}(x)$ означава цео број који није већи од x, то је, очигледно, $q_0 = Q_0$. Означимо са R_1 *и* R_2 верижне разломке $[q_1;\ q_2, q_3, \dots, q_k]$ и $[Q_1;\ Q_2, Q_3, \dots, Q_n]$. На исти начин закључујемо да из $E(R_1) = q_1$ *и* $E(R_2) = Q_1$ имамо да је $q_1 = Q_1$. Ако продужимо расуђивање, добићемо редом: $q_2 = Q_2$, $q_3 = Q_3$, ...

Могућа су следећа три случаја: $n = k, n < k$ *или* $n > k$.

Ако је $n = k$, онда је $q_0 = Q_0$, $q_1 = Q_1$, ... , $q_k = Q_k$ и представљања су једнака, што је у супротности са претпоставком да су представљања различита.

Ако је $n < k$, онда је : $q_0 = Q_0$, $q_1 = Q_1$, ... , $q_{n-1} = Q_{n-1}$ и

$$Q_n = q_n + \cfrac{1}{q_{n+1} + \cfrac{1}{\ddots + \cfrac{1}{q_k}}}.$$

Лева страна последње једнакости је цео број, па мора бити и десна, а то је могуће само за $k = n$ и $q_k = 1$, што је у супротности са дефиницијом верижног разломка $q_k \neq 1$.

На потпуно исти начин се разматра трећи случај: $n > k$.

Дакле, није тачна претпоставка да постоје два различита представљања рационалног броја

$\frac{a}{b}$ у облику верижног разломка. ■

Ако је рационалан број $\frac{a}{b}$ негативан, онда одговарајући верижни разломак можемо наћи на два начина. Први начин се састоји у томе да се прво разложи у верижни разломак број $-\frac{a}{b}$, па се, затим, испред добијеног верижног разломка стави знак „ −“. Тако је, на пример,

$-\frac{61}{27} = -[2; 3, 1, 6]$. Други начин се састоји у том да елемент a_0 верижног разломка $[a_0;\ a_1, a_2, \dots, a_n]$ буде негативан, а остали елементи $a_1, a_2, \dots, a_n$ позитивни. На пример,

$$-\frac{61}{27} = -3 + \frac{20}{27} = -3 + \frac{1}{1 + \frac{7}{20}} = -3 + \frac{1}{\frac{20}{7}} = -3 + \frac{1}{1 + \frac{1}{2 + \frac{6}{7}}} = -3 + \frac{1}{1 + \frac{1}{2 + \frac{1}{\frac{7}{6}}}}$$

$$= -3 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{6}}}} = [-3; 1, 2, 1, 6].$$

Алгоритам за развој реалног броја у верижни разломак:

```
import math
import decimal
x = decimal.Decimal(input())

niz_a = []
q = math.floor(x)
niz_a.append(q)
x = x - q
i = 0
while x != 0 and i < 20:
    q = math.floor(1 / x)
    niz_a.append(q)
    x = 1 / x - q
    i = i + 1

print(niz_a)
```

Нажалост, овај алгоритам није довољно прецизан (зауставиће се на 20. кораку).

Ако је $\alpha = \frac{73}{31}$, парцијални количници су 2, 2, 1, 4, 2. Ако је $\alpha = \sqrt{11}$ парцијални количници су на крају периодични: 3, 3, 6, 3, 6, 3, 6, ...

Овај поступак се завршава ако и само ако је α рационалан број.

Иако написано као алгоритам, ово је више математичка конструкција него алгоритам из више разлога:

1. за неке примере неће успети да се заврши;
2. реални бројеви не могу бити представљени у коначном смислу погодном за рачунар;

3. тестирање реалних бројева за једнакост је (алгоритамски) проблематично.

Веза између α_i и α_{i+1} може се записати као:

$$\alpha_i = a_i + \frac{1}{\alpha_{i+1}}.$$

Примењујући ово за $\alpha = \frac{73}{31}$ даје коначан верижни разломак

$$\frac{73}{31} = 2 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{2}}}} = [2; 2, 1, 4, 2],$$

где је са $[a_0; a_1, \dots, a_n]$ означена вредност верижног разломка са парцијалним количницима a_i.

7. Диофантове једначине

Диофантова једначина је једначина облика

$$f(x_1, \dots, x_k) = 0,$$

где је f полином (од k променљивих $x_1, \dots, x_k$) са целим коефицијентима, чија решења тражимо искључиво у скупу целих бројева. Једначине овог типа први је проучавао грчки математичар Диофант, па се проблем налажења њихових решења назива Диофантов проблем. Диофант је био грчки математичар који је живео у Александрији у трећем веку наше ере.

На списку од 23 проблема које је 1900. године на Међународном математичком конгресу изнео Д. Хилберт, десети проблем односио се на Диофантове једначине, а формулисан данашњим терминима гласио би: *Наћи алгоритам да по произвољној једначини овога типа буде утврђена њена решивост или нерешивост међу целим бројевима.* 1970. године, Јурij В. Матијашевич је показао да не постоји алгоритам који би за сваку Диофантову једначину одлучивао да ли она има решења; на тај начин је негативно решен 10. Хилбертов проблем. Наћи решења дате једначине која припадају скупу целих бројева, доказати или оповргнути егзистенцију таквих решења, представљају често веома тешка

питања. Међутим, управо због тога Диофантове једначине и јесу фасцинантно поље истраживања у теорији бројева.

7.1 Линеарне Диофантове једначине

Ако су a, b и c цели бројеви и $a, b \neq 0$, линеарна једначина облика

$$ax + by = c,$$

при чему су и вредности x и y из скупа целих бројева, назива се линеарна Диофантова једначина.

Захтев да су решења целобројна компликује проблем. На пример, једначина $x + y = 5$, очигледно, има бесконачно много решења, док једначина $8x + 12y = 11$ нема ниједно (за све целобројне вредности x и y лева страна једнакости је паран број и никад није једнака 11).

Следећа теорема даје потребан и довољан услов да линеарна Диофантова једначина са две променљиве има решење.

Теорема 7.1. Нека су $a, b, c \in \mathbb{Z}$ тако да је $a \neq 0$ или $b \neq 0$. Тада Диофантова једначина $ax + by = c$ има решења ако и само ако $(a, b) \mid c$.

Доказ. ($\Rightarrow$): Означимо $d = (a, b)$. Нека је (x_0, y_0) неко решење дате једначине. Пошто $d \mid a$ и $d \mid b$, важи

$$d \mid ax_0 + by_0 = c.$$

($\Leftarrow$): Претпоставимо да $d \mid c$, тј. да је $c = dc'$. Према *Теореми 3.2.*, постоје $\alpha, \beta \in \mathbb{Z}$ тако да је $d = \alpha a + \beta b$. То значи да је

$$c = a(\alpha c') + b(\beta c'),$$

односно, $x = \alpha c', y = \beta c'$ је једно решење дате једначине. ■

Очигледно је да доказ *Теореме 7.1.* и проширени Еуклидов алгоритам дају и метод за налажење једног решења Диофантове једначине са две променљиве.

Пример 7.1. Диофантова једначина $28x + 70y = 39$ нема решења, јер је $\text{НЗД}(28,70) = 14$, а 14 није делитељ броја 39.

Пример 7.2. Посматраћемо линеарну Диофантову једначину

$$13x + 32y = 5.$$

Овде је $a = 32$ и $b = 13$. Тада, користећи Еуклидов алгоритам добијамо да је

$$\begin{aligned} 32 &= 2 \cdot 13 + 6, \\ 13 &= 2 \cdot 6 + 1, \\ 6 &= 6 \cdot 1. \end{aligned}$$

Бројеви 32 и 13 су узајамно прости. По *Теореми 3.2.,* број 1 можемо представити као линеарну комбинацију бројева 32 и 13. Користећи проширени Еуклидов алгоритам налазимо тражене коефицијенте на следећи начин:

$m_0 = 1$ $\quad$ $n_0 = 0$
$m_1 = 0$ $\quad$ $n_1 = 1$
$m_2 = 1 - 2 \cdot 0 = 1$ $\quad$ $n_2 = 0 - 2 \cdot 1 = -2$
$m_3 = 0 - 2 \cdot 1 = -2$ $\quad$ $n_3 = 1 - 2 \cdot (-2) = 5$
$m_4 = 1 - 6 \cdot (-2) = 13$ $\quad$ $n_4 = -2 - 6 \cdot 5 = -32$

Дакле,

$$1 = 5 \cdot 13 + (-2) \cdot 32.$$

Коначно добијамо, множењем целог израза са 5, да је

$$13 \cdot (25) + 32 \cdot (-10) = 5.$$

Дакле, једно решење линеарне Диофантове једначине $13x + 32y = 5$ је $(25, -10)$. Лако се проверава да су решења и

$$x = 25 + 32t \quad , \quad y = -10 - 13t,$$

где је t цео број.

Теорема 7.2. Ако је $d = (a, b), d \mid c$ и (x_0, y_0) једно решење Диофантове једначине $ax + by = c$, тада су сва њена решења (x, y) дата са

$$x = x_0 + \frac{b}{d}t \text{ и } y = y_0 - \frac{a}{d}t,$$

где је $t \in \mathbb{Z}$.

Доказ. Ако је (x_0, y_0) решење Диофантове једначине $ax + by = c$, тада заменом добијамо:

$$a\left(x_0 + \frac{b}{d}t\right) + b\left(y_0 - \frac{a}{d}t\right) = ax_0 + by_0 = c.$$

Обратно, претпоставимо да (x, y) јесте једно од решења једначине $ax + by = c$. Тада добијамо:

$$\begin{aligned} ax + by &= ax_0 + by_0, \\ ax - ax_0 &= by_0 - by, \\ a(x - x_0) &= b(y_0 - y), \\ \frac{a}{d}(x - x_0) &= \frac{b}{d}(y_0 - y). \end{aligned}$$

Како је $d = (a, b)$, добијамо да је $\left(\frac{a}{d}, \frac{b}{d}\right) = 1$. Дакле,

$$\frac{b}{d} \mid (x - x_0) \text{ и } \frac{a}{d} \mid (y_0 - y),$$

одакле је

$$x - x_0 = \frac{b}{d}t \text{ и } y_0 - y = \frac{a}{d}t,$$

где је t цео број. Коначно добијамо да је

$$x = x_0 + \frac{b}{d}t \text{ и } y = y_0 - \frac{a}{d}t, \quad t \in \mathbb{Z}. \quad \blacksquare$$

Пример 7.3. Решити Диофантову једначину $69x + 111y = 9000$.

Како је $\text{НЗД}(69{,}111) = 3$, решаваћемо еквивалентну једначину $23x + 37y = 3000$. Овде је $a = 37$ и $b = 23$. Користећи Еуклидов алгоритам добијамо

$$\begin{aligned} 37 &= 1 \cdot 23 + 14, \\ 23 &= 1 \cdot 14 + 9, \\ 14 &= 1 \cdot 9 + 5, \\ 9 &= 1 \cdot 5 + 4, \\ 5 &= 1 \cdot 4 + 1, \\ 4 &= 4 \cdot 1. \end{aligned}$$

Бројеви 37 и 23 су узајамно прости. По Теореми 3.2., број 1 можемо представити као линеарну комбинацију бројева 37 и 23. Користећи проширени Еуклидов алгоритам налазимо тражене коефицијенте на следећи начин:

$m_0 = 1$
$m_1 = 0$
$m_2 = 1 - 1 \cdot 0 = 1$
$m_3 = 0 - 1 \cdot 1 = -1$
$m_4 = 1 - 1 \cdot (-1) = 2$
$m_5 = -1 - 1 \cdot 2 = -3$
$m_6 = 2 - 1 \cdot (-3) = 5$
$m_7 = -3 - 4 \cdot 5 = -23$

$n_0 = 0$
$n_1 = 1$
$n_2 = 0 - 1 \cdot 1 = -1$
$n_3 = 1 - 1 \cdot (-1) = 2$
$n_4 = -1 - 1 \cdot 2 = -3$
$n_5 = 2 - 1 \cdot (-3) = 5$
$n_6 = -3 - 1 \cdot 5 = -8$
$n_7 = 5 - 4 \cdot (-8) = 37$

Дакле,

$$(-8) \cdot 23 + (5) \cdot 37 = 1.$$

Коначно добијамо, множењем целог израза са 3000, да је

$$(-24000) \cdot 23 + (15000) \cdot 37 = 3000.$$

Решења (x, y) Диофантове једначине $69x + 111y = 9000$ су:

$$x = -24000 + 37t \quad \text{и} \quad y = 15000 - 23t, \quad t \in \mathbb{Z}.$$

Проблем решавања линеарних Диофантових једначина може се посматрати и са геометријске тачке гледишта. Тачка (x, y) равни назива се целобројна тачка ако су њене координате цели бројеви. Решавање Диофантове једначине $ax + by = c$ еквивалентно је налажењу свих целобројних тачака које задовољавају једначину праве $ax + by = c$. Нагиб те праве је рационалан број

$$-\frac{a}{b} = -\frac{\frac{a}{d}}{\frac{b}{d}},$$

где је $d = (a, b)$.

Партикуларно решење (x_0, y_0) се може лако добити применом Еуклидовог алгоритма: ако је $a, b, r_1, r_2, \ldots, r_m = (a, b)$ низ остатака који тај алгоритам генерише, тада се једначина $ax + by = c$ $(a \geq b)$ може редом свести на једначине

$$\begin{aligned} r_1 x_1 + b y_1 &= c_1, \\ r_1 x_2 + r_2 y_2 &= c_2, \\ &\vdots \end{aligned}$$

за неке целе бројеве $c_1, c_2, \ldots$, при чему је последња једначина облика $(a, b)x_m + r_{m-1} y_m = c_m$ или $r_{m-1} x_m + (a, b) y_m = c_m$, те се она може непосредно решити, имајући у виду да $(a, b) \mid r_{m-1}$ и $(a, b) \mid c_m$. Увртавањем њеног решења (општег или партикуларног) у претходне једначине, добија се одговарајуће решење за $ax + by = c$.

Илустроваћемо ово на примеру једначине

$$26x + 37y = 79.$$

Изражавајући x, следи

$$x = -y + 3 + \frac{1 - 11y}{26}.$$

Дакле, мора бити $26 \mid 1 - 11y$, што води ка решавању једначине $26u + 11y = 1$. Одавде изражавамо y, па добијамо

$$y = -2u + \frac{1 - 4u}{11}.$$

Тако је наредни корак решавање једначине $4u + 11v = 1$. Сада је

$$u = -2v + \frac{1 - 3v}{4},$$

па закључујемо да $1 - 3v$ мора бити дељиво са 4, тј. $4w + 3v = 1$. Коначно,

$$v = -w + \frac{1 - w}{3},$$

одакле је $w = -3t + 1, t \in \mathbb{Z}$. Стога $w = 1$ јесте једно партикуларно решење, које редом даје $v = -1, u = 3, y = -7$ и $x = 13$. Према томе, опште решење посматране једначине гласи

$$x = 13 + 37t, \quad y = -7 - 26t,$$

$t \in \mathbb{Z}$, пошто је $(26, 37) = 1$.

8. Конгруенције

Имајући у виду поступак дељења са остатком, природно је извршити класификацију скупа $\mathbb{Z}$ на класе бројева који дају исти остатак при дељењу са неким фиксним делитељем m. Ово се реализује бинарном релацијом конгруенције по модулу m. Наиме, дефинишемо да је

$$a \equiv b \pmod{m} \text{ ако и само ако } m \mid (a - b).$$

При томе не представља никакво ограничење општости ако претпоставимо да је $m > 0$, будући да $m \mid (a-b)$, ако и само ако $-m \mid (a-b)$. Ова еквиваленција је веома значајна, јер се помоћу ње дефинише конгруенција по модулу m и у случају кад је $m > a$ или $m > b$. На пример, $10 \equiv -3 \ (mod\ 13)$.

Теорема 8.1. За све $a, b, c, d \in \mathbb{Z}$ и $m > 0$ важи:

i. $a \equiv a \ (mod\ m)$;
ii. $a \equiv b \ (mod\ m) \ \Rightarrow \ b \equiv a \ (mod\ m)$;
iii. $a \equiv b \ (mod\ m), b \equiv c \ (mod\ m) \ \Rightarrow a \equiv c \ (mod\ m)$;
iv. $a \equiv b \ (mod\ m), c \equiv d \ (mod\ m) \ \Rightarrow a + c \equiv b + d \ (mod\ m), a - c \equiv b - d \ (mod\ m)$;
v. $a \equiv b \ (mod\ m), c \equiv d \ (mod\ m) \ \Rightarrow ac \equiv bd \ (mod\ m)$.

Доказ. Приметимо да из особина $i.-iii.$ следи да је конгруенција по модулу релација евиваленције, а из особина $iv.$ и $v.$ да се слаже са операцијама на прстену $(\mathbb{Z}, +, \cdot)$. Доказујемо само особину $v.$ Имамо да важи:

$$a \equiv b \ (mod\ m) \Leftrightarrow m \mid a - b,$$

$$c \equiv d \ (mod\ m) \Leftrightarrow m \mid c - d,$$

па m дели и сваку њихову линеарну комбинацију, тј.

$$m \mid c(a-b) + b(c-d) = ac - bc + bc - bd = ac - bd.$$

Дакле, $ac \equiv bd \ (mod\ m)$. ■

Поставља се питање да ли важи:

$$a \equiv b \ (mod\ m), c \equiv d \ (mod\ m) \Rightarrow a^c \equiv b^d \ (mod\ m),$$

и као одговор добијамо да ово не мора да важи. На пример, имамо да је $2 \equiv 6 \ (mod\ 4)$ и $5 \equiv 1 \ (mod\ 4)$. Међутим, не важи $32 \equiv 6 \ (mod\ 4)$.

Оно што директно добијамо као последицу особине $v.$ из претходне теореме јесте:

$$a \equiv b \ (mod\ m) \Rightarrow a^n \equiv b^n \ (mod\ m), \text{за све } n \geq 1.$$

Последица 8.1. Нека је $a, b \in \mathbb{Z}, m > 0$, док је $f(x)$ полином са целим коефицијентима. Тада је

$$f(a) \equiv f(b) \ (mod\ m).$$

Према томе, релација конгруенције по модулу је инваријантна на три од четири основне рачунске операције: сабирање, одузимање, множење. Међутим, то није случај са дељењем; на пример $6 \equiv 2 \ (mod\ 4)$, али то не значи да можемо конгруенцију скратити са 2, јер није

тачно да је $3 \equiv 1\ (mod\ 4)$. Скраћивање конгруенција је могуће само уз истовремену промену модула у односу на којег се конгруенција посматра; о томе говори наредни резултат.

Теорема 8.2. Нека $a, b, c \in \mathbb{Z}, m > 0$ и нека је $d = (c, m)$. Тада важи:

$$ac \equiv bc\ (mod\ m)\ \text{ако и само ако}\ a \equiv b\ \left(mod\ \frac{m}{d}\right).$$

Доказ. Конгруенција $ac \equiv bc\ (mod\ m)$ еквивалентна је услову $m \mid (a-b)c$. Он је, даље, еквивалентан са

$$\frac{m}{d} \mid (a-b)\frac{c}{d}.$$

Пошто је $\left(\frac{m}{d}, \frac{c}{d}\right) = 1$, горња дељивост важи ако и само ако $\frac{m}{d} \mid a-b$, тј. $a \equiv b\ \left(mod\ \frac{m}{d}\right)$. ■

Када су модул и број којим се скраћује узајамно прости, имамо специјалан случај у којем скраћивање конгруенције јесте коректно.

Последица 8.2. Ако су $a, b, c \in \mathbb{Z}$ и $m > 0$ такви да $ac \equiv bc\ (mod\ m)$ и $(c, m) = 1$, тада је $a \equiv b\ (mod\ m)$.

Сви цели бројеви који су конгруентни по датом модулу образују једну класу бројева. Нека је m фиксиран број. Како је конгруенција по модулу m релација еквиваленције, онда дели скуп $\mathbb{Z}$ на класе еквиваленције. У једној класи налазе се сви бројеви конгруентни с неким задатим бројем по модулу m. Сваки цео број припада тачно једној класи остатака по датом модулу m и свака класа остатака садржи тачно један од бројева $0, 1, 2, \dots, m-1$; број различитих класа је m.

Дефиниција 8.1. За $a \in \mathbb{Z}$ и фиксан модул m, класу евиваленције елемента a у односу на релацију $\cdot \equiv \cdot\ (mod\ m)$ означавамо са $(a)_m$ и зовемо је класа остатака.

Дефиниција 8.2. Сваки скуп целих бројева изабран тако да садржи тачно један број из сваке класе остатака по модулу m, назива се потпун систем остатака по модулу m.

Очигледно, $0, 1, \dots, m-1$ јесте један потпун систем остатака; њега ћемо звати стандардним.

Сада ћемо испитати како су бројеви који су узајамно прости са m распоређени у односу на класе остатака. У наредном тврђењу ће се испоставити да је тај распоред изузетно правилан: ако за неко $b \in (a)_m$ важи $(b, m) = 1$, тада су сви елементи класе $(a)_m$ узајамно прости са m.

Теорема 8.3. Нека су $a, b \in \mathbb{Z}$ *и* $m > 0$ такви да је $a \equiv b \pmod m$. Тада је $(a, m) = (b, m)$. Специјално, важи $(a, m) = 1$ ако и само ако $(b, m) = 1$.

Доказ. Претпоставка да $m \mid a - b$ значи да је $b = cm + a$ за неко $c \in \mathbb{Z}$. Одавде је очито да $(a, m) \mid b$, јер $(a, m) \mid a$ *и* $(a, m) \mid m$; стога $(a, m) \mid (b, m)$. С друге стране, изражавајући $a = b - cm$, аналогно добијамо да $(b, m) \mid (a, m)$, па следи $(a, m) = (b, m)$. ■

Дефиниција 8.3. Редукована класа остатака по модулу m је класа генерисана $(a)_m$ где важи $(a, m) = 1$.

Број елемената редуковане класе остатака по модулу m обележава се са $\varphi(m)$, где је $\varphi(m)$ Ојлерова функција.

Сада ћемо извести експлицитну формулу која даје вредности Ојлерове функције $\varphi(n)$, полазећи од претпоставке да је број $n > 1$ дат у каноничком облику

$$n = {p_1}^{\alpha_1} {p_2}^{\alpha_2} \cdots {p_k}^{\alpha_k},$$

и то у ужем смислу, дакле уз услов да је $\alpha_i > 0$ за све $1 \leq i \leq k$.

Теорема 8.4. Нека је $n > 1$ природан број дан у каноничком облику. Тада је

$$\varphi(n) = \prod_{i*1}^{k} ({p_i}^{\alpha_i} - {p_i}^{\alpha_i - 1}) = n \cdot \prod_{\substack{p \mid n \\ p \text{ прост}}} \left(1 - \frac{1}{p}\right).$$

Тврђење ове теореме ће бити директна последица следеће две леме.

Лема 8.1. За сваки прост број p и $\alpha \geq 1$ важи $\varphi(p^\alpha) = p^\alpha - p^{\alpha - 1}$.

Доказ. За природан број $m < p^\alpha$ важи $(m, p^\alpha) = 1$ ако и само ако је $(m, p) = 1$, односно, ако и само ако $p \nmid m$. Дакле, $\varphi(p^\alpha)$ је број свих елемената скупа $\{0, 1, \ldots, p^\alpha - 1\}$ који нису дељиви са p, а што је очито $p^\alpha - \left\lfloor \frac{p^\alpha}{p} \right\rfloor = p^\alpha - p^{\alpha - 1}$. ■

Лема 8.2. Ојлерова функција је мултипликативна.

Доказ. Претпоставимо да су $a, b \in \mathbb{N}$ такви да је $(a, b) = 1$. Посматраћемо најпре редукован систем остатака $r_1, r_2, \ldots, r_{\varphi(a)}$ по модулу a који је садржан у стандардном потпуном систему остатака по том модулу, дакле, у скупу $\{0, 1, \ldots, a - 1\}$. Имајући у виду *Теорему 8.5.*, тада су сви бројеви мањи од ab који су узајамно прости са a управо они који су набројани у следећој таблици:

$$\begin{array}{rrcrcr}
r_1 & r_2 & \cdots & r_i & \cdots & r_{\varphi(a)} \\
a+r_1 & a+r_2 & \cdots & a+r_i & \cdots & a+r_{\varphi(a)} \\
\vdots & \vdots & & \vdots & & \vdots \\
ka+r_1 & ka+r_2 & \cdots & ka+r_i & \cdots & ka+r_{\varphi(a)} \\
\vdots & \vdots & & \vdots & & \vdots \\
(b-1)a+r_1 & (b-1)a+r_2 & \cdots & (b-1)a+r_i & \cdots & (b-1)a+r_{\varphi(a)}
\end{array}$$

Да би неки број (заправо, остатак по модулу ab) из ове таблице био узајамно прост са ab потребно је и довољно да буде узајамно прост са b, имајући у виду *Лему 4.1*.

Посматрајмо сада произвољну колону ове таблице, на пример, i-ту колону: $r_i, a+r_i, \dots, (b-1)a+r_i$. Овај низ се састоји од b различитих бројева. Они су неконгруентни по модулу b; заиста, $ka+r_i \equiv la+r_i \ (mod\ b)$ повлачи да $b \mid (k-l)a$, тј. да $b \mid k-l$ (пошто је $(a,b)=1$). Будући да је $|k-l| < b$, следи да мора бити $k=l$. Према томе, свака колона претходне таблице чини потпун систем остатака по модулу b. Из тог разлога, свака колона садржи тачно $\varphi(b)$ бројева који су узајамно прости са b. Како тих колона има $\varphi(a)$, закључујемо да у посматраној таблици има тачно $\varphi(a)\varphi(b)$ бројева који су узајамно прости са ab. Међутим, по дефиницији Ојлерове функције, тај број је истовремено једнак $\varphi(ab)$, па жељена једнакост $\varphi(ab)=\varphi(a)\varphi(b)$ следи. ■

Леонард Ојлер је 1736. године доказао тврђење у коме функција која данас носи његово име игра централну улогу и које представља уопштење од раније познате „мале“ Фермаове теореме.

Теорема 8.5. *(Ојлерова теорема)* Нека је $a \in \mathbb{Z}$ *и* $m > 0$ тако да је $(a,m)=1$. Тада је

$$a^{\varphi(m)} \equiv 1 \ (mod\ m).$$

Доказ. Нека је $r_1, r_2, \dots, r_{\varphi(m)}$ неки редуковани систем остатака по модулу m. Тада је и $ar_1, ar_2, \dots, ar_{\varphi(m)}$ такође редукован систем остатака по модулу m. То заправо значи да је пресликавање $\pi: \{1,2,\dots,\varphi(m)\} \longrightarrow \{1,2,\dots,\varphi(m)\}$ дефинисано са $\pi(i)=j$ ако и само ако је $ar_i \equiv r_j \ (mod\ m)$ бијекција, тј. пермутација скупа $\{1,2,\dots,\varphi(m)\}$. Тако је

$$ar_1 \equiv r_{\pi(1)} \ (mod\ m),$$

$$ar_2 \equiv r_{\pi(2)} \ (mod\ m),$$

$$\vdots$$

$$ar_k \equiv r_{\pi(k)} \ (mod\ m),$$

где смо ради краћег записа означили $k=\varphi(m)$. Множећи ове конгруенције, добијамо

$$a^{\varphi(m)} r_1 r_2 \cdots r_k \equiv r_{\pi(1)} r_{\pi(2)} \cdots r_{\pi(k)} \ (mod\ m),$$

па пошто је $r_1 r_2 \cdots r_k = r_{\pi(1)} r_{\pi(2)} \cdots r_{\pi(k)}$ и $(r_1 r_2 \cdots r_k, m) = 1$, следи да је

$$a^{\varphi(m)} \equiv 1 \ (mod\ m),$$

као што се и тражило. ■

Последица 8.3. *(Мала Фермаова теорема)* Нека је p прост број и $a \in \mathbb{Z}$ такав да $p \nmid a$. Тада је

$$a^{p-1} \equiv 1 \ (mod\ p).$$

Другим речима, за све целе бројеве a важи $a^p \equiv a \ (mod\ p)$.

Доказ. Претпоставимо да имамо на располагању a различитих боја перли. Прво правимо низ од p перли, па постоји a^p различитих низова. Ако изузмемо једнобојне низове остаје нам $a^p - a$ низова. Спајамо крајеве да бисмо добили огрлицу. Низ перли који представља цикличну пермутацију неког другог низа даје идентичну огрлицу као тај низ. Постоји p цикличних пермутација p перли у низу, па је број различитих огрлица $\frac{a^p - a}{p}$. Одатле закључујемо да мора бити $p \mid a^p - a$. ■

Претходна теорема носи име по Пјеру де Фермау. Неколико теорема из теорије бројева носе његово име. Водио је опширну преписку са највећим математичарима свог времена, у којој је износио дубоку анализу и критичка мишљења о постојећим математичким теоријама. Један од проблема који је поставио (Фермаова велика теорема) остао је нерешен више од 300 година, али су покушаји његовог решавања од стране многих великих математичара имали за резултат многа значајна открића у математици.

8.1 Линеарне конгруенције

Посматрајмо линеарну конгруенцију

$$ax \equiv b \ (mod\ m),$$

где је $a, b \in \mathbb{Z}, m \in \mathbb{N}$. Њено решење је сваки цео број x_0 такав да је $ax_0 \equiv b \ (mod\ m)$. Следећа теорема даје потребан и довољан услов да број x_0 буде решење линеарне ове линеарне конгруенције.

Теорема 8.6. Број x_0 је решење линеарне конгруенције

$$ax \equiv b \ (mod\ m)$$

ако и само ако постоји цео број y_0 такав да је (x_0, y_0) решење линеарне Диофантове једначине

$$ax - my = b.$$

Доказ. Претпоставимо да је x_0 решење горње конгруенције. Тада постоји цео број y_0 такав да је

$$ax_0 - b = y_0 m.$$

Дакле, $ax_0 - my_0 = b$, што значи да је (x_0, y_0) решење Диофантове једначине

$$ax - my = b.$$

Обратно, ако је (x_0, y_0) решење Диофантове једначине $ax - my = b$, онда је $ax_0 - my_0 = b$, одакле следи да је

$$ax_0 \equiv b \ (mod\ m). \ \blacksquare$$

Следећа теорема даје критеријум решивости једначине $ax \equiv b \ (mod\ m)$.

Теорема 8.7. Једначина $ax \equiv b \ (mod\ m)$ има решења ако и само ако $(a, m) \mid b$.

Доказ. Нека је $s \in \mathbb{Z}$ тако да је $as \equiv b \ (mod\ m)$. Тада постоји $c \in \mathbb{Z}$ тако да је $as - b = mc$, односно $as + m(-c) = as - mc = b$. Према томе, Диофантова једначина $ax + my = b$ има решења, што је по *Теореми 7.1.* еквивалентно услову $(a, m) \mid b$. Обратно, ако важи овај услов, тада и једначина $ax + my = b$ има решење (x_0, y_0). Али, тада је $ax_0 \equiv b \ (mod\ m)$, па посматрана линеарна конгруенција има решење. $\blacksquare$

Питање броја и облика свих решења за $ax \equiv b \ (mod\ m)$ расправљено је у наредном тврђењу.

Теорема 8.8. Претпоставимо да једначина $ax \equiv b \ (mod\ m)$ има решења, и нека је $s \in \mathbb{Z}$ једно њено решење. Тада елементи низа

$$s, s + \frac{m}{(a, m)}, \dots, s + \big((a, m) - 1\big)\frac{m}{(a, m)}$$

чине потпун скуп неконгруентних решења по модулу m; другим речима, сви ти бројеви су решења конгруенције, међусобно су неконгруентни по модулу m, и свако решење је конгруентно неком броју тога скупа по модулу m. Према томе, линеарна конгруенција $ax \equiv b \ (mod\ m)$ има тачно (a, m) неконгруентних решења по модулу m.

Доказ. По датим условима, $as \equiv b \ (mod\ m)$. Нека је сада t било које решење једначине $ax \equiv b \ (mod\ m)$; тада је $at \equiv b \ (mod\ m)$. Отуда је $at \equiv as \ (mod\ m)$, па по *Теореми 8.2.* следи

$$t \equiv s \left(mod\ \frac{m}{(a,m)}\right).$$

Другим речима,

$$t = s + k\frac{m}{(a,m)}$$

за неко $k \in \mathbb{Z}$. Лако се проверава да се заправо за свако $k \in \mathbb{Z}$ добија по једно решење за $ax \equiv b\ (mod\ m)$, тако да наведена („двострано“ бесконачна) аритметичка прогресија представља тражени скуп решења.

Преостаје да се утврди која су од ових решења међусобно неконгруентна по модулу m. Заиста, ако је

$$t_1 = s + k_1\frac{m}{(a,m)}, \quad t_2 = s + k_2\frac{m}{(a,m)},$$

тада је $t_1 \equiv t_2\ (mod\ m)$ ако и само ако је $k_1 m' \equiv k_2 m'\ (mod\ m)$ (где је $m' = \frac{m}{(a,m)}$), а што је даље еквивалентно са $k_1 \equiv k_2\ \big(mod\ (a,m)\big)$ (поново по *Теореми 8.2.)*. Према томе, добићемо максимални скуп неконгруентних решења ако и само ако пустимо k да узима вредности у неком потпуном систему остатака по модулу (a,m). У формулацији теореме фигурише управо стандардни потпуни систем остатака по модулу (a,m), па је доказ тиме окончан. ■

Теорема 8.9. Ако је $(a,m) = 1$, тада конгруенција $ax \equiv b\ (mod\ m)$ има решење $x \equiv ba^{\varphi(m)-1}\ (mod\ m)$.

Доказ. Према Фермаовој теореми,

$$a^{\varphi(m)} \equiv 1\ \ (mod\ m),$$

одакле је

$$a^{\varphi(m)}b \equiv b\ \ (mod\ m),$$

па следи да је $x \equiv ba^{\varphi(m)-1}\ (mod\ m)$ решење конгруенције $ax \equiv b\ (mod\ m)$. ■

Наравно, преостаје питање како одредити то једно партикуларно решење s из *Теореме 8.6.* Навешћемо три приступа.

1. *Претраживање грубом силом.* Овај метод је погодан за „ручну“ примену само у случају малих модула. На пример, конгруенција $23x \equiv 11\ (mod\ 5)$ се може поједноставити до $3x \equiv 1\ (mod\ 5)$, када није тешко непосредно утврдити да је са $x \equiv 2\ (mod\ 5)$ дато јединствено решење посматране конгруенције.

2. *Диофантове једначине.* Раније смо већ видели да је $ax \equiv b \ (mod\ m)$ еквивалентно линеарној Диофантовој једначини $ax + my = b$ у смислу да за свако x које решава линеарну конгруенцију постоји $y \in \mathbb{Z}$ тако да је (x, y) решење одговарајуће Диофантове једначине; обратно, прва компонента сваког решења (x, y) потоње једначине решава и линеарну конгруенцију. У начелу, (целобројна) решења једначине $ax + my = b$ (где $(a, m) \mid b$) добијају се на основу Еуклидовог алгоритма за израчунавање (a, m).
3. *Ојлерова теорема.* У начелу, тражење партикуларног решења конгруенције $ax \equiv b \ (mod\ m)$ може се свести на случај када је $(a, m) = 1$; у супротном, ако је $a = (a, m)a_1, b = (a, m)b_1$ и $m = (a, m)m_1$, тада је по *Теореми 8.2.* свако решење једначине $a_1 x \equiv b_1 \ (mod\ m_1)$ уједно и решење за полазну конгруенцију. Приметимо да је тада $(a_1, m_1) = 1$; због тога је, по Ојлеровој теореми ${a_1}^{\varphi(m_1)} \equiv 1 \ (mod\ m_1)$, па $s = {a_1}^{\varphi(m_1)-1} b_1$ представља једно тражено решење, с обзиром на $a_1 \cdot {a_1}^{\varphi(m_1)-1} b_1 = {a_1}^{\varphi(m_1)} b_1 \equiv b_1 \ (mod\ m_1)$.

8.1.1 Модуларни инверз

Модуларни инверз броја a по модулу m је број x за који важи $ax \equiv 1 \ \ (mod\ m)$. Број x можемо означити и са a^{-1}.

Поставља се питање да ли модуларни инверз увек постоји. Показује се да он постоји ако и само ако су бројеви a и m узајамно прости и тада је јединствен.

Теорема 8.10. За дате целе бројеве a и m постоји цео број a' такав да је $aa' \equiv 1 \ \ (mod\ m)$ ако и само ако $(a, m) = 1$.

Доказ. Претпоставимо прво да важи $aa' \equiv 1 \ \ (mod\ m)$ за неки цео број a'. Одавде следи $aa' = 1 + my$ за одговарајуће a' и y, што значи да су a и m узајамно прости.

Обратно, ако је $(a, m) = 1$ онда према *Теореми 3.3.*, $1 = ax + my$ за погодне x и y одакле следи $ax \equiv 1 \ \ (mod\ m)$. ■

Наивна метода налажења модуларног инверза броја a по модулу m састоји се у следећем:

1. Израчунати $ax \ \ mod\ m$ за све вредности $x \in \{0, \dots, m-1\}$.
2. Модуларни инверз за a по модулу m је вредност x за коју је $ax \ \ mod\ m = 1$.

Илуструјмо овај поступак на следећем примеру.

Пример 8.1. Нека је $a = 3$ и $m = 7$. Израчунати модулрани инверз броја a по модулу m.

1. Прво ћемо одредити вредности ax по модулу m. У овом случају x ће узимати вредности из скупа $\{0,1,2,3,4,5,6\}$.
   $3 \cdot 0 \equiv 0 \pmod 7$
   $3 \cdot 1 \equiv 3 \pmod 7$
   $3 \cdot 2 \equiv 6 \pmod 7$
   $3 \cdot 3 \equiv 9 \equiv 2 \pmod 7$
   $3 \cdot 4 \equiv 12 \equiv 5 \pmod 7$
   $3 \cdot 5 \equiv 15 \equiv 1 \pmod 7$
   $3 \cdot 6 \equiv 18 \equiv 4 \pmod 7$
2. Закључујемо да је 5 модуларни инверз за број 3 $\bmod 7$ јер је $3 \cdot 5 \bmod 7 = 1$.

Да бисмо добили ефикаснији алгоритам који ради у случају када су бројеви a и m узајамно прости, можемо користити проширени Еуклидов алгоритам. Проширени Еуклидов алгоритам одређује највећи заједнички делилац d и бројеве s и t такве да за дате бројеве a и b важи $as + bt = d$. Да бисмо пронашли модуларни инверз, у претходној једначини уместо b уврстићемо m. Према претпоставци a и m су узајамно прости, па је $d = 1$. Дакле, важи једнакост

$$as + mt = 1,$$

односно, важи и

$$as + mt \equiv 1 \pmod m.$$

Други сабирак на левој страни једначине је дељив са m па га можемо уклонити, те добијамо:

$$ax \equiv 1 \pmod m.$$

Посматрајмо следећу табелу:

$r_0 = a$	$s_0 = 1$	$t_0 = 0$
$r_1 = m$	$s_1 = 0$	$t_1 = 1$
$r_2 = r_0 - q_1 r_1$	$s_2 = s_0 - q_1 s_1$	$t_2 = t_0 - q_1 t_1$
…	…	…
$r_{i+1} = r_{i-1} - q_i r_i$	$s_{i+1} = s_{i-1} - q_i s_i$	$t_{i+1} = t_{i-1} - q_i t_i$
…	…	…
$r_k = r_{k-2} - q_{k-1} r_{k-1}$	$s_k = s_{k-2} - q_{k-1} s_{k-1}$	$t_k = t_{k-2} - q_{k-1} t_{k-1}$
$r_{k+1} = 0$	$s_{k+1} = s_{k-1} - q_k s_k$	$t_{k+1} = t_{k-1} - q_k t_k$

Када пронађемо коефицијенте s и t користећи проширени Еуклидов алгоритам, коефицијент t је тражени модуларни инверз. Ако је тај број негативан, онда се за инверз може узети $t + m$.

Алгоритам за израчунавање модуларног инверза помоћу проширеног Еуклидовог алгоритма дат је у наставку.

```
def bezu(a, b):
    r0 = a
    r1 = b

    s0 = 1
    s1 = 0

    t0 = 0
    t1 = 1

    while r1 > 0:
        q = r0//r1

        r = r0
        r0 = r1
        r1 = r - q*r1

        s = s0
        s0 = s1
        s1 = s - q * s1

        t = t0
        t0 = t1
        t1 = t - q * t1

    return s0, t0, r0

a = int(input())
m = int(input())

(s, t, r) = bezu(m, a)

if t < 0:
    t = t + m

print(t)
```

Пример 8.2. Одредити модуларни инверз за број 41 по модулу 7.

Прво ћемо применити основни Еуклидов алгоритам за налажење највећег заједничког делиоца:

$$41 = 5 \cdot 7 + 6,$$
$$7 = 1 \cdot 6 + 1,$$
$$6 = 6 \cdot 1.$$

Пошто је $(41,7) = 1$, модуларни инверз постоји. Дакле, $q_1 = 5, q_2 = 1, q_3 = 6$.

$r_0 = 41$	$s_0 = 1$	$t_0 = 0$
$r_1 = 7$	$s_1 = 0$	$t_1 = 1$
$r_2 = 41 - 5 \cdot 7 = 6$	$s_2 = 1 - 5 \cdot 0 = 1$	$t_2 = 0 - 5 \cdot 1 = -5$
$r_3 = 7 - 1 \cdot 6 = 1$	$s_3 = 0 - 1 \cdot 1 = -1$	$t_3 = 1 - 1 \cdot (-5) = 6$
$r_4 = 6 - 6 \cdot 1 = 0$	$s_4 = 1 - 6 \cdot (-1) = 7$	$t_4 = -5 - 6 \cdot 6 = -41$

На основу претходне табеле уочавамо да 1 можемо представити као линеарну комбинацију 41 и 7 као $1 = 41 \cdot (-1) + 7 \cdot 6$. Закључујемо да је модуларни инверз за број 41 по модулу 7 једнак броју 6.

Још један начин за израчунавање модуларног инверза је коришћење Ојлерове теореме (*Теорема 8.5.*). Ојлерова теорема тврди да ако су бројеви a и m узајамно прости важи следеће:

$$a^{\varphi(m)} \equiv 1 \ (mod\ m).$$

Множењем обе стране са a^{-1} добијемо:

$$a^{\varphi(m)-1} \equiv a^{-1} \ (mod\ m).$$

Алгоритам за израчунавање модуларног инверза заснован на Ојлеровој теореми дат је у наставку.

```
def phi(m):
    d = 2
    proizvod = m
    while d*d <= m:
        if m%d == 0:
            proizvod = (proizvod/d) * (d - 1)

            while n%d == 0:
                m = m//d
        d = d+1
    if m > 1:
        proizvod = (proizvod/m) * (m - 1)

    return proizvod

def stepen_po_modulu(x, k, m):
    k = k % m
    stepen = 1
    while k > 0:
        if k % 2 == 1:
            stepen = (stepen * x) % m
        x = (x*x) % m
        k = k//2
    return stepen

a = int(input())
b = int(input())

inverz = stepen_po_modulu(a, phi(m) - 1, m)
```

```
print(inverz)
```

Иако је ова метода лакша за разумевање него претходна, у случају када m није прост број, морамо израчунати Ојлерову функцију која укључује факторизацију броја m, што може бити изузетно тешко.

8.1.2 Кинеска теорема о остацима

Посматрајмо сада систем линеарних конгруенција

$$\begin{aligned} a_1 x &\equiv b_1 \ (mod\ m_1) \\ a_2 x &\equiv b_2 \ (mod\ m_2) \\ &\vdots \\ a_n x &\equiv b_n \ (mod\ m_n) \end{aligned}$$

Под решењем тог система подразумеваћемо цео број x_0 који је решење сваке конгруенције система. Свака конгруенција одређује једну класу остатака. Зато су решења система конгруенција бројеви у пресецима тих класа. Свака од горњих конгруенција може се заменити другом која одређује исту класу остатака.

Следећа теорема и њене последице дају потребне и довољне услове за постојање решења система линеарних конгруенција и одређују облик општег решења.

Теорема 8.11. Систем линеарних конгруенција

$$\begin{aligned} x &\equiv a \ (mod\ m_1) \\ x &\equiv b \ (mod\ m_2) \end{aligned}$$

има решење ако и само ако је $(m_1, m_2) \mid a - b$. Ако је, при томе, x_0 једно решење, онда је опште решење тог система

$$x \equiv x_0 \ (mod\ [m_1, m_2]).$$

Доказ. Цео број x_0 је решење горњег система конгруенција ако и само ако постоји цео број k, такав да је

$$x_0 = a + km_1$$

и

$$a + km_1 \equiv b \ (mod\ m_2).$$

Према *Теореми 7.1.,* такав број постоји ако и само ако је

$$(m_1, m_2) \mid a - b.$$

Претпоставимо сада да је $(m_1, m_2) \mid a - b$ и да је x_0 једно решење система конгруенција. За било које друго решење x_1 тог система је

$$x_1 \equiv a \equiv x_0 \ (mod\ m_1)$$

и

$$x_1 \equiv b \equiv x_0 \ (mod\ m_1).$$

Дакле, $x_1 - x_0$ је заједнички садржалац од m_1 и m_2; према томе је

$$[m_1, m_2] \mid x_1 - x_0,$$

одакле следи

$$x_1 \equiv x_0 \ (mod\ [m_1, m_2]).$$

Обратно, ако је $x_1 \equiv x_0 \ (mod\ [m_1, m_2])$, онда је, очигледно

$$x_1 \equiv x_0 \equiv a \ (mod\ m_1)$$

и

$$x_1 \equiv x_0 \equiv b \ (mod\ m_2),$$

па је x_1, такође, решење система. Дакле, опште решење је

$$x \equiv x_0 \ (mod\ [m_1, m_2]). \ \blacksquare$$

Последица 8.4. Систем линеарних конгруенција

$$\begin{aligned} x &\equiv a_1 \ (mod\ m_1) \\ x &\equiv a_2 \ (mod\ m_2) \\ &\vdots \\ x &\equiv a_n \ (mod\ m_n) \end{aligned}$$

има решење ако и само ако је $[m_i, m_j] \mid a_i - a_j$, за $1 \leq i < j \leq n$. У том случају, ако је x_0 једно решење, онда је опште решење система

$$x \equiv x_0 \ (mod\ [m_1, m_2, \dots, m_n]).$$

Последица 8.5. Ако је $(m_1, m_2) = 1$, тада систем

$$\begin{aligned} x &\equiv a \ (mod\, m_1) \\ x &\equiv b \ (mod\, m_2) \end{aligned}$$

има решење за свако $a, b \in \mathbb{Z}$, и при томе скуп свих решења чини једну класу остатка по модулу $m_1 m_2$.

У специјалном случају кад су модули $m_1, m_2, \dots, m_n$ по паровима узајамно прости, систем очигледно има решење. Та чињеница била је позната кинеском математичару Сун-Тсе, у првом веку нове ере. У питању је кинеска теорема о остацима, вероватно најстарија записана математичка теорема, која у потпуности даје одговор на питање о решивости система конгруенцијских једначина када су модули у том систему узајамно по паровима прости.

Теорема 8.12. *(Кинеска теорема о остацима)* Нека су $m_1, m_2, \dots, m_n$ по паровима узајамно прости позитивни цели бројеви. Тада систем линеарних конгруенција

$$\begin{aligned} x &\equiv a_1 \ (mod\, m_1) \\ x &\equiv a_2 \ (mod\, m_2) \\ &\vdots \\ x &\equiv a_n \ (mod\, m_n) \end{aligned}$$

има тачно једно решење по модулу $M = m_1 m_2 \cdots m_n$.

Доказ. Нека је $M_i = \frac{M}{m_i}$, за $i = 1,2, \dots, n$. Посматрајмо n конгруенција

$$M_i x \equiv 1 \ \ (mod\ m_i).$$

Како је НЗД$(M_i, m_i) = 1$, свака од тих конгруенција има јединствено решење $x \equiv x_i \ \ (mod\ m_i)$. Нека је

$$X \equiv M_1 x_1 a_1 + M_2 x_2 a_2 + \cdots + M_n x_n a_n \ \ (mod\ M).$$

Уврштавањем X уместо x у конгруенцију $x \equiv a_1 \ \ (mod\, m_1)$, закључујемо да је X, као и сваки број облика $X + kM$, решење те конгруенције. На исти начин следи да то важи и за све остале конгруенције полазног система. Међутим, $X + kM$ је и једино решење система, јер ако је X_1, такође, решење тог система, онда уврштавањем X и X_1 у те конгруенције, добијамо да је

$$X \equiv X_1 \ \ (mod\, m_i), \quad i = 1,2, \dots, n,$$

па је

$$X \equiv X_1 \pmod{M},$$

јер су бројеви m_i по паровима узајамно прости. ■

Из доказа кинеске теореме о остацима, види се јасно и алгоритам за решавање одговарајућег система конгруенција.

```
def inverz(a ,n):
    r0 = n
    r1 = a

    t0 = 0
    t1 = 1

    while r1 > 0:
        q = r0//r1

        r = r0
        r0 = r1
        r1 = r - q*r1

        t = t0
        t0 = t1
        t1 = t - q * t1

    if t0 < 0:
        t0 = t0 + n

    return t0


a = [int(i) for i in input().split(',')]
n = [int(i) for i in input().split(',')] #a i n su nizovi, brojevi se unose
odvojeni zarezom

N = 1
for x in n:
    N = N*x

rezultat = 0
for i in range(len(n)):
    pi = N//n[i]
    pi_inverz = inverz(pi, n[i])
    rezultat = (rezultat + (a[i]*pi_inverz*pi) % N) % N

print(rezultat)
```

Пример 8.3. Нађи све целе бројеве који дају остатке 2, 6, 5, при дељењу са 5, 7 и 11, редом.

Треба, уствари, решити систем конгруенција

$$x \equiv 2 \ (mod\ 5)$$
$$x \equiv 6 \ (mod\ 7)$$
$$x \equiv 5 \ (mod\ 11)$$

Овде је $m_1 = 5, m_2 = 7, m_3 = 11, \mathrm{M} = 385, \mathrm{M}_1 = 77, \mathrm{M}_2 = 55, \mathrm{M}_3 = 35$, па формирамо конгруенције:

$$77x \equiv 1 \ (mod\ 5)$$
$$55x \equiv 1 \ (mod\ 7)$$
$$35x \equiv 1 \ (mod\ 11),$$

које су еквивалентне, редом, конгруенцијама

$$2x \equiv 1 \ (mod\ 5)$$
$$6x \equiv 1 \ (mod\ 7)$$
$$2x \equiv 1 \ (mod\ 11),$$

чија су решења, редом:

$$x \equiv 3 \ (mod\ 5),$$
$$x \equiv 6 \ (mod\ 7),$$
$$x \equiv 6 \ (mod\ 11).$$

Дакле,

$$X \equiv 77 \cdot 3 \cdot 2 + 55 \cdot 6 \cdot 6 + 35 \cdot 6 \cdot 5 \ (mod\ 385),$$

тј.

$$X \equiv 27 \ (mod\ 385).$$

Закључак

Појавом дигиталних рачунара и дигиталних комуникација, дошло је до спознаје да теорија бројева пружа много одговора и решења у стварним проблемима. Истовремено развојем рачунарске технологије омогућава се решавање многих проблема из теорије бројева, као што су факторизација великих природних бројева, генерисање простих бројева и проверавање простости бројева, те проверавање неких постављених хипотеза.

Алгоритми, као алати за решавање проблема, предмет су проучавања од најранијих времена. О томе најбоље сведочи Еуклидов алгоритам који је окарактерисан као најстарији нетривијални алгоритам (око 300.године п.н.е.). На основу онога што је изложено у раду, увиђамо да Еуклидов алгоритам има велику теоријску и практичну примену. Практичне примене су следеће: решавање линеарних Диофантових једначина, решавање линеарних конгруенција, одређивање модуларног инверза, развој рационалног броја у верижни разломак. Нагли развој рачунарства у 20. веку дао је нови значај проучавању алгоритама. Процес из реалног света моделује се алгоритмом, при чему су улазне и излазне величине процеса уједно и улазне и излазне величине алгоритма. Алгоритам за неки проблем је прецизно наведен низ инструкција које рачунар треба да изврши ради решавања тог проблема, односно ради трансформисања сваке инстанце проблема у њено решење. Наравно, један проблем може имати више алгоритама различите ефикасности који га решавају. Налажење ефикасног алгоритма представља централну тему којом се баве рачунарске науке. Псеудокодови наведени у раду садрже елементе програмског језика Пајтон и уз мало корекција могу се превести и користити у било ком програмском језику.

Модуларна аритметика заједно са неколико основних теорема и поступака као што су Еуклидов алгоритам, Ојлерова теорема, Мала Фермаова теорема, омогућава да се представе главне идеје савремене криптографије.

Литература

[1] Bach E. / Shallit J., Algorithmic Number Theory, Volume 1: Efficient Algorithms, Cambridge: The MIT Press, 1997.

[2] Buhler J. / Wagon S., *Basic algorithms in number theory,* MSRI Publications, 2008.

[3] Dujella A., *Diskretna matematika: Matematičke osnove kriptografije javnog ključa (bilješke s predavanja).*

[4] Edwards H. M., Higher Arithmetic: An Algorithmic Introduction to Number Theory, Providence, Rhode Island: American Mathematical Society, 2008.

[5] http://poincare.matf.bg.ac.rs/~vesnap/kaa/07_kaa-algebarski-algoritmi.pdf.

[6] https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-inverses.

[7] Knuth D. E., The Art of Computer Programming: Volume 2 Seminumerical Algorithms, Addison-Wesley Longman, 1998.

[8] Levitin A., Introduction to the Design and Analysis of Algorithms, Pearson Education, 2012.

[9] Rosen K. H., Elementary Number Theory and Its Applications, Reading: Addison-Wesley Publishing Company, 1986.

[10] Yan S. Y., Number Theory for Computing, Birmingham: Springer, 2001.

[11] Долинка И., *Теорија бројева - предавања,* Нови Сад: Природно-математички факултет, 2014.

[12] Живковић Д., Увод у алгоритме и структуре података, Београд: Универзитет Сингидунум, 2010..

[13] Живковић М., Алгоритми, Београд: Математички факултет, 2000.

[14] Золић А., *Верижни разломци и примјене.*

[15] Кечкић Ј. Д., Математика са збирком задатака за трећи разред средње школе, Београд: Завод за уџбенике и наставна средства, 2006.

[16] Ковачевић М. А., Основе програмирања у Пајтону, Београд: Академска мисао, 2017..

[17] Марић Ф.: http://poincare.matf.bg.ac.rs/~filip//zbirka3/toc.html.

[18] Мићић В. / Каделбург З., Увод у теорију бројева, Београд: Друштво математичара Србије, 1989.

[19] Огризовић П., Математика 5: уџбеник за пети разред основне школе, Београд: Нови Логос, 2018.

[20] Радан М., Рачунарство и информатика, Српско Сарајево: Завод за уџбенике и наставна средства, 2004.

[21] Станић М. / Икодиновић Н., Теорија бројева - збирка задатака, Београд: Завод за уџбенике и наставна средства, 2004.

[22] Стојсављевић-Радовановић М. / Вуковић Љ. / Ранчић Ј., Математика: уџбеник за пети разред основне школе, Београд: Креативни центар, 2019.

[23] Тасић М., „Од интуитивног до прецизираног појма "алгоритам",“ *Математика: стручно-методички часопис,* pp. 15-22, 1978.

[24] Тошић Д., Рачунарство и информатика, Београд: Завод за уџбенике и наставна средства, 2006.

[25] Тошић Р. / Вукославчевић В., Елементи теорије бројева, Нови Сад: Алеф, 1995.