RAZISKOVALNA NALOGA

Uporaba kongruenc v kriptologiji

Avtor: **Jana Avšič**
Mentor: prof. Karmen Kete
Somentor: prof. Jakob Avšič
Šolski center Nova Gorica, Cankarjeva 10, Nova Gorica

Področje: SŠ matematika

Nova Gorica, marec 2024

POVZETEK

V raziskovalni nalogi želimo pokazati uporabo modularne aritmetike v nekaterih metodah šifriranja ter prikazati delovanje teh metod s pomočjo programskega jezika Python. V začetnem delu opišemo matematična orodja, ki jih potrebujemo v kriptologiji, med njimi Eulerjev izrek, reševanje linearne kongruence z eno in z dvema neznankama, reševanje sistema dveh linearnih kongruenc z dvema neznankama, računanje potenc z velikimi eksponenti po danem modulu ter računanje inverza proti modulu tujega števila s pomočjo Eulerjevega izreka in s pomočjo razširjenega Evklidovega algoritma.

V nadaljevanju se ukvarjamo z dvema metodama šifriranja: šifriranje s pomočjo afine transformacije ter RSA šifriranje. Spotoma vpeljemo matematične pripomočke, ki jih potrebujemo pri posameznem tipu šifriranja ter izdelamo računalniške programe, s katerimi šifriranje prikažemo na konkretnih primerih. Programi temeljnih algoritmov iz naloge so navedeni znotraj teksta naloge, ostali, ki služijo lepšemu izpisu, prikazom,…pa so zapisani v datotekah, ki se nahajajo v mapi Programi, ki je priložena nalogi. V primeru afine transformacije se ukvarjamo tudi s kriptoanalizo in pokažemo, da je ta metoda glede na današnjo moč računalniške tehnologije za uporabo preveč ranljiva.

In this research project, we aim to demonstrate the application of modular arithmetic in certain encryption methods and to illustrate the operation of these methods using the Python programming language. In the initial section, we describe the mathematical tools needed in cryptography, including Euler's theorem, solving linear congruences with one and two unknowns, solving a system of two linear congruences with two unknowns, calculating powers with large exponents under a given modulus, and computing the modular inverse of a coprime number using Euler's theorem and the extended Euclidean algorithm.
Subsequently, we deal with two encryption methods: encryption using affine transformation, and RSA encryption. Along the way, we introduce mathematical tools needed for each type of encryption and create computer programs to demonstrate the encryption on a specific example. In the case of affine transformation, we also deal with cryptanalysis and show that this method, given the current strength of computer technology, is too vulnerable for use.

KAZALO VSEBINE

UVOD

Med najbolj osnovne lastnosti človeških skupnosti sodi sporazumevanje oziroma izmenjava sporočil.
Ljudje se med seboj sporazumevamo z besednim in nebesednim sporazumevanjem. Pri besednem sporazumevanju je sporočilo podano z besedami, ki so izgovorjene ali zapisane, pri nebesednem pa kako drugače, na primer s kretnjami, z izrazom na obrazu, z držo telesa, s slikami, risbami ali zemljevidi.
Osebi, ki sporočilo sporoča, rečemo $sporočevalec$, osebi, ki ji je sporočilo namenjeno pa $naslovnik$ oziroma $prejemnik$. V skladu s tem lahko sporazumevanje opišemo kot izmenjevanje sporočil med sporočevalcem in naslovnikom. Bistvo vsakega sporočila je njegova $vsebina$, ki pomeni tisto, kar sporočilo vsebuje oziroma, kar želimo sporočiti. Različna sporočila imajo lahko isto vsebino (npr. vremenska napoved v različnih jezikih). Sporočilo mora biti izdelano tako, da ga njegov naslovnik lahko razume, kar pomeni, da lahko iz njega izlušči njegovo vsebino. Že v antiki (in še prej) se je pri sporazumevanju v diplomaciji, vojaških zadevah in trgovini pojavljala potreba, da bi bil naslovnik edina oseba, ki bi lahko razumela dano sporočilo. Ohranile so se metode za doseganje tajnosti sporočil, ki so jih pred več kot dva tisoč leti uporabljali v Rimskem cesarstvu. Razvitih je bilo ogromno načinov za zavarovanje vsebine sporočil. Prav tako starodavni so tudi poskusi razvozlavanja prestreženih in prepisanih sporočil s strani nasprotnikov. Ti so bili še posebej aktualni v vojnih časih, vendar zgodovinski viri obenem poročajo, da so bili skoraj običajna praksa na dvorih pri prenosu diplomatske pošte.
V raziskovalni nalogi bomo opisali nekatere klasične metode šifriranja sporočil, ki temeljijo na modularni aritmetiki. Prve med njimi izvirajo iz časov Julija Cezarja, novejši tajni sistemi pa so bili razviti šele v sedemdesetih letih dvajsetega stoletja. Vsem tem klasičnim tajnim sistemom je skupno, da si morata dve osebi, ki želita zasebno komunicirati, deliti tako imenovani skrivni ključ.

1. OSNOVNI POJMI KRIPTOLOGIJE

Kadar je vsebina sporočila dostopna le njegovemu naslovniku, rečemo, da je to sporočilo $skrivno$ oziroma $tajno$. Za osebi, ki si izmenjujeta tajna sporočila, rečemo, da $zasebno\ komunicirata$. V sodobnem svetu je s pojavom svetovnega spleta, elektronskega sporočanja in poslovanja tajno sporočanje postalo še bolj pomembno. Vejo matematike, ki proučuje pojave, vezane na tajno sporočanje imenujemo $kriptologija$. Beseda $kriptologija$ izvira iz grške besede $kryptós$, ki pomeni $skrit$ ter iz besede $logos$, ki pomeni $beseda$. Dobesedno bi jo lahko opisali kot $vedo\ o\ skritih\ besedah$, vendar uporabljamo širšo definicijo in ji rečemo $veda\ o\ tajnih\ sistemih$. V kriptologiji uporabljamo naslednje osnovno besedišče: Sporočilo, ki ga želimo preoblikovati v tajno obliko, imenujemo $čistopis$. Postopku, s katerim čistopis preoblikujemo v tajno obliko, rečemo $enkripcija$ oziroma $šifriranje$ (na kratko $šifra$), tajno sporočilo, ki je rezultat enkripcije pa poimenujemo s katero od sopomenk: $tajnopis$, $kriptogram$, $šifropis$ oziroma $šifrirano\ besedilo$. Naslovnik sporočila, ki pozna izbrani postopek šifriranja, dostopa do vsebine sporočila tako, da tajnopis preoblikuje v čistopis z obratnim postopkom, ki ga imenujemo $dekripcija$ ali $dešifriranje$. V matematičnem smislu je enkripcija bijektivna

funkcija, ki numeričnim ekvivalentom črk abecede (oziroma blokom, ki jih iz njih sestavimo) priredi cela števila, dekripcija pa njena inverzna funkcija, ki tem številom nazaj priredi ustrezne numerične ekvivalente črk. Rečemo jima tudi enkripcijska in dekripcijska $transformacija$.
Temeljna naloga kriptologije je torej z ustrezno enkripcijo zavarovati vsebino sporočila v tajnopisu, iz katerega jo bo s samo njemu znano dekripcijo izluščil edino le naslovnik tega sporočila. Iz različnih, bodisi pozitivnih bodisi negativnih razlogov, želi vsebino tajnopisa prestreči oseba ali skupina, ki ni predvideni prejemnik tega sporočila. Postopek, ki ga ta oseba uporabi, da naredi sporočilo razumljivo, imenujemo $razbijanje$ šifre. Beseda razbijanje kaže bolj na etično negativni vidik tega dejanja, vendar je razbijanje šifre s stališča etike lahko tudi pozitivno, npr. prizadevanje odporniškega gibanja okupiranega naroda za razbitje šifre, s pomočjo katero komunicira okupatorjeva vojska. Znana je električna naprava $Enígma$ (grško uganka) za šifriranje sporočil, ki so jo med 2. svetovno vojno uporabljale nemške oborožene sile. Strateška prednost, ki so jo z razbitjem šifer Enigme dosegli zavezniki, je rešila ogromno človeških življenj in po sicer ne enotnem mnenju zgodovinarjev skrajšala vojno do enega leta.
Sporočevalec in naslovnik tajnopisa se torej soočata s tveganjem razbitja njune šifre in s tem z morda usodnim tveganjem ugrabitve vsebine sporočila. To tveganje ju sili v raziskovanje, ki je podobno tistemu od njunih nasprotnikov, namreč v raziskovanje odpornosti njune enkripcije proti njenemu razbitju. Veda kriptologija se zato razdeli na dve veji, in sicer na

- $kriptografijo$, ki se ukvarja z načrtovanjem in izvajanjem metod preoblikovanja sporočil v tajna sporočila in
- $kriptoanalizo$, ki se ukvarja z raziskovanjem odpornosti metod enkripcije proti razbitju.

2. EULERJEV IZREK

$Eulerjeva\ funkcija$ je funkcija, ki preslikuje iz množice naravnih števil v množico naravnih števil: $\varphi: \mathbb{N} \longrightarrow \mathbb{N}$. Vrednost φ pri nekem $n \in \mathbb{N}$ je definirana kot število proti njemu tujih manjših števil. Npr. $\varphi(1) = 0,\ \varphi(2) = 1,\ \varphi(5) = 4,\ \varphi(8) = 4, \ldots$
Vrednost Eulerjeve funkcije pri izbranem modulu je enaka številu proti temu modulu tujih ostankov.
Predpis Eulerjeve funkcije ugotovimo v več korakih. Najprej zapišimo njen predpis za praštevila, nato za praštevilske potence, nato za produkt tujih si števil in končno za poljubno naravno število.

Trditev 2.1. Vrednost Eulerjeve funkcije pri praštevilu je njegov predhodnik: $\varphi(p) = p - 1$.

Dokaz. Naj bo p poljubno praštevilo. Število p je tuje vsem naravnim številom, ki so od njega manjša, deliti jih namreč ne more, saj so od njega manjša. Teh naravnih števil je ravno $p - 1 \Longrightarrow \varphi(p) = p - 1$.

Konec dokaza.

Trditev 2.2. Vrednost Eulerjeve funkcije pri praštevilski potenci je

$$\varphi(p^n) = p^n - p^{n-1} = p^n\left(1 - \frac{1}{p}\right).$$

Dokaz. Naj bo p^n poljubna praštevilska potenca ($n \in \mathbb{N}$). Označimo množico vseh naravnih števil od 1 do p^n s črko G. Množica G ima p^n elementov. Števila iz G, ki niso tuja p^n, so natanko večkratniki praštevila p. To so števila: $1p, 2p, 3p, \ldots, \underbrace{p^n - p}_{(p^{n-1}-1)\cdot p}, \underbrace{p^n}_{p^{n-1}\cdot p}$. Teh števil je p^{n-1}. Preostala števila iz G so proti p^n tuja in manjša od p^n, torej je njihovo število enako vrednosti Eulerjeve funkcije pri p^n: $\varphi(p^n) = p^n - p^{n-1}$. Konec dokaza.

Trditev 2.3. Eulerjeva funkcija od produkta tujih si števil je enaka produktu svojih vrednosti pri faktorjih tega produkta: $d(b,c) = 1 \Longrightarrow \varphi(b \cdot c) = \varphi(b) \cdot \varphi(c)$.

Dokaz. Naj bosta b in c poljubni tuji si števili. Zapišimo števila od 1 do $b \cdot c$ v obliki tabele:

$0b+1$	$0b+2$	$0b+3$	...	$0b+b$
$1b+1$	$1b+2$	$1b+3$	...	$1b+b=2b$
$2b+1$	$2b+2$	$2b+3$	...	$2b+b=3b$
$\vdots$				
$(c-1)b+1$	$(c-1)b+2$	$(c-1)b+3$	...	$(c-1)b+b=cb$

Pokažimo, da v tabeli obstaja $\varphi(b)$ stolpcev proti b tujih števil, vsak izmed njih pa vsebuje $\varphi(c)$ števil, ki so tuja c.

a) V tabeli obstaja $\varphi(b)$ stolpcev proti b tujih števil:

V prvi vrstici je očitno b števil, ki so paroma nekongruentna po modulu b, vsebuje $\varphi(b)$ proti b tujih števil. Ker so vsa števila izbranega stolpca kongruentna zgornjemu elementu tega stolpca po modulu b, v tabeli obstaja $\varphi(b)$ stolpcev proti b tujih števil.

b) Vsak stolpec vsebuje $\varphi(c)$ števil, ki so tuja c:

V vsakem stolpcu je c števil, ki so paroma nekongruentna po modulu c, zato vsak stolpec vsebuje $\varphi(c)$ števil, ki so tuja c. Ker ima tabela b stolpcev, vsebuje $b \cdot \varphi(c)$ proti c tujih števil.

Pokažimo, sta poljubnega elementa danega stolpca nekongruentna po modulu c.

j-ti stolpec tabele vsebuje elemente: $0b + j, 1b + j, \ldots, (c-1)b + j$. Recimo, da sta i −ti in k −ti ($i < j$) element tega stolpca kongruentna po modulu c:

$$(i-1)b + j \equiv (k-1)b + j\ (c)$$

Na obeh straneh kongruence odštejmo j, nato pa delimo z b:

$$(i-1)b \equiv (k-1)b\ (c)$$
$$i - 1 \equiv k - 1(c)$$
$$i \equiv k(c),$$ to pa ni možno, ker sta i in k ostanka modulu c.

Ker v tabeli obstaja $\varphi(b)$ stolpcev proti b tujih števil, vsak izmed njih pa vsebuje $\varphi(c)$ števil, ki so tuja c, je števil, ki so tuja obenem proti b in proti c natanko $\varphi(b) \cdot \varphi(c)$, kar smo želeli dokazati. Konec dokaza.

Trditev 2.4. Vrednost Eulerjeve pri poljubnem naravnem številu m je

$$\varphi(m) = m \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_s}\right)$$

Dokaz. Naj bo m poljubno naravno število s praštevilskim razcepom $m = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_s^{\alpha_s}$. Računajmo:

$$\varphi(m) \underbrace{=}_{T.2.3} \varphi\left(p_1^{\alpha_1}\right) \cdot \varphi\left(p_2^{\alpha_2}\right) \cdot \ldots \cdot \varphi\left(p_s^{\alpha_s}\right) \underbrace{=}_{T.10.2}$$

$$p_1^{\alpha_1}\left(1 - \frac{1}{p_1}\right) \cdot p_2^{\alpha_2}\left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot p_s^{\alpha_s}\left(1 - \frac{1}{p_s}\right) =$$

$$m \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_s}\right)$$

Konec dokaza.

Za Eulerjevo funkcijo velja znana kongruenca, ki jo navaja Eulerjev izrek.
Eulerjev izrek. Za vsako proti modulu m tuje celo število a velja kongruenca: $a^{\varphi(m)} \equiv 1(m)$

Dokaz. Naj bodo $a_1 < a_2 < \cdots < a_{\varphi(m)} < m$ proti m tuji ostanki. Če vse te ostanke pomnožimo s proti m tujim številom a, dobimo $\varphi(m)$ števil $aa_1, aa_2, \ldots, aa_{\varphi(m)}$, ki so paroma nekongruentna po modulu m. Če bi bili namreč $i-$to in $j-$to izmed števil $aa_1, aa_2, \ldots, aa_{\varphi(m)}$ kongruentni po modulu m, bi dobili $aa_i \equiv aa_j(m) \overset{:a}{\Longrightarrow} a_i \equiv a_j(m)$, kar je protislovje, saj so števila a_i ostanki po modulu m.
Vsako od števil $aa_1, aa_2, \ldots, aa_{\varphi(m)}$ je kongruentno enemu izmed ostankov po modulu m. Zato jih lahko spravimo v pare:

$$a_1 \cdot a_2 \cdot \ldots \cdot a_{\varphi(m)} \equiv aa_1 \cdot aa_2 \cdot \ldots \cdot aa_{\varphi(m)}(m) \xrightarrow{:a_1 \cdot a_2 \cdot \ldots \cdot a_{\varphi(m)}}$$

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

Konec dokaza.

Opomba. Če kongruenco $a^x \equiv 1(m)$ (a tuje m) poimenujemo *eksponentna kongruenca*, dobi Eulerjev izrek obliko: Vrednost Eulerjeve funkcije pri danem modulu je rešitev eksponentne kongruence po tem modulu $a^x \equiv 1(m)$. Pri tem je a poljubno proti m tuje celo število.

3. INVERZ PO DANEM MODULU

Trditev 3.1 a) Linearna kongruenca $ax \equiv 1(m)$ je rešljiva natanko tedaj, ko je njen vodilni koeficient a tuj modulu.

b) Linearna kongruenca $ax \equiv 1(m)$ s proti vodilnemu koeficientu tujim modulom ima natanko eno rešitev do modula natančno.

Dokaz a) $\Leftarrow$) Naj velja $d(a,m)=1$. Dokažimo, da je kongruenca v tem primeru rešljiva. Izrazimo največji skupni delitelj števil a, m kot njuno linearno kombinacijo:

$$d(a,m)=1 \overset{\exists! k,l}{\Longrightarrow} ka+lm=1 \Rightarrow ka+\underbrace{lm}_{\equiv 0(m)} \equiv 1(m) \Rightarrow ak \equiv 1(m) \Rightarrow x_0=k$$

$\Rightarrow$) Naj obstaja x_0 kongruence $ax \equiv 1(m)$. Dokažimo, sta potem a in m tuji si števili. Označimo z n skupnega delitelja števil a in m.

$$ax_0 \equiv 1(m) \Rightarrow ax_0=1+km \Rightarrow\ ax_0-km=1 \Rightarrow n|1 \Rightarrow n=1 \Rightarrow d(a,m)=1$$

b) Naj bosta x_0 in y_0 rešitvi kongruence $ax \equiv 1(m)$. Potem velja $ax_0 \equiv 1(m)$ in $ay_0 \equiv 1(m)$, od koder z upoštevanjem simetričnosti in tranzitivnosti kongruenčne relacije dobimo $ax_0 \equiv ay_0(m)$. To kongruenco delimo z a in dobimo $x_0 \equiv y_0(m)$, kar smo želeli dokazati.

Konec dokaza.

S pomočjo linearne kongruence $ax \equiv 1(m)$ definiramo inverz proti modulu tujega celega števila.

Trditev 3.2. Za vsako proti modulu tuje celo število a obstaja natanko določen ostanek po tem modulu, ki reši kongruenco $ax \equiv 1(m)$. Rečemo mu $inverz$ števila a po modulu m in ga označimo z znanim zapisom a^{-1}.

Dokaz. Ker je a proti modulu tuje celo število, ima kongruenca $ax \equiv 1(m)$ neskončno paroma kongruentnih rešitev po modulu m. Ker so med seboj kongruentne, so vse kongruentne istemu ostanku po modulu m. Torej res obstaja natanko določen ostanek po tem modulu, ki reši kongruenco $ax \equiv 1(m)$. Konec dokaza.

Inverz proti modulu tujega celega števila je torej tisti natanko določen ostanek po tem modulu, za katerega velja, da je njun produkt kongruenten 1 po tem modulu:

$$x_0 \text{ je } inverz \text{ od } a \text{ po modulu } m,\ d(a,m)=1 \overset{def}{\Longleftrightarrow} ax_0 \equiv 1(m), 0 \leq x_0 < m$$

Trditev 3.3. Dve proti modulu tuji si števili imata enak inverz po tem modulu natanko tedaj, ko sta kongruentni.

Dokaz. Naj bosta a in b proti modulu tuji si celi števili. Pokažimo dano ekvivalenco v obe smeri:

$\Rightarrow$) Če je x_0 skupni inverz števil a in b, potem velja $ax_0 \equiv 1(m)$ in $bx_0 \equiv 1(m)$. Od tod po tranzitivnosti kongruence sledi $ax_0 \equiv bx_0(m)$. Ker je inverz celega števila tuj modulu, dobi ta kongruenco pri deljenju z x_0 obliko $a \equiv b(m)$, kar smo želeli dokazati.

$\Leftarrow$) Naj bosta a in b kongruentni števili po modulu m, a^{-1} in b^{-1} pa njuna inverza po tem modulu. Dokažimo, da sta inverza enaka. Iz $a \cdot a^{-1} \equiv 1(m)$ in $b \cdot b^{-1} \equiv 1(m)$ po tranzitivnosti kongruence sledi $a \cdot a^{-1} \equiv b \cdot b^{-1}(m)$, v kateri lahko število b zamenjamo z a, saj sta kongruentni. Pri tem dobimo kongruenco $a \cdot a^{-1} \equiv a \cdot b^{-1}(m)$, ki jo delimo z a, kar nam da kongruenco $a^{-1} \equiv b^{-1}(m)$. Ker pa sta a^{-1} in b^{-1} ostanka po modulu m, sta lahko kongruentna samo, če sta enaka. Torej res dobimo $a^{-1}=b^{-1}$. Konec dokaza.

Trditev 3.4. Inverz proti modulu tujega števila je tudi sam tuj modulu: $d(a^{-1}, m) = 1$.

Dokaz. Naj bo n skupni delitelj inverza in modula, se pravi $a^{-1} = k_1 n$ in $m = k_2 n$. Kongruenco $aa^{-1} \equiv 1(m)$ zapišimo v obliki $aa^{-1} = 1 + km$ in v njej upoštevajmo, da sta inverz in modul večkratnika števila n: $a \cdot k_1 n = 1 + k \cdot k_2 n$. Iz tega pa sledi $ak_1 n + kk_2 n = 1$, torej n deli 1, potem je enak 1. Inverz in modul sta zato tuji si števili. Konec dokaza.

Ker je inverz proti modulu tujega števila tuj temu modulu, ima po tem modulu po tem modulu svoj inverz, torej inverz od inverza danega proti modulu tujega števila.

Trditev 3.5. Inverz inverza proti modulu m tujega celega števila a je ostanek pri deljenju števila a z m.

$$(a^{-1})^{-1} \equiv a(m) \ \wedge \ 0 \leq (a^{-1})^{-1} < m$$

Dokaz. Naj velja $a \cdot a^{-1} \equiv 1(m)$ in $0 \leq a^{-1} < m$. Po osnovnem izreku o deljenju velja $a = km + r$ in $0 \leq r < m$. Pokažimo, da je r inverz od a^{-1} po modulu m. Res: $a^{-1} \cdot r \equiv a^{-1} \cdot a \equiv 1(m)$ in $0 \leq r < m$. Konec dokaza.

Opomba. Inverz inverza števila po danem modulu torej s splošnem ni enak, ampak manjši ali enak od tega števila:

$$(a^{-1})^{-1} \leq a$$

Enakost nastopi v primeru, ko je to število ostanek po tem modulu.

Med $\varphi(m)$ proti modulu tujih ostankov sta pri poljubnem modulu m tudi 1 in $m - 1$. Zanju velja, da sta sama svoja inverza: $1 \cdot 1 \equiv 1(m)$ in $(m-1) \cdot (m-1) = m^2 - 2m + 1 \equiv 1(m)$.

Praštevilskem modulu so razen števila 0 tuji vsi njegovi ostanki, zato imajo vsi neničelni ostanki po praštevilskem modulu svoje inverze. V naslednji trditvi pokažemo, da od 1 in $p - 1$ ne morejo biti sami svoji inverzi.

Trditev 3.6. Števili 1 in predhodnik praštevila sta edina ostanka po praštevilskem modulu, ki sta sama svoja inverza. Preostale neničelne ostanke po praštevilskem modulu lahko spravimo v pare med seboj inverznih števil.

Dokaz. Recimo, da je od 1 in $p - 1$ različen ostanek a po modulu p sam svoj inverz. Od tod sledi

$$a^2 \equiv 1(p) \implies p|a^2 - 1 \implies p|(a-1)(a+1)$$

Praštevilo p deli vsaj enega od faktorjev produkta: $p|a - 1 \vee p|a + 1$. Iz privzetka $1 < a < p - 1$ z odštevanjem in prištevanjem števila 1 sledi $0 < a - 1 < p - 2$ in $2 < a + 1 < p$, kar pomeni, da p ne deli niti $a - 1$ niti $a + 1$. Prišli smo do protislovja in trditev velja.

Konec dokaza.

Inverz proti modulu tujega števila lahko izračunamo s pomočjo Eulerjevega izreka.

Trditev 3.7. Inverz proti modulu tujega števila lahko izračunamo s pomočjo Eulerjevega izreka po formuli: $a^{-1} \equiv a^{\varphi(m)-1}(m)$

Dokaz.

$$a^{\varphi(m)} \equiv 1(m) \xrightarrow{\cdot a^{-1}} \left(a^{\varphi(m)-1} \cdot a\right) \cdot a^{-1}(m) \equiv a^{-1}(m) \implies a^{\varphi(m)-1}(m) \equiv a^{-1}(m)$$

Konec dokaza.

Iz Eulerjevega izreka enostavno sledi še en znan izrek teorije števil:

Mali Fermatov izrek. Za vsako celo število a in poljubno praštevilo p velja kongruenca $a^p \equiv a(p)$.
Ekvivalentno: Vsako celo število je kongruentno svoji p-ti potenci po modulu p.

Dokaz. Če je a proti p tuje celo število, po Eulerjevem izreku sledi $1 \equiv a^{\varphi(p)}(p) \equiv a^{p-1}(p)$ Če to kongruenco pomnožimo z a, dobimo $a \equiv a^p(p)$, kar smo želeli pokazati. Če število a ni tuje praštevilu p, je njegov večkratnik, torej velja $a \equiv 0(p)$. Če to upoštevamo v kongruenci $a^p \equiv a(p)$, dobimo $0^p \equiv 0(p)$, kar je res. Konec dokaza.

4. POTENCE Z VELIKIMI EKSPONENTI

Nekatere enkripcije temeljijo na kongruencah, ki vsebujejo velike potence celih števil. Reševanje kongruence

$$b^N \equiv x(m) \quad 0 \leq x < m, \qquad b, N, m > 0$$

katere rešitev je ostanek potence po danem modulu, poimenujmo $modularno\ potenciranje$.
Postopek modularnega potenciranja je sledeč:

1) Eksponent N zapišemo v dvojiškem sistemu: $N_{10} = (a_k a_{k-1} \dots a_1 a_0)_2$
2) Z zaporednim kvadriranjem zapišemo ostanke potenc $b^{(2^i)}$ po modulu m: $b^{(2^i)} \equiv r_i(m)$
3) Rešitev je ostanek produkta tistih ostankov r_i, za katere je $a_i = 1$, po modulu m

Premislimo točko 3):

$$\boldsymbol{b^N} = b^{a_k \cdot 2^k + a_{k-1} \cdot 2^{k-1} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0} = b^{a_k \cdot 2^k} \cdot b^{a_{k-1} \cdot 2^{k-1}} \cdot \dots \cdot b^{a_1 \cdot 2^1} \cdot b^{a_0 \cdot 2^0}$$
$$\equiv r_k \cdot r_{k-1} \cdot \dots \cdot r_1 \cdot r_0(\boldsymbol{m})$$

Ostanki r_i, pri katerih je $a_i = 0$, so enaki 1, torej pomnožimo le tiste ostanke potenc $b^{(2^i)}$, za katere je $a_i = 1$.

Izvedimo ta postopek na naslednjem primeru:

Primer. Določimo ostanek pri deljenju števila 7^{123} s številom 121 .
1) Eksponent 123 zapišemo v dvojiškem sistemu: $123_{10} = (1111011)_2$

2) Z zaporednim kvadriranjem in deljenjem po modulu 121 zapišemo potence $7^{(2^k)}$:

$$7 \equiv 7(121)$$
$$7^2 \equiv 49(121)$$
$$7^4 \equiv 102(121)$$
$$7^8 \equiv 119(121)$$
$$7^{16} \equiv 4(121)$$
$$7^{32} \equiv 16(121)$$
$$7^{64} \equiv 14(121)$$

$$\mathbf{7^{123}} = 7^{2^6+2^5+2^4+2^3+2^1+1} = 7^{64} \cdot 7^{32} \cdot 7^{16} \cdot 7^8 \cdot 7^2 \cdot 7^1 \equiv$$
$$14 \cdot 16 \cdot 4 \cdot 119 \cdot 49 \cdot 7 \equiv \mathbf{24}\ (\mathbf{121})$$

Dodajmo še program za ta postopek

```
''' 0 < b < m. Funkcija seznam generira seznam ostankov potenc
b**(2**0),...,b**(2**k) po modulu m. Pri tem je k največje
število, pri katerem še velja 2**(2**k)<= N. '''

def seznam(b,N,m):
    s=[b]
    for i in range(1,len(bin(N))-2):
        s.append(s[i-1]**2%m)
    return s

# ostanek potence b**N po modulu m
def modularna_potenca(b,N,m):
    s_1=list(map(int,list(bin(N)[2:]))) # seznam števk v dvojiškem zapisu eksp. N
    s_2=seznam(b,N,m)[::-1] # obrne seznam ostankov
    rezultat=1 # vrednost b**N po modulu m
    for i in range(0,len(s_2)):
        if s_1[i]!=0:
            rezultat=(rezultat*s_1[i]*s_2[i])%m
    return rezultat
```

Modularno potenciranje potrebujemo za določanje inverza števila po danem modulu pri eksponentnem šifriranju. Program za določanje tega inverza temelji na Eulerjevem izreku.

```
'''Funkcija inverz(a,m,sl) vrne inverz števila a po modulu m.
sl je slovar, v katerem je spravljen razcep od m:
ključi so prafaktorji, vrednosti pa njihovi eksponenti)'''

def inverz(a,m,sl):
    fi=1 # vrednost Eulerjeve funkcije pri modulu m
    for p in sl:
        pom=p**sl[p]-p**(sl[p]-1)# vrednost fi pri p**sl[p]
        fi=fi*pom
    return modularna_potenca(a,fi-1,m)
```

Opomba. Funkcije iz tega poglavja se nahajajo v datoteki modularno_potenciranje.

5. LINEARNA KONGRUENCA IN SISTEM DVEH LINEARNIH KONGRUENC Z DVEMA NEZNANKAMA

Definicija. *Popolni sistem ostankov po danem modulu* m (oznaka: $pso(m)$) je množica m števil, ki so po tem modulu paroma nekongruentna:

$$R \coloneqq \{r_1, \dots, r_m\} \text{ je } pso(m) \overset{def}{\iff} \forall i \neq j \in \{0, \dots, m\}: r_i \not\equiv r_j(m)$$

Trditev 5.1. Če je $R \coloneqq \{r_1, \dots, r_m\}$ popolni sistem ostankov po modulu m in je a proti m tuje število, b pa poljubno število, je tudi množica $R \coloneqq \{ar_1 + b, \dots, ar_m + b\}$ popolni sistem ostankov po modulu m.

Dokaz. Pokažimo, da sta za vsaka $i \neq j \in \{1, \dots, m\}$, števili $ar_i + b$ in $ar_j + b$ nekongruentni po modulu m:
Recimo, da sta pri neki izbiri i, j ti dve števili kongruentni: $ar_i + b \equiv ar_j + b(m)$. Od tod sledi $ar_i \equiv ar_j(m)$, po deljenju z a pa $r_i \equiv r_j(m)$, kar je protislovno.
Ker so števila iz R paroma nekungruentna, nobeni dve med njimi nista enaki, torej jih je m.
Konec dokaza.

Trditev 5.2. Funkcija, ki vsakemu številu iz popolnega sistema ostankov $R \coloneqq \{r_1, \dots, r_m\}$ po modulu m priredi njegov ostanek po tem modulu, je bijekcija.

Dokaz. Množica R in množica vseh ostankov po danem modulu sta enako močni. Funkcija je obenem injektivna, ker so števila iz $pso(m)$ nekongruentna. Injektivna funkcija med enako močnima množicama pa je tudi surjektivna in zato bijektivna. Konec dokaza.

Linearna kongruenca je kongruenca, ki ima same linearne in konstantne člene. Zapišemo jo v obliki $ax \equiv b(m)$ in uporabimo enake pojme kot pri linearni funkciji. Številu a recimo vodilni koeficient, številu b pa prosti člen linearne kongruence. Posebnega pomena je že opisana linearna kongruenca $ax \equiv 1(m)$, s pomočjo katere definiramo inverz proti modulu tujega celega števila.
O rešitvah linearne kongruence govori naslednja trditev:

Trditev 5.3. 1) Če največji skupni delitelj vodilnega koeficienta in modula ne deli prostega člena, linearna kongruenca nima rešitve.
2) Če sta vodilni koeficient in modul linearne kongruence tuji si števili, ima ta kongruenca natanko eno rešitev do modula natančno.
3) Če največji skupni delitelj (d) vodilnega koeficienta in modula deli prosti člen, ima linearna kongruenca d nekongruentnih rešitev po svojem modulu.
Natančnejši zapis trditve:
Naj bo $ax \equiv b(m)$ linearna kongruenca in naj bo $d \coloneqq d(a, m)$ največji skupni delitelj vodilnega koeficienta in modula. Potem velja:

- $d \nmid b \implies$ kongruenca nima rešitve
- $d = 1 \implies$ kongruenca ima natanko eno rešitev po modulu m.

- $d \mid b \implies$ kongruenca ima natanko d nekongruentnih rešitev po modulu m, in sicer:

$$x_1, x_1 + m_1, \dots, x_1 + (d-1)m_1,$$

pri čemer je x_1 (enolična) rešitev kongruence $\frac{a}{d}x \equiv \frac{b}{d}(\frac{m}{d})$. V posebnem primeru $d = 1$, ima kongruenca nazanko eno rešitev do modula natančno.

Dokaz. 1) Najprej poglejmo linearno kongruenco, za katero velja: $d \nmid b$. S protislovjem pokažimo, da nima rešitve. Recimo torej, da ima rešitev x_0. Potem velja $ax_0 \equiv b(m)$ in od tod $a_1 d x_0 \equiv b(m_1 d)$. Ker se med seboj kongruentni števili po danem modulu razlikujeta za večkratnik tega modula, sledi $a_1 d x_0 = b + l m_1 d$, od tod pa $d \mid b$, kar je protislovje.

2) Poglejmo primer, ko velja $d = 1$ in pokažimo, da ima tedaj kongruenca po svojem modulu natanko eno rešitev. Množica $\{0,\ 1, \dots, m-1\}$ je popolni sistem ostankov po modulu m. Ker število a tuje modulu, je tudi množica $\{0a,\ 1a, \dots, (m-1)a\}$ popolni sistem ostankov po modulu m. Potem pa obstaja natanko določeno število ia v tej množici, ki je kongruentno b po modulu m, se pravi $ia \equiv ai \equiv b(m)$. To pa pomeni, da je število i rešitev kongruence $ax \equiv b(m)$ po modulu m. Pokažimo še enoličnost: recimo, da velja tudi $aj \equiv b(m)$. Od tod sledi $ai \equiv aj(m)$, po deljenju z a pa $i \equiv j(m)$.

3) Ostane še zadnja možnost, ko $d \mid b$ in $d > 1$. Količnik pri deljenju b in d označimo s k. Pokažimo, da ima tedaj linearna kongruenca d nekongruentnih rešitev po modulu m.
.) Naj prej ugotovimo, da lahko kongruenco $ax \equiv b(m)$ v tem primeru preoblikujemo v ekvivalentno kongruenco $a_1 x \equiv k(m_1)$:

$$ax_0 \equiv b(m) \Leftrightarrow \underbrace{a_1 d}_{=a} x_0 \equiv kd\ (\underbrace{m_1 d}_{=m}) \overset{:d}{\Longleftrightarrow} a_1 x_0 \equiv k(m_1)$$

Število x_0 torej reši kongruenco $ax \equiv b(m)$ natanko tedaj, ko reši kongruenco $a_1 x \equiv k(m_1)$, torej sta res ekvivalentni. Poslej se bomo torej ukvarjali s kongruenco $a_1 x \equiv k(m_1)$. Njen vodilni koeficient je tuj modulu, zato ima po točki 2) te trditve natanko eno rešitev (označimo jo z x_0) po modulu m_1. Množica vseh njenih rešitev je torej $\{x_0 + t m_1; t \in \mathbb{Z}\}$. Poglejmo, katerim ostankom modula m so kongruentna ta števila. Število t izrazimo po osnovnem izreku o deljenju kot $t = ld + r,\ 0 \le r < d$:

$$x_0 + t m_1 = x_0 + (ld + r)m_1 = x_0 + l\underbrace{d m_1}_{=m} + r m_1 \equiv x_0 + r m_1\ (m)$$

Vsaka rešitev kongruence $ax \equiv b(m)$ je po modulu m kongruentna enemu od naslednjih števil:

$$x_0 + 0m_1,\ \ x_0 + 1m_1, \dots,\ \ x_0 + (d-1)m_1$$

S protislovjem pokažimo, da so ta števila po modulu m nekongruentna. Recimo, da velja

$$x_0 + v m_1 \equiv x_0 + u m_1\ \ (m) \qquad 0 \le v < u \le d - 1$$

$$(u - v)m_1 \equiv 0\ \ (m)\ \ /: m_1$$

$$u - v \equiv 0\ \ (d),$$

od tod pa sledi $d|u-v$, kar je protislovno, saj velja $0 < u - v < d$.
Iz osnov kongruenc vemo, da so rešitve $ax \equiv b(m)$ tudi vsa števila, ki so po modulu m kongruentna številom $x_0,\ x_0 + m_1, \ldots,\ x_0 + (d-1)m_1$. Torej ima kongruenca $ax \equiv b(m)$ po modulu m res d nekongruentnih rešitev. Konec dokaza.

Opomba. Rešitve linearne kongruence lahko poiščemo s poskušanjem po popolnem sistemu ostankov, kar pa postane v primeru velikega modula zamudno.
Če je vodilni koeficient tuj modulu, lahko rešitev izračunamo s pomočjo Eulerjevega izreka:

$$ax \equiv b(m) \quad /\cdot a^{-1}$$

$$\boldsymbol{x \equiv ba^{-1} \equiv ba^{\varphi(m)-1}(m)}$$

Relacijo kongrunce definirajmo še na množici urejenih parov celih števil: urejena para sta kongruentna po danem modulu, ko sta po tem modulu kongruentni njuni prvi komponenti in njuni drugi komponenti:

$$(x_0, y_0) \equiv (x_1, y_1)\ (mod\ m) \overset{def}{\Leftrightarrow} x_0 \equiv x_1(m)\ \wedge\ y_0 \equiv y_1(m)$$

Poglejmo linearno kongruenco z dvema neznankama: $ax + by \equiv c(m)$. Števili a in b imenujemo koeficienta te kongruence, c pa njen prosti člen. Njena rešitev je urejeni par celih števil (x_0, y_0), pri katerem sta njena levi in desni izraz kongruentna: $ax_0 + by_0 \equiv c(m)$.
Vsak urejeni par, ki je kongruenten dani rešitvi kongruence, je tudi sam rešitev te kongruence. Če so vsi urejeni pari, ki rešijo kongruenco $ax + by \equiv c(m)$ med seboj kongruentni po modulu m, rečemo, da ima ta kongruenca natanko eno rešitev do modula natančno.
V zvezi z rešitvami linearne kongruence z dvema neznankama velja podobna trditev kot pri linearni kongruenci z eno neznanko.

Trditev 5.4. Za linearno kongruenco z dvema neznankama $ax + by \equiv c(m)$, pri čemer so a, b, c cela števila, $m > 0$ modul in $d := d(a, b, m)$, velja:
- Če d ne deli prostega člena (c), potem kongruenca nima rešitve.
- Če d deli prosti člen (c), potem ima kongruenca natanko dm nekongruentnih rešitev po modulu m.

Dokaz. 1) Najprej pokažimo, da velja prva poved trditve. Če je urejeni par (x_0, y_0) rešitev kongruence, velja $ax_0 + by_0 \equiv c(m)$ in zato obstaja celo število k, da velja $ax_0 + by_0 + km = c$. Ker d deli števila a, b, m, deli tudi njihovo linearno kombinacijo c.

.) Dokažimo še drugo poved. Naj bo d delitelj prostega člena c. Kongruenca $ax + by \equiv c(m)$ je ekvivalentna kongruenci $ax \equiv c - by(m)$, ta pa ima po trditvi 5.3. rešitve, če velja $g := d(a, m) \mid c - by$. V tem primeru ima natanko g nekongruentnih rešitev po modulu m: $x_0 + 0\frac{m}{g},\ x_0 + 1\frac{m}{g},\ x_0 + 2\frac{m}{g}, \ldots,\ x_0 + (g-1)\frac{m}{g}$, pri čemer je x_0 po modulu $\frac{m}{g}$ edina rešitev kongruence $\frac{a}{g}x \equiv \frac{c-by}{g}(\frac{m}{g})$.

.) Kongruenca $ax + by \equiv c(m)$ je ekvivalentna kongruenci $ax \equiv c - by(m)$, ta pa ima pri danem y rešitve pod pogojem $g \mid c - by$, ki pa je ekvivalenten kongruenci $by \equiv c(g)$. Linearna kongruenca $ax \equiv c - by(m)$ ima torej rešitve pod pogojem, da velja $by \equiv c(g)$. Pri reševanju linearne kongruence z dvema neznankama $ax + by \equiv c(m)$ je potrebno torej najprej poiskati rešitve linearne kongruence $by \equiv c(g)$, nato pa pri vsaki od njenih rešitev y_0 poiskati še rešitve linearne kongruence $ax \equiv c - by_0(m)$.

.) Poglejmo torej kongruenco $by \equiv c(g)$. Ker je to linearna kongruenca, bo imela rešitve v primeru, ko bo veljalo $d(b, g) \mid c$. Premislimo, da velja $d(b, g) = d(a, b, m) = d$:

$$d(b, g) = d(b, d(a, m)) = d(a, b, m) = d$$

Pogoj za obstoj rešitev $d(b, g) \mid c$ je torej ekvivalenten pogoju $d|c$. Kongruenca $by \equiv c(g)$ ima tedaj d nekongruentnih rešitev po modulu g, in sicer

$$y_0 + 0\frac{g}{d},\ y_0 + 1\frac{g}{d}, \dots,\ y_0 + (d-1)\frac{g}{d},$$

pri čemer je y_0 (enolična) rešitev kongruence $\frac{b}{d}y \equiv \frac{c}{d}(\frac{g}{d})$. Ta števila so nekongruenta po modulu m.
Ker vsako od njih določa $\frac{m}{g}$ (nekongruentnih) ostankov po modulu m, s tem dobimo $d \cdot \frac{m}{g}$ po m nekongruentnih števil, ki so rešitve kongruence $by \equiv c(g)$. Za vsako od njih nadalje dobimo g nekongruentnih rešitev kongruence $ax + by \equiv c(m)$ po modulu m, kar pomeni, da ima kongruenca $ax + by \equiv c(m)$ po modulu m vsega $\left(d \cdot \frac{m}{g}\right) \cdot g = dm$ rešitev, kar smo želeli dokazati. Konec dokaza.

Trditev 5.5. Sistem dveh linearnih kongruenc z dvema neznankama in s proti modulu tujo determinanto ima po tem modulu natanko eno rešitev.
Natančneje: Sistem dveh linearnih kongruenc s celimi koeficienti a, b, c, d, e, f in s proti modulu tujo determinanto $D = ad - bc$, $d(D, m) = 1$, to je

$$ax + by \equiv e(m)$$
$$cx + dy \equiv f(m)$$

ima po modulu m natanko eno rešitev, in sicer

$$x \equiv D^{-1}(de - bf)(m)$$
$$y \equiv D^{-1}(af - ce)(m),$$

pri čemer je D^{-1} inverz determinante D po modulu m, se pravi $D \cdot D^{-1} \equiv 1(m)$.

Dokaz. Naj bo (x, y) rešitev tega sistema.
a.) Prvo kongruenco sistema pomnožimo z d, drugo pa z b in sistem dobi obliko

$$adx + bdy \equiv de(m)$$

$$bcx + bdy \equiv bf(m)$$

Dobljeni kongruenci odštejmo in dobimo kongruenco

$$adx - bcx \equiv de - bf(m),$$

oziroma (z upoštevanjem determinante)

$$Dx \equiv de - bf(m).$$

Dobljeno kongruenco pomnožimo z inverzom determinante po modulu m in dobimo

$$x \equiv \boldsymbol{D^{-1}(de - bf)(m)}$$

b.) Prvo kongruenco sistema pomnožimo z c, drugo pa z a in sistem dobi obliko

$$acx + bcy \equiv ce(m)$$
$$acx + ady \equiv af(m)$$

Dobljeni kongruenci odštejmo in dobimo kongruenco

$$ady - bcy \equiv af - ce(m),$$

oziroma (z upoštevanjem determinante)

$$Dy \equiv af - ce(m).$$

Dobljeno kongruenco pomnožimo z inverzom determinante po modulu m in dobimo

$$\boldsymbol{y \equiv D^{-1}(af - ce)(m)}$$

.) V točkah $a)$ in $b)$ tega dokaza smo pokazali naslednje: Če je urejeni par celih števil (x, y) rešitev danega sistema kongruenc, potem velja

$$x \equiv D^{-1}(de - bf)(m), \;\; y \equiv D^{-1}(af - ce)(m)$$

Pokažimo še, da je vsak par števil take oblike rešitev sistema:

$$\boldsymbol{ax + by} \equiv aD^{-1}(de - bf) + bD^{-1}(af - ce) \equiv D^{-1}(ade - abf + baf - bce) \equiv$$

$$D^{-1}(ad - bc)e \equiv D^{-1}De \equiv \boldsymbol{e(m)}$$

ter

$$\boldsymbol{cx + dy} \equiv cD^{-1}(de - bf) + dD^{-1}(af - ce) \equiv D^{-1}(cde - cbf + daf - dce) \equiv$$

$$D^{-1}(ad - bc)f \equiv D^{-1}Df \equiv \boldsymbol{f(m)}$$

Konec dokaza.

6. RAZŠIRJENI EVKLIDOV ALGORITEM

Izračun inverza proti modulu tujega števila po tem modulu, to je $a^{-1} \equiv a^{\varphi(m)-1}(m)$, predpostavlja faktorizacijo modula, ta pa pri velikem modulu lahko postane zelo zahtevna.
Zato v kriptografiji za izračun inverza števila po danem modulu poleg tega načina uporabljamo še t.i. razširjeni Evklidov algoritem, ki omogoča izračun inverza brez faktorizacije modula oziroma poznavanja $\varphi(m)$. V posebnih primerih, ko je modul praštevilo ali $\varphi(m)$ poznamo vnaprej, pa je zaradi učinkovitega potenciranja v modularni aritmetiki izračun inverza ugodnejši z uporabo Eulerjeve funkcije. V tem poglavju si natančneje poglejmo razširjeni Evklidov algoritem.

Z zapisom $S(a,b)$ označimo množico vseh skupnih deliteljev števil a in b.

Trditev 6.1. Naj bodo deljenec, delitelj, količnik in ostanek poljubna nenegativna števila iz osnovnega izreka o deljenju. Množica skupnih deliteljev od deljenca in delitelja je enaka množici skupnih deliteljev od delitelja in ostanka:

$$S(deljenec, delitelj) = S(delitelj, ostanek)$$

Dokaz.) Vpeljimo naslednje oznake $a := deljenec, m := delitelj,\ r := ostanek$. Pokazati moramo enakost dveh množic. To storimo tako, da pokažemo, da sta ti dve množici podmnožici druga druge:

$$S(a,m) = S(m,r) \Leftrightarrow S(a,m) \subset S(m,r) \wedge S(m,r) \subset S(a,m)$$

$\subset$) $\quad x \in S(a,m) \Longrightarrow x|a \wedge x|m \overset{\exists! l,s}{\Longrightarrow} a = sx \ \wedge\ m = lx \overset{a=km+r}{\Longrightarrow} sx = k \cdot lx + r \Longrightarrow$
$r = sx - k \cdot lx \Longrightarrow x(s-kl) = r \Longrightarrow x|r \Longrightarrow x \in S(m,r)$

$\supset$) $\quad x \in S(m,r) \Longrightarrow x|m \wedge x|r \overset{\exists! s,l}{\Longrightarrow} m = sx \wedge r = lx \overset{a=km+r}{\Longrightarrow} a = ksx + lx \Longrightarrow$

$a = x(ks+l) \Longrightarrow x|a \Longrightarrow x \in S(a,m)$ Konec dokaza.

Trditev 6.2. Naj bodo deljenec, delitelj, količnik in ostanek poljubna nenegativna števila iz osnovnega izreka o deljenju. Največji skupni delitelj od deljenca in delitelja je enak največjemu skupnemu delitelju od delitelja in ostanka: $d(deljenec, delitelj) = d(delitelj, ostanek)$.

Dokaz. Trditev očitno sledi iz trditve T.6.1.

Ponovimo Evklidov algoritem: Naj bosta a (deljenec) in m (delitelj) poljubni naravni števili, pri čemer je a večje od m. Deljenca označimo z $\boldsymbol{r_0 := a}$, delitelja pa z $\boldsymbol{r_1 := m}$. Nato na vsakem koraku uporabimo osnovni izrek o deljenju tako, da novonastala delitelj in ostanek zavzameta vlogi deljenca in delitelja v naslednjem koraku postopka:

$$r_0 = k_1 r_1 + r_2, \qquad 0 \le r_2 < r_1\,,$$

$$
\begin{aligned}
r_1 &= k_2 r_2 + r_3, && 0 \le r_3 < r_2\,,\\
&\vdots\\
r_{i-2} &= k_{i-1} r_{i-1} + r_i, && 0 \le r_i < r_{i-1}\,,\\
&\vdots\\
r_{n-3} &= k_{n-2} r_{n-2} + r_{n-1}, && 0 \le r_{n-1} < r_{n-1}\\
r_{n-2} &= k_{n-1} r_{n-1} + r_n, && 0 \le r_n < r_{n-1}\\
r_{n-1} &= k_n r_n + 0
\end{aligned}
$$

V naslednji trditvi pokažimo, da nas Evklidov algoritem res pripelje do največjega skupnega delitelja dveh števil ter do še ene ugotovitve.

Trditev 6.3. a) Največji skupni delitelj dveh števil je zadnji od 0 različen ostanek Evklidovega algoritma.
b) Neko število deli dve drugi števili natanko tedaj, ko deli njun največji skupni delitelj. Oziroma: skupni delitelji dveh števil so natanko delitelji njunega največjega skupnega delitelja.

Dokaz. V točki a) želimo pokazati enakost $d(a,b) = r_n$. Dobimo jo iz trditve 6.2:

$$d(a,b) = d(r_0, r_1) = d(r_1, r_2) = \cdots = d(r_{n-2}, r_{n-1}) = d(r_{n-1}, r_n) = d(r_n, 0) = r_n.$$

V točki b) želimo pokazati enakost $S(a,b) = \{n \in \mathbb{N}; n | d(a,b)\}$. Dobimo jo iz trditve 6.2:

$$S(a,b) = S(r_0, r_1) = \cdots = S(r_{n-1}, r_n) = S(r_n, 0) =$$

$$\{n \in \mathbb{N}; n | r_n\} = \{n \in \mathbb{N}; n | d(a,b)\}$$ Konec dokaza.

V tem zapisu je $k_1, \dots, k_n$ zaporedje količnikov, $r_0, r_1, \dots, r_n - 1$ pa zaporedje deljencev. Postopek ima n korakov, njegov rezultat, se pravi največji skupni delitelj števil a in m, pa je r_n.
Evklidov algoritem razširimo tako, da na vsakem njegovem koraku izračunamo še par t.i. $koeficientov$: α_i in β_i, $i \in \{0, \dots, n-1\}$ in sicer takole:

$$\alpha_0 \overset{\text{def}}{=} 1, \qquad \alpha_1 \overset{\text{def}}{=} 0, \qquad \alpha_i \overset{\text{def}}{=} \alpha_{i-2} - k_{i-1} \cdot \alpha_{i-1}, \quad i \in \{2, \dots, n-1\}$$

$$\beta_0 \overset{\text{def}}{=} 0, \qquad \beta_1 \overset{\text{def}}{=} 1, \qquad \beta_i \overset{\text{def}}{=} \beta_{i-2} - k_{i-1} \cdot \beta_{i-1}, \quad i \in \{2, \dots, n-1\}$$

Na koncu po isti formuli izračunamo še števili α_n in β_n:

$$\alpha_n \overset{\text{def}}{=} \alpha_{n-2} - k_{n-1} \cdot \alpha_{n-1} \quad \text{in} \quad \beta_n \overset{\text{def}}{=} \beta_{n-2} - k_{n-1} \cdot \beta_{n-1}$$

Opomba. Opazimo, da podobna rekurzivna zveza kot za koeficiente α_i, β_i velja tudi za števila r_i. Iz Evklidovega algoritma preberemo: $r_0 = a, r_1 = m$, $r_i = r_{i-2} - k_{i-1} r_{i-1}$.

Trditev 6.4. Razširjeni Evklidov algoritem ima enako število (n) korakov kot Evklidov algoritem, njegov rezultat pa so tri števila:
- r_n: največji skupni delitelj števil a in m: $r_n = d(a,m)$,
- α_n, β_n: koeficienta linearne kombinacije deljenca a in delitelja m, ki je enaka njunemu največjemu skupnemu delitelju: $\alpha_n \cdot a + \beta_n \cdot m = d(a,m)$

Dokaz. Ker je dokaz Evklidovega algoritma poznan, pokažimo le, da velja razširjeni del, to je, da velja enakost $\alpha_n \cdot a + \beta_n \cdot m = r_n$. Uporabimo izrek o popolni indukciji in pokažimo, da za vsak $i \in \{0, \dots, n\}$ velja enakost $\alpha_i \cdot a + \beta_i \cdot m = r_i$.
.) $i = 0$: $\boldsymbol{\alpha_0 \cdot a + \beta_0 \cdot m} = r_0$:$\alpha_0 \cdot a + \beta_0 \cdot m = 1 \cdot a + 0 \cdot m = a = \boldsymbol{r_0}$.

.) $i = 1$: $\boldsymbol{\alpha_1 \cdot a + \beta_1 \cdot m} = 0 \cdot a + 1 \cdot m = m = \boldsymbol{r_1}$

.) Dokažimo še implikacijo: če enakost $\alpha_j \cdot a + \beta_j \cdot m = r_j$ velja za vse $j \in \{0, \dots, i-1\}$ potem velja tudi pri i, torej pokažimo, da potem velja $\alpha_i \cdot a + \beta_i \cdot m = r_i$:

$$\boldsymbol{\alpha_i \cdot a + \beta_i \cdot m} = (\alpha_{i-2} - k_{i-1}\alpha_{i-1})a + (\beta_{i-2} - k_{i-1}\beta_{i-1})m =$$
$$\alpha_{i-2}a - k_{i-1}\alpha_{i-1}a + \beta_{i-2}m - k_{i-1}\beta_{i-1}m = (\alpha_{i-2}a + \beta_{i-2}m) - k_{i-1}(\alpha_{i-1}a + \beta_{i-1}m) = *$$

Nadaljujemo po indukcijski predpostavki: $* = r_{i-2} - k_{i-1}r_{i-1} = \boldsymbol{r_i}$. Konec dokaza.

Razširjeni Evklidov algoritem pokažimo na primeru števil 3289 in 899. Program se nahaja v datoteki razširjeni_Evklidov_algoritem.

$3289 = 3 * 899 + 592$	$\alpha_0 = 1$	$\beta_0 = 0$
$899 = 1 * 592 + 307$	$\alpha_1 = 0$	$\beta_1 = 1$
$592 = 1 * 307 + 285$	$\alpha_2 = 1 - 3 * 0 = 1$	$\beta_2 = 0 - 3 * 1 = -3$
$307 = 1 * 285 + 22$	$\alpha_3 = 0 - 1 * 1 = -1$	$\beta_3 = 1 - 1 * -3 = 4$
$285 = 12 * 22 + 21$	$\alpha_4 = 1 - 1 * -1 = 2$	$\beta_4 = -3 - 1 * 4 = -7$
$22 = 1 * 21 + 1$	$\alpha_5 = -1 - 1 * 2 = -3$	$\beta_5 = 4 - 1 * (-7) = 11$
$21 = 21 * 1 + 0$	$\alpha_6 = 2 - 12 * -3 = 38$	$\beta_6 = -7 - 12 * 11 = -139$
	$\alpha_7 = -3 - 1 * 38 = -41$	$\beta_7 = 11 - 1 * -139 = 150$

Preverimo, da res velja $-41 * 3289 + 150 * 899 = 1$.

V nadaljevanju zapišimo še rekurzivno funkcijo $gcd(a,m)$, ki vrne samo največji skupni delitelj $\boldsymbol{d} := d(a,m)$ ter koeficienta linearne kombinacije $\boldsymbol{\alpha_n \cdot a + \beta_n \cdot m} = d$.
V funkciji $gcd(a,m)$ rekurzivno pokličemo njo samo na številih m in $a\%m$. V zadnjem koraku je to klic funkcije $gcd(d,0)$, ki vrne koeficienta 1 in 0, saj velja: $1 \cdot d + 0 \cdot m = d$. Premislimo, kako algoritem pri prehajanju skozi rekurzivne klice nazaj izračunava koeficienta α_i in β_j: Če $gcd(u,v)$ vrne koeficienta x in y, to pomeni enakost $d = x \cdot u + y \cdot v$. Funkcija $gcd(u,v)$ pokliče funkcijo $gcd(v, u\%v)$, ki vrne koeficienta x_1 in y_1, to pa pomeni enakost $d = x_1 \cdot v + y_1 \cdot (u\%v)$. Velja namreč $d(a,m) = d(r_0,r_1) = d(r_1,r_2) = \cdots = d(r_{n-1},r_n) = d(r_{n-1},r_n) = d(r_n,0)$.

Iz koeficientov x_1 in y_1 enostavno izračunamo koeficienta x in y tako, da iz enačbe $u = (\underbrace{u//v}_{:=k})v + (u\%v)$ izrazimo $u\%v$ in to upoštevajmo v drugi enakosti:

$$d = x_1 \cdot v + y_1 \cdot (u\%v) = x_1 \cdot v + y_1 \cdot (u - kv) = \boldsymbol{y_1} \cdot u + (\boldsymbol{x_1} - \boldsymbol{k y_1}) \cdot v$$

Z upoštevanjem prve enakosti: $d = \boldsymbol{x} \cdot u + \boldsymbol{y} \cdot v$ dobimo $\boldsymbol{x} = \boldsymbol{y_1}$ in $\boldsymbol{y} = \boldsymbol{x_1} - \boldsymbol{k y_1}$.

```
'''Rekurzivna funkcija gcd za razširjeni Evklidov algoritem sprejme
deljenca in delitelja ter vrne njun d, ter koeficienta α_n in β_n.'''
def gcd(u, v):
    if v == 0:
        return (u, 1, 0)
    else:
        g, x1, y1 = gcd(v, u % v)
        k = u // v
        return (g, y1, x1 - k* y1)
```

Razširjeni Evklidov algoritem v kriptografiji uporabljamo za izračun inverza števila po danem modulu. Poiskati inverz torej pomeni rešiti ekvivalenco $ax \equiv 1(m)$ $x \in \{0,1,\dots,m-1\}$. Ker sta a in m tuji si števili, je njun največji skupni delitelj enak 1. Potem iz razširjenega Evklidovega algoritma dobimo števili α_n in β_n, za kateri velja $\alpha_n a + \beta_n m = 1$. Od tod pa sledi $\alpha_n a \equiv 1(m)$, oziroma $\alpha_n (mod\ m)$ je iskani inverz.

```
''' Funkcija inverz s pomočjo razširjenega Evklidovega algoritma vrne inverz
proti modulu tujega števila po tem modulu.'''
def inverz(a, m):
    g, x, y = gcd(a, m)
    if g != 1:  # Če števili nista tuji, inverz ne obstaja
        return 0
    else:
        return (x % m)
```

7. AFINA TRANSFORMACIJA

Definicija. *Afina transformacija* določena z modulom m, s proti modulu tujim ostankom a ter s poljubnim ostankom b, je bijektivna funkcija na množici ostankov danega modula, definirana s predpisom $f(x) = (ax + b)\%\, m$, pri $\%$ pomeni ostanek pri deljenju z modulom. Natančneje: Naj bo $D = \{0,\dots,m-1\}$ množica vseh ostankov modula m in naj velja $a, b \in D$, $d(a,m) = 1$.

$$f: D \longrightarrow D$$
$$\boldsymbol{f(x) = (ax + b)\%\, m}$$

Dokažimo, da je f bijekcija. Po trditvi 5.1. je množica $\{a \cdot 0 + b, \dots, a \cdot (m-1) + b\}$ popolni sistem ostankov po modulu m, oziroma funkcija $f_1(x) \overset{\text{def}}{=} ax + b$ je injekcija. Po trditvi 5.2. pa je funkcija $f_2(x) \overset{\text{def}}{=} x\%\, m$ definirana na $Z_{f_1} = \{f_1(0), \dots, f_1(m-1)\}$ bijekcija. Funkcija f je kompozitum funkcij f_2 in f_1, saj za vsak $x \in D$ velja

$$\boldsymbol{f_2 \circ f_1}(x) = f_2\big(f_1(x)\big) = f_2(ax + b) = (ax + b)\%\, m = \boldsymbol{f}(x)$$

Funkcija f je torej kompozitum bijekcije in injekcije: $D \xrightarrow{f_1} Z_{f_1} \xrightarrow{f_2} D$, zato je bijekcija.

Konec dokaza.

Trditev 7.1. Inverzna funkcija afine transformacije $f(x) = (ax + b)\ \%\ m$ afina transformacija, dana s predpisom

$$g(x) = (a^{-1}x - (a^{-1}b)\%\ m)\%\ m$$

Opomba. Vrednosti funkcije g lahko računamo z enostavnejšim predpisom:

$$\boldsymbol{g(x) = (a^{-1}x - a^{-1}b)\%\ m}$$

Iz $a^{-1}x - a^{-1}b \equiv a^{-1}x - (a^{-1}b)\%\ m\ (mod\ m)$ sledi enakost njunih ostankov po modulu m.

Dokaz. .) Funkcija g je res afina transformacija: po definiciji inverza in po trditvi 3.4. je število a^{-1} proti modulu tuj ostanek, število $(a^{-1}b)\%\ m$ pa je ostanek po modulu m.
.) Pokažimo, da velja $g \circ f(x) = x$. Ker sta števili $g \circ f(x)$ in x ostanka po modulu m, zadostuje pokazati, da sta kongruentni:

$$\boldsymbol{g \circ f(x)} = (a^{-1}f(x) - a^{-1}b)\%\ m \equiv a^{-1}f(x) - a^{-1}b =$$

$$a^{-1}\big((ax + b)\ \%\ m\big) - a^{-1}b \equiv a^{-1}(ax + b) - a^{-1}b = a^{-1}ax + a^{-1}b - a^{-1}b \equiv \boldsymbol{x}\ (\boldsymbol{m})$$

Podobno pokažemo še enakost $f \circ g(x) = x$:

$$\boldsymbol{f \circ g(x)} = (ag(x) + b)\%\ m \equiv ag(x) + b \equiv ag(x) + b =$$

$$a\big((a^{-1}x - a^{-1}b)\%\ m\big) + b \equiv a(a^{-1}x - a^{-1}b) + b =$$

$$aa^{-1}x - aa^{-1}b + b \equiv x - b + b \equiv \boldsymbol{x}\ (\boldsymbol{m})$$ Konec dokaza.

Za potrebe enkripcije slovenskega čistopisa izberimo za modul število vseh črk slovenske abecede, to je $\boldsymbol{m = 25}$. Množica vseh ostankov po modulu 25 je $\boldsymbol{D} = \{\boldsymbol{0}, \dots, \boldsymbol{24}\}$. Vrednost Eulerjeve funkcije φ pri številu 25: $\boldsymbol{\varphi(25)} = \varphi(5^2) = 5^2 - 5^1 = \boldsymbol{20}$.

Tajni sistemi, ki temeljijo na modularni aritmetiki, črke abecede, v kateri je sporočilo napisano, najprej nadomestijo s števili, ki jih imenujmo *numerični ekvivalenti* posameznih črk. Za potrebe našega primera nadomestimo črke slovenske abecede s celimi števili od 0 do 24 z uporabo podatkovne strukture slovar :

```
slovar={'a': 0, 'b': 1, 'c': 2, 'č': 3, 'd': 4,
        'e': 5, 'f': 6, 'g': 7, 'h': 8, 'i': 9,
        'j': 10, 'k': 11, 'l': 12, 'm': 13, 'n': 14,
        'o': 15, 'p': 16, 'r': 17, 's': 18, 'š': 19,
        't': 20, 'u': 21, 'v': 22, 'z': 23, 'ž': 24}
```

S pomočjo modularne aritmetike izdelani tajni sistemi imajo svoj izvor pri Juliju Cezarju in njegovi t.i. Cezarjevi šifri, ta pa je poseben primer enkripcije, ki je matematično opisana z afino transformacijo $f(x) = (ax + b)\%m$.
S funkcijo enkripcija poljubni črki (označimo jo s spremenljivko s) slovenske abecede najprej priredimo njen numerični ekvivalent slovar[s], ki ga s funkcijo f preslikamo v ustrezni ostanek po modulu 25, tega pa z inverznim slovarjem v črko, ki je šifra črke s:

```
def enkripcija(s,a,b):
    return inverzni_slovar[(a*slovar[s]+b)%25]
```

Npr. pri $a = 2$ in $b = 3$ dobimo naslednji šifrant (funkcija izpis na datoteki enkripcija):

```
(a, č) (b, e) (c, g) (č, i) (d, k)
(e, m) (f, o) (g, r) (h, š) (i, u)
(j, z) (k, a) (l, c) (m, d) (n, f)
(o, h) (p, j) (r, l) (s, n) (š, p)
(t, s) (u, t) (v, v) (z, ž) (ž, b)
```

Preden sporočilo šifriramo, iz njega najprej odstranimo vsa ločila, velike črke spremenimo v male ter jih združimo v skupine npr. po pet. Z združevanjem v skupine po pet onemogočimo poskus razbijanja šifre na podlagi prepoznavanja značilnih besed (npr. pogosta dvočrkovna beseda zelo verjetno pomeni in, je, …). To naredimo s funkcijo pretvorba_v_po_pet na datoteki enkripcija. Ko s to funkcijo preoblikujemo sporočilo:

To sporočilo je velika skrivnost.

dobi obliko:

tospo ročil ojeve likas krivn ost

V naslednjem koraku vsako črko tega zapisa zamenjamo z njeno šifro glede na ustrezni ključ, ki je par izbranih števil a, b. To naredimo s funkcijo pretvorba_v_tajnopis na datoteki enkrip. Če izberemo npr. $a = 2$ in $b = 3$, dobimo tajnopis našega sporočila v obliki

shnjh lhiuc hzmvm cuačn aluvf hns

in ga pošljemo predvidenemu prejemniku. Prejemnik tega tajnopisa ima ključ, to je pozna števili a in b in zato na tajnopisu uporabi metodo dekripcije.
S funkcijo dekripcija poljubni črki tajnopisa (označimo jo s spremenljivko s) najprej priredimo njen numerični ekvivalent slovar[s], ki ga nato s funkcijo

$$f^{-1}(x) = \left(a^{-1}(x-b)\right)\%\, m \overset{T.2.8}{\cong} \left(a^{\varphi(m)-1}(x-b)\right)\%\, m = \left(a^{19}(x-b)\right)\%\, 25$$

preslikamo v ustrezni ostanek po modulu 25, tega pa z inverznim slovarjem v črko, ki je zašifrirana s črko s:

```
def dekripcija(s,a,b):
    return inverzni_slovar[((a**19)*(slovar[s]-b))%25]
```

Pretvorbo šifriranega besedila v čistopis opravimo s funkcijo, ki na vsaki črki tajnopisa pokliče metodo dekripcije. To naredimo s funkcijo pretvorba_v_čistopis na datoteki dekrip. Besedilo, ki ga ta funkcija vrne seveda vsebuje besede dolžine pet, vendar prejemniku iz njih ni težko ugotoviti vsebine sporočila.

Primer. Recimo, da pošiljatelj in prejemnik uporabljata tajni ključ $a = 23, b = 19$ in je prejemnik prejel datoteko 'šifrirano_besedilo_tajnopis.txt' z naslednjim sporočilom

dnhln jnmbt nžirb tnzšv nkbjš
pnhln snmžn šgbpi djšph gnjsš
obži

S funkcijo pretvorba_v_čistopis, ki jo pokliče na tej datoteki in na ključu, dobi datoteko 'šifrirano_besedilo_čistopis.txt', ki vsebuje odšifrirano sporočilo, ki ga lahko prebere:

tospo ročil ojebi lozak odira
nospo močjo afine trans forma
cije

8. KRIPTOANALIZA TAJNOPISA ŠIFRIRANEGA S POMOČJO AFINE TRANSFORMACIJE

V tem poglavju se ukvarjamo s kriptoanalizo tajnopisa, za katerega vemo, da je bila pri njegovi enkripciji uporabljena afina transformacija. To pomeni, da obstaja funkcija

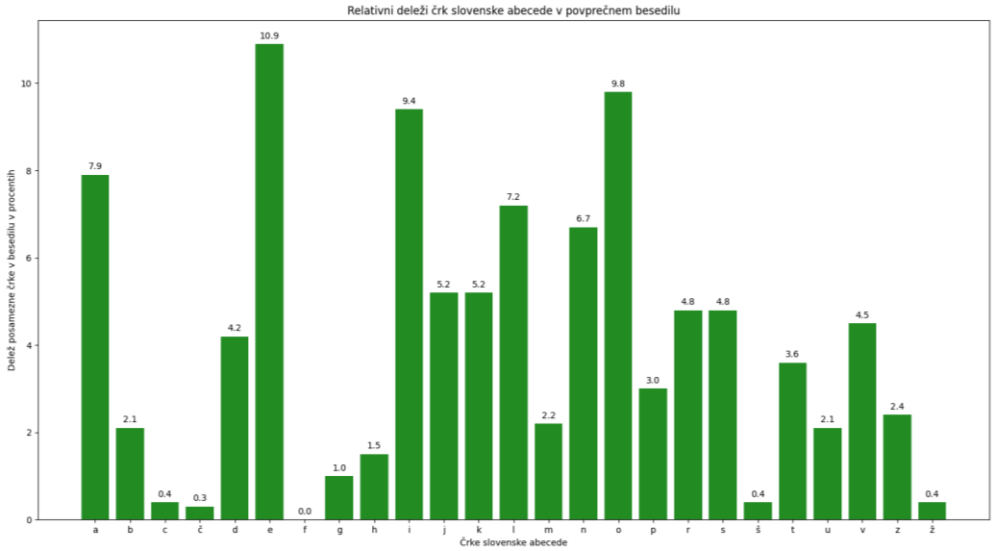
$$f(x) = (ax + b)\%\, m$$

s pomočjo katere je pošiljatelj sporočila šifriral svoje sporočilo. Na podlagi tajnopisa poskusimo torej poiskati ključ a, b. Pri tem si pomagajmo s primerjavo relativnih frekvenc posameznih črk v tajnopisu z relativnimi frekvencami črk v poljubnem slovenskem besedilu. V nadaljevanju pokažimo, da relativna frekvenca črk v slovenskem besedilu ni popolnoma poljubna, ampak se giblje znotraj nekih meja. Napišimo Pythonovo funkcijo delež_črk(ime_datoteke), ki sprejme ime dane tekstovne datoteke ter vrne slovar, katerega ključi so črke slovenske abecede, vrednosti pa relativne frekvence črk v tem tekstu. Ta funkcija se nahaja v datoteki relativna_frekvenca_črk.
Napišimo še funkcijo histogram(ime_datoteke), ki relativne frekvence črk prikaže s histogramom. Tudi ta funkcija se nahaja v datoteki relativna_frekvenca_črk. Za "običajno" slovensko" besedilo s pomočjo umetne inteligence chatGpt generirajmo npr. novico o odprtju nove trgovine v izmišljenem kraju.

```
V slikovitem slovenskem mestu Zeleni Vrh, obdanem z valovitimi hribi in bujnim
zelenjem, se je odprla nova trgovina, ki obljublja svež veter v lokalno skupnost.
"Zeleni Bazar", kot so jo poimenovali, stoji na osrednji lokaciji mesta in ponuja
širok nabor lokalno pridelanih izdelkov, od svežega sadja in zelenjave do rokodelskih
izdelkov in ekoloških prehrambnih izdelkov.
Lastnik trgovine, Jaka Novak, dolgoletni prebivalec Zelenega Vrha, je povedal,
da je bil njegov cilj ustvariti prostor, kjer bi lahko ljudje iz lokalne
```

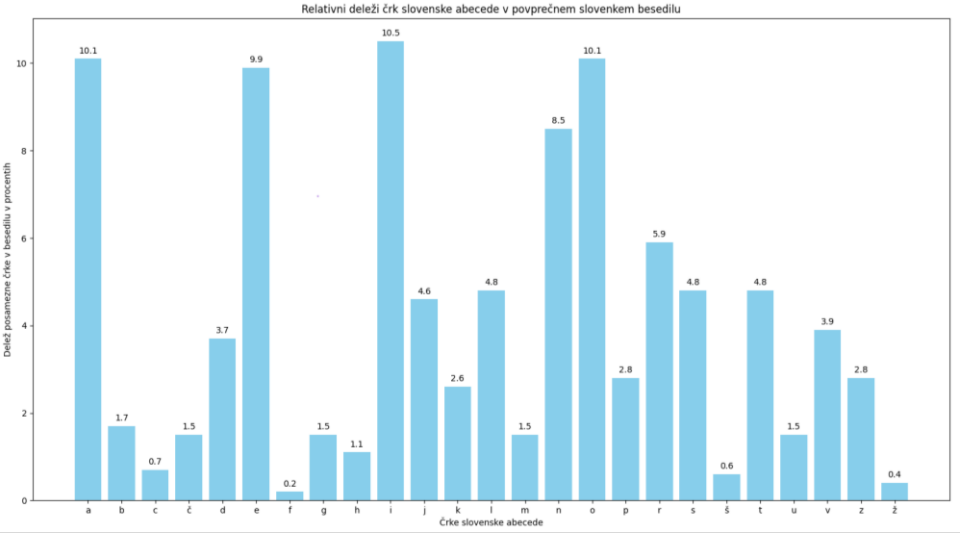
```
skupnosti kupovali zdrave, trajnostno pridelane izdelke in spoznali vrednost
podpore lokalni ekonomiji. "Želel sem odpreti trgovino, ki ne bi samo prodajala
izdelkov, ampak bi bila tudi središče skupnosti, kjer se ljudje lahko srečajo,
izmenjujejo ideje in skupaj rastejo," je dejal Novak.
```

Funkcija histogram vrne naslednji histogram, v katerem jasno vidimo deleže posameznih črk. Opazimo, da so najpogostejše črke tega besedila samoglasniki $'e', 'a', 'i', 'o'$, dokaj pogosti sta tudi $'n'$ in $'l'$.



Za primerjavo poglejmo še relativno frekvenco črk v kratkem opisu pasme border collie:

```
Border collie je pasma psa, znana po svoji inteligenci,
energičnosti in delovni sposobnosti. Izvira iz Velike
Britanije, kjer so jih prvotno uporabljali za čuvanje
in vodenje ovac. Ti psi so izjemno učljivi in željni
ugajati svojemu lastniku, kar jih dela idealne za
različne pasje športe, kot so agilnost, frizbi in poslušnost.
Videz border collieja je zelo raznolik, vendar so najpogostejše
barve črna z belimi oznakami, lahko pa najdemo tudi rdeče, modre
in trikolorne variante. Njihova dlaka je srednje dolga, kar
zahteva redno nego, da ostane čista in zdrava.
Imajo atletsko postavo, ki odraža njihovo energično naravo in
potrebo po redni telesni aktivnosti.
```



z primerov obeh povzamemo, da so v slovenskem besedilu črke z največjo relativno frekvenco $'a'$, $'e'$, $'i'$ in $'o'$. Zato bomo ključ a, b poiskali z uporabo afine transformacije $f(x) = (ax + b)\%\, m$ kot rešitev sistema dveh linearnih kongruenc z neznankama a in b:

$$numerični\ ekvivalent\ (šifra\ najpogostejše\ črke\ tajnopisa) \equiv a \cdot (numerični\ ekvivalent\ (šifra\ najpogostejše\ črke)) + b\ (mod\ m)$$

$$numerični\ ekvivalent\ (šifra\ druge\ ajpogostejše\ črke\ tajnopisa) \equiv a \cdot (numerični\ ekvivalent\ (šifra\ druge\ najpogostejše\ črke)) + b\ (mod\ m)$$

Analizirajmo tajnopis

```
nlzčm unllb thnss uzčrn lkulu
zhpol rhčtr lknsr jtžhr euščt
spčzž mlplp sgžol lrušt szlps
gžosp zžmlp lasgl orlpl nshil
elhrz llpčr čzčrn shrst hvlzh
agsšt sgžnz čuasm lkpsg žosar
čzčrf rs
```

za katerega vemo, da je bil zašifriran s pomočjo afine transformacije. S funkcijo prvih_nekaj, ki se nahaja v datoteki relativna_frekvenca_črk najprej naredimo slovar črk ter njihovih relativnih frekvenc v tem tajnopisu, nato pa v obliki seznama vrnemo dve črki, ki se v tem tajnopisu naj pogosteje pojavljata: najpogostejša je 'l', druga najpogostejša pa 's'. Glede na to, da sta najpogostejši črki slovenskega besedila vsebovani v množici $\{a, e, i, o\}$ imamo glede našega tajnopisa 12 možnosti, in sicer pare

$$(i,o), (o,i), (e,o), (o,e), (e,i), (i,e), (a,e), (e,a), (a,i), (i,a), (a,o), (o,a)$$

Za natanko enega med njimi velja, da je zašifriran s parom (l, s), torej je v najslabšem primeru potrebno preveriti 12 možnosti, kar pomeni rešiti 12 sistemov linearnih kongruenc z dvema neznankama. Recimo torej, da je najpogostejša črka poslanega sporočila $'i'$, druga pa $'o'$. Se pravi, da je $'i'$ zašifrirana z $'l'$, $'o'$ pa s $'s'$. Potem velja

$$slovar[l] = (a \cdot slovar[i] + b)\ \%m$$

$$slovar[s] = (a \cdot slovar[o] + b\ )\%m$$

Ključ (a, b) je potem rešitev sistema dveh linearnih kongruenc z dvema neznankama

$$12 \equiv 9 \cdot a + b\ (25)$$

$$18 \equiv 15 \cdot a + b\ (25)$$

Pri reševanju uporabimo trditev 5.5. Determinanta tega sistema je $D = \begin{vmatrix} 9 & 1 \\ 15 & 1 \end{vmatrix} = 9 - 15 = -6$. Njen inverz po modulu 25 izračunamo s funkcijo inverz na datoteki relativna_frekvenca_črke: $D^{-1} = 4$. Števili a in b nato izračunamo po formuli

$$a \equiv 4 \cdot (12 \cdot 1 - 1 \cdot 18) \equiv 1(25) \Longrightarrow a = 1$$
$$b \equiv 4 \cdot (9 \cdot 18 - 15 \cdot 12) \equiv 3(25) \Longrightarrow b = 3$$

Potem, ko smo dobili kandidata za ključ, in sicer (1,3), s funkcijo pretvorba_v_čistopis na datoteki dekrip poskusimo dešifrirati dani tajnopis. V primeru, da je njen rezultat smiselno besedilo v slovenskem jeziku, je ključ pošiljatelj res uporabil ključ (1,3). V nasprotnem primeru pregledamo še ostale pare.
V našem primeru funkcija pretvorba_v_čistopis vrne besedilo

```
kitaj skiiz rekoo stank ihsis
temli nearn ihkon gruen cspar
omatu jimim oduli inspr otimo
dulom tujim ivodi lnimi koefi
cient iiman atank oenor ešite
vdopr odukt asvoj ihmod ulovn
atanč no
```

,

v katerem prepoznamo znani izrek iz teorije števil. Ključ smo torej našli že v prvem poskusu. Na podoben način bi po potrebi preverili ostalih 11 možnosti. Ker predpostavka $d(D, 25) = 1$ trditve 5.5. v vseh primerih ni nujno izpolnjena, ima sistem lahko več rešitev (trditev 5.4) in tako dobimo še več možnih izidov, med katerimi pa lahko hitro prepoznamo smiselno besedilo v slovenskem jeziku. Šifriranje s pomočjo afine transformacije je zato za varno tajno komuniciranje preveč ranljivo za razbitje in zato v današnjem času neuporabno. Čudimo se uporabi t.i. Cezarjeve šifre iz časov tega rimskega vojskovodje, ki pomeni šifriranje s pomočjo preproste afine transformacije s ključem (1,3). Vsako črko so nadomestili s tisto črko abecede, ki se nahaja tri mesta od nje naprej. Zadnje tri črke pa so nadomestili z začetnimi tremi črkami.
V naslednjem poglavju poglejmo vrsto šifriranja, ki v smislu odpornosti proti razbitju predstavlja nasprotni pol šifriranju z afino transformacijo. Gre za v kriptologiji znamenito RSA šifriranje.

9. ŠIFRIRNI SISTEM RSA

Ena izmed najbolj znanih in široko uporabljenih metod za varno elektronsko komunikacijo je *RSA* šifrirni sistem. Sodi med t.i. *kriptografske sisteme z javnim ključem*. Pri šifriranju z javnim ključem se uporabljata dva različna ključa: *javni ključ*, ki ga lahko vidi kdorkoli, in *zasebni ključ*, ki ostane tajen. Varno šifriranje in dešifriranje sporočil po sistemu RSA temelji na dejstvu, da trenutno ni znan učinkovit algoritem za razcep produkta dveh velikih praštevil ($m = p * q$). Za razbitje RSA šifre je namreč potrebno tak produkt faktorizirati, to pa z znanimi algoritmi za faktorizacijo in s trenutno zmogljivostjo računalniške tehnologije ni mogoče v realnem časovnem okviru.

Opišimo delovanje RSA šifrirnega sistema:
Vsak udeleženec komunikacije, ki želi prejemati tajna sporočila preko RSA sistema od ostalih udeležencev, najprej generira par ključev – t.i. javni in zasebni ključ. To stori tako, da izbere dve tako veliki praštevili p in q, da njunega produkta $m = pq$ ne more razcepiti nihče, razen

njega samega. V praksi to pomeni vsaj šestomestna praštevila, če pa je potrebno tajnost ohraniti za zelo dolgo časovno obdobje, pa več kot tisočmestna praštevila. Udeleženec komunikacije nadalje izračuna vrednost Eulerjeve funkcije pri produktu $\varphi(m) = (p-1)(q-1)$ ter izbere poljuben ostanek e po modulu $\varphi(m)$, ki je tuj $\varphi(m)$. To naredi tako, da naključno izbere število med 1 in $\varphi(m)-1$, nato pa z razširjenim Evklidovim algoritmom preveri, ali je res tuje modulu $\varphi(m)$; če je, ga izbere za število e, če pa ni, postopek naključne izbire ponovi. Nekatere vrednosti števila e niso varne (npr. $e = 1$), zato je potrebno zavrniti tudi take naključne izbire.

```
#naključen izračun ostanka e po modulu fi(p*q), ker sta p in q lihi praštevili
def ključ(p,q):
    fi_od_m=(p-1)*(q-1)
    d=0
    while(d==0):
        e = random.randint(1,fi_od_m-1)
        d = ra.inverz(e,fi_od_m)
    return e,d
```

Ko je število e (število med 1 in $\varphi(m)-1$, ki je tuje $\varphi(m)$) izbrano, mu s pomočjo razširjenega Evklidovega algoritma izračuna inverz d, to je število med 1 in $\varphi(m)-1$, ki je tuje $\varphi(m)$ in zanj velja $ed \equiv 1\ (\varphi(m))$. Končno udeleženec javno objavi svoj $javni\ ključ\ (m,e)$, ki ga ostali udeleženci komunikacije uporabijo za šifriranje njemu namenjenih sporočil, sam pa varno shrani svoj $privatni\ ključ\ (m,d)$, ki ga potrebuje za dešifriranje sporočil, ki so bila zašifrirana z njegovim javnim ključem. Iz posameznikovega javnega ključa je v praksi nemogoče izračunati njegov zasebni ključ, brez zasebnega ključa pa ni mogoče dešifrirati šifriranih sporočil, zato RSA šifrirni sistem omogoča varno izmenjavo informacij tudi prek nezavarovanih kanalov. Vendar pa se računalniška moč povečuje in razvijajo vedno bolj učinkoviti algoritmi za faktorizacijo, zato dolžina ključev, ki zagotavljajo varnost pred potencialnimi kibernetskimi grožnjami, postopoma raste.
Opišimo postopek šifriranja: Ko nekdo želi poslati šifrirano sporočilo kateremu od udeležencev komunikacije, uporabi njegov javni ključ. Najprej črke sporočila zamenja z njihovimi numeričnimi ekvivalenti in jih združi v enako dolge bloke.
Nato vsak blok s funkcijo $f(x) \equiv x^e\ (m)\ , 0 \leq f(x) < m$ pretvori v 'tajni' blok, s čimer dobi tajnopis blokov, ki ga pošlje prejemniku. Ko prejemnik dobi šifrirano sporočilo, ga dešifrira z uporabo svojega zasebnega ključa (m,d). Vsak tajni blok s funkcijo $f^{-1}(x) \equiv x^d\ (m)\ ,\quad 0 \leq f^{-1}(x) < m$ pretvori nazaj v numerične ekvivalente črk, te pa s pomočjo inverznega slovarja v čistopis z vsebino, ki je namenjena le njemu. Metoda RSA temelji na naslednji trditvi:

Trditev 9.1. Naj bo $m = p \cdot q$ produkt dveh lihih praštevil, e pa proti $\varphi(m)$ tuje pozitivno celo število. Naj bo $\mathbb{Z}_m = \{0, \dots, m-1\}$ množica vseh ostankov po modulu m. Funkcija

$$f: \mathbb{Z}_m \longrightarrow \mathbb{Z}_m$$

$$f(x) \equiv x^e\ (m)\ ,\quad 0 \leq f(x) < m$$

je bijekcija.

Njena inverzna funkcija je $f^{-1}: \mathbb{Z}_m \longrightarrow \mathbb{Z}_m$

$$f^{-1}(x) \equiv x^d \ (m)\,, \quad 0 \le f^{-1}(x) < m\,,$$

pri čemer je d inverz od e glede na modul $\varphi(m)$.

Dokaz.
.) Ker sta števili e in $\varphi(m)$ tuji, obstaja inverz d števila e glede na modul $\varphi(m)$:

$$(e, \varphi(m)) = 1 \Longrightarrow \ \exists d: ed \equiv 1 \ (\varphi(m)) \ \Longrightarrow \exists d, k: ed = 1 + k \cdot \varphi(m)$$

.) Bijektivnost funkcije f pokažimo tako, da poiščemo njeno inverzno funkcijo. Uporabimo trditev, da je funkcija bijektivna natanko tedaj, ko ima inverzno funkcijo. V ta namen definirajmo funkcijo

$$g: \mathbb{Z}_m \longrightarrow \mathbb{Z}_m$$

$$g(x) \equiv x^d \ (m)\,, \quad 0 \le g(x) < m$$

in pokažimo, da je g inverzna funkcija funkcije f, to je, da velja: $f(g(x)) = x$ in $g(f(x)) = x$ za vsak x iz $\mathbb{Z}_m$:

$$f\big(g(x)\big) \ \equiv g(x)^e \ \equiv (x^d)^e \ \equiv x^{de} \ (m)$$

$$g\big(f(x)\big) \ \equiv f(x)^d \ \equiv (x^e)^d \ \equiv x^{de} \ (m)$$

.) Pokažimo, da velja $x^{de} \ \equiv x(m)$:

- Če je $\boldsymbol{x}$ **tuj** $\boldsymbol{m}$, po Eulerjevem izreku sledi $x^{\varphi(m)} \equiv 1(m)$ in od tod

$$\boldsymbol{x^{de}} = x^{1+k\cdot\varphi(m)} = \left(x^{\varphi(m)}\right)^k \cdot x \equiv \boldsymbol{x(m)}$$

- Če $\boldsymbol{x}$ **ni tuj** $\boldsymbol{m}$, potem imata od 1 večjega skupnega delitelja, torej velja $p|x$ ali $q|x$ oziroma $x \equiv 0(p)$ ali $x \equiv 0(q)$. Vse možnosti razdelimo na dva dela: $x \not\equiv 0(p)$ ali $x \equiv 0(p)$:a)
  a) $x \not\equiv 0(p) \Longrightarrow x$ tuj p in po Eulerjevem izreku sledi $x^{p-1} \equiv 1(p)$, od tod pa

$$\boldsymbol{x^{de}} = x^{1+k\cdot\varphi(m)} = x^{k(p-1)(q-1)} \cdot x^1 = (x^{p-1})^{(q-1)k} \cdot x \equiv 1^{(q-1)k} \cdot x \equiv \boldsymbol{x(p)}$$

  b) $x \equiv 0(p) \Longrightarrow \boldsymbol{x^{de}} \equiv 0^{de} = 0 \ \equiv \boldsymbol{x(p)}$

  V primeru, da x ni tuj m, torej velja $x^{de} \equiv x(p)$ in z enako utemeljitvijo tudi $x^{de} \equiv x(q)$. Ker sta modula p in q tuja, velja tudi $\boldsymbol{x^{de}} \equiv \boldsymbol{x(m)}$. Konec dokaza.

Zgled RSA šifriranja.
Privzemimo, da si je udeleženec komunikacije A, ki želi prejemati tajna sporočila preko RSA sistema za svoj enkripcijski modul izbral produkt praštevil $p = 43$ in $q = 59$, $m = 2537$, za javni ključ pa število $e = 13$, ki je res tuje $\varphi(m)$: $d(e, \varphi(m)) = d(13, 42 \cdot 58) = 1$. S pomočjo razširjenega Evklidovega algoritma je izračunal inverz $d = 937$ števila $e = 13$ po modulu $\varphi(m) = 2436$. Javni ključ $(2537{,}13)$ je nato javno objavil, da ga bo udeleženec

komunikacije B lahko uporabil za šifriranje njemu namenjenih sporočil, sam pa je shranil svoj zasebni ključ $(2537{,}937)$, da ga bo uporabil za dešifriranje sporočil, ki jih bo prejel od B –ja. Recimo, da mu B želi poslati tajno sporočilo, ki vsebuje znani Wilsonov izrek:

Predhodnik praštevilskega modula
modula in fakulteta tega predhodnika
sta kongruenta po tem modulu.

Oseba B ta čistopis najprej preoblikuje v dolgi niz, iz katerega odstrani vse znake razen črk ter velike črke zamenja z malimi (pretvori_čistopis_v_dolgi_niz v datoteki RSA_šifriranje):

```
predhodnikpraštevilskegamodulamodulainfakultetategapredhodnikastakongruentapote
mmodulu
```

V naslednjem koraku dolgi niz pretvori v nov niz, v katerem sta vsaki dve zaporedni črki dolgega niza zamenjani s svojima numeričnima ekvivalentoma. Pri tem dobi niz štirimestnih števil (blokov), ločenih s presledki (pretvori_dolgi_niz_v_bloke v datoteki RSA_šifriranje).

```
1617 0504 0815 0414 0911 1617 0019 2005 2209 1218 1105 0700 1315 0421 1200 1315
0421 1200 0914 0600 1121 1220 0520 0020 0507 0016 1705 0408 1504 1409 1100 1820
0011 1514 0717 2105 1420 0016 1520 0513 1315 0421 1221
```

Nato vsak blok s funkcijo $f(x) \equiv x^e \ (m)\ , 0 \leq f(x) < m$ pretvori v 'tajni' blok, s čimer dobi tajnopis blokov, ki ga pošlje prejemniku A (šifriraj_bloke v datoteki RSA_šifriranje).

```
1363 1072 0064 1771 0715 1363 2299 0825 2013 1788 2348 0906 0487 2221 2268 0487
2221 2268 2111 0279 1947 2185 2247 0793 1705 1841 1582 0385 1934 0822 1906 2089
0821 0918 0448 1182 2323 1841 0095 2439 0487 2221 0787
```

Prejemnik A dešifrira ta tajnopis s pomočjo svojega zasebnega ključa (m, d) tako, da vsak tajni blok x s funkcijo $f^{-1}(x) \equiv x^d \ (m)\ ,\ \ 0 \leq f^{-1}(x) < m$ pretvori nazaj v numerične ekvivalente črk, te pa s pomočjo inverznega slovarja v čistopis z vsebino, ki je namenjena le njemu. Pri tem dobi štiri ali manj mestna števila. Tista, ki niso štirimestna dopolni z začetnimi ničlami. Dobljeni niz štirimestnih blokov končno z inverznim slovarjem preoblikuje v čistopis (dekripcija v datoteki RSA_šifriranje):

```
predh odnik prašt evils kegam odula modul ainfa kulte tateg apred hodni kasta k
ongr uenta potem modul u
```

ZAKLJUČEK

Kongruence so matematični temelj za zagotavljanje varnosti zasebne komunikacije, elektronskega poslovanja ter drugih področij povezanih z informacijsko tehnologijo.
V nalogi je opisana ena ranljiva in ena robustna metoda šifriranja, izziv za nadaljevanje pa predstavljajo še številne druge zanimive metode šifriranja, ki so nastale tekom zgodovine ter izdelava čim lepših računalniških programov za njihovo predstavitev. Zahtevnejši izziv je poglobiti znanje o kongruencah potrebno za razumevanje nekaterih sodobnih še bolj učinkovitih metod šifriranja.

IV. VIRI IN LITERATURA

1. Grasselli, J. (2009). Elementarna teorija števil. Ljubljana: DMFA.
2. Grasselli, J. (2008). Enciklopedija števil. Ljubljana: DMFA.
3. Majcen, I. (2011). Smelo na Olimp. Ljubljana: DMFA.
4. Moškon, M. (2020). Osnove programiranja v jeziku Python za neračunalničarje. Ljubljana: UL FRI
5. Rosen, K. H. (2011). Elementary Number Theory. Addison-Wesley.
6. https://users.fmf.uni-lj.si/potocnik/Ucbeniki/Proseminar-izrocki.pdf
7. splet, chatGPT, moja raziskovalna naloga o kongruencah iz šolskega leta 2022/23

Na naslednjih straneh se nahajajo programi v Pythonu, izdelani za potrebe raziskovalne naloge.

datoteka: modularno_potenciranje.py

```
''' 0 < b < m. Funkcija seznam generira seznam ostankov potenc
b**(2**0),...,b**(2**k) po modulu m. Pri tem je k največje
število, pri katerem še velja 2**(2**k)<= N. '''

def seznam(b,N,m):
    s=[b]
    for i in range(1,len(bin(N))-2):
        s.append(s[i-1]**2%m)
    return s


# ostanek potence b**N po modulu m
def modularna_potenca(b,N,m):
    s_1=list(map(int,list(bin(N)[2:]))) # seznam števk v dvojiškem zapisu eksp. N
    s_2=seznam(b,N,m)[::-1] # obrne seznam ostankov
    rezultat=1 # vrednost b**N po modulu m
    for i in range(0,len(s_2)):
        if s_1[i]!=0:
            rezultat=(rezultat*s_1[i]*s_2[i])%m
    return rezultat
```

Povezava do programa: https://trinket.io/python3/ff87eef83e?showInstructions=true

datoteka: razširjeni_Evklidov_algoritem.py

```
''' funkcija evklidov_algoritem(x,y) vrne sezname deljencev, količnikov,
koeficientov alfa, beta, ki jih dobimo pri razširjenem Evklidovem algoritmu.'''

def evklidov_algoritem(x,y):
    if x < y:
        x, y = y, x
    r=[x,y]
    k=[0,x//y]
    al=[1,0]
    be=[0,1]
    i=1
    while (r[i-1]%r[i])!=0:
        i+=1
        r.append(r[i-2]-k[i-1]*r[i-1])
        al.append(al[i-2]-k[i-1]*al[i-1])
        be.append(be[i-2]-k[i-1]*be[i-1])
        k.append(r[i-1]//r[i])
    return r,k,al,be
```

```
'''Funkcija izpis izpiše celoten postopek razširjenega Evklidovega algoritma,
kot je na strani 20 v nalogi.'''

def izpis(r,k,al,be):
    print()
    print(r[0],'=',k[1],'*',r[1],'+',r[2],'    ','al_0','=',1,'    ','be_0','=',0)
    print(r[1],'=',k[2],'*',r[3],'+',r[4],'    ','al_1','=',0,'    ','be_1','=',1)
    i=2
    while (i<len(r)-2):
        print(r[i],'=',k[i+1],'*',r[i+1],'+',r[i+2],'    ',
              'al_',i,'=',al[i-2],'-',k[i-1],'*',al[i-1],'=',al[i],'    ',
              'be_',i,'=',be[i-2],'-',k[i-1],'*',be[i-1],'=',be[i])
        i+=1
    print(r[i],'=',k[i+1],'*',r[i+1],'+',0,'    ',
        'al_',i,'=',al[i-2],'-',k[i-1],'*',al[i-1],'=',al[i],'    ',
        'be_',i,'=',be[i-2],'-',k[i-1],'*',be[i-1],'=',be[i])
    i+=1
    print('                    ',
        'al_',i,'=',al[i-2],'-',k[i-1],'*',al[i-1],'=',al[i],'    ',
        'be_',i,'=',be[i-2],'-',k[i-1],'*',be[i-1],'=',be[i])

'''Rekurzivna funkcija gcd za razširjeni Evklidov algoritem sprejme
deljenca in delitelja ter vrne njun d, ter koeficienta α_n in β_n.'''
def gcd(u, v):
    if v == 0:
        return (u, 1, 0)
    else:
        g, x1, y1 = gcd(v, u % v)
        k = u // v
        return (g, y1, x1 - k* y1)

''' Funkcija inverz s pomočjo razširjenega Evklidovega algoritma vrne inverz
proti modulu tujega števila po tem modulu.'''
def inverz(a, m):
    g, x, y = gcd(a, m)
    if g != 1:  # Če števili nista tuji, inverz ne obstaja
        return 0
    else:
        return (x % m)
```

Povezava do programa: https://trinket.io/python3/08185786c7?showInstructions=true

datoteka: enkrip.py

```
''' oštevilčimo črke slovenske abecede '''
slovar={'a': 0, 'b': 1, 'c': 2, 'č': 3, 'd': 4,
            'e': 5, 'f': 6, 'g': 7, 'h': 8, 'i': 9,
            'j': 10, 'k': 11, 'l': 12, 'm': 13, 'n': 14,
            'o': 15, 'p': 16, 'r': 17, 's': 18, 'š': 19,
            't': 20, 'u': 21, 'v': 22, 'z': 23, 'ž': 24}

''' slovarju priredimo njegov inverzni slovar '''
inverzni_slovar = {y:x for x,y in slovar.items()}

''' enkripcija vrne šifrirano črko od črke s glede na a,b '''
def enkripcija(s,a,b):
    return inverzni_slovar[(a*slovar[s]+b)%25]

'''sifrant vrne slovar črk in njihovih šifer glede na izbrana a,b '''
def sifrant(slov,a,b):
    novi_slov={x:enkripcija(x,a,b) for x in slovar}
    return novi_slov

''' izpis šifranta '''
def izpis(slov,a,b):
    i=1
    for kljuc, vrednost in sifrant(slovar,a,b).items():
        print(f"({kljuc}, {vrednost})", end=" ")
        if i%5 ==0:
            print()
        i+=1
```

```
''' pretvorba čistopisa neke datoteke v besede iz petih črk,
odstranitev vseh ločil in zamenjava vseh velikih črk z
malimi. Nova datoteka se imenuje: ime_datoteke+_po_pet '''
def pretvorba_v_po_pet(ime_datoteke):
    f=open(ime_datoteke, encoding ='UTF-8') # ime_datoteke je niz
    nova_datoteka=ime_datoteke[:-4]+"_po_pet.txt" # niz, ki je ime nove datoteke
    g=open(nova_datoteka,'w',encoding='utf-8')
    niz=""
    n=1 # števec besed v vsaki vrstici datoteke
    for vrstica in f:
        vrstica=vrstica.strip()   # odstrani '₩n'
        if vrstica:  # preveri, da ni prazna
            for beseda in vrstica.split():# vrstica.split() je seznam besed
                for crka in beseda.lower():
                    if not crka in slovar: # če znak ni v slovarju
                        continue

                    if len(niz)<5:
                        niz+=crka
                    elif n<6: # v vsaki vrstici datoteke je 5 besed
                        g.write(niz+" ")
                        niz=crka
                        n+=1
                    else:
                        g.write("₩n"+niz+" ")# zapis v novo vrstico datoteke
                        niz=crka
                        n=2
    if n<6:
        g.write(niz+" ")
    else:
        g.write("₩n"+niz+" ")
    g.close()
```

```
''' pretvorba datoteke_po_pet v tajnopis '''
def pretvorba_v_tajnopis(ime_datoteke,a,b):
    f=open(ime_datoteke, encoding ='UTF-8')
    nova_datoteka=ime_datoteke[:-11]+"_tajnopis.txt" # zbrišemo 11 črk _po_pet.txt
    g=open(nova_datoteka,'w',encoding='utf-8')
    for vrstica in f:
        tajna_vrstica = ""
        for beseda in vrstica.split():
            tajna_beseda = ""
            for crka in beseda:
                tajna_crka=enkripcija(crka,a,b)
                tajna_beseda += tajna_crka
            tajna_vrstica += tajna_beseda+" "
        g.write(tajna_vrstica+'₩n')
    g.close()
```

datoteka: dekrip.py

```
''' oštevilčimo črke slovenske abecede '''
slovar={'a': 0, 'b': 1, 'c': 2, 'č': 3, 'd': 4,
            'e': 5, 'f': 6, 'g': 7, 'h': 8, 'i': 9,
            'j': 10, 'k': 11, 'l': 12, 'm': 13, 'n': 14,
            'o': 15, 'p': 16, 'r': 17, 's': 18, 'š': 19,
            't': 20, 'u': 21, 'v': 22, 'z': 23, 'ž': 24}

''' slovarju priredimo njegov inverzni slovar '''
inverzni_slovar = {y:x for x,y in slovar.items()}

''' dekripcija vrne dešifrirano črko od črke s glede na a,b '''
def dekripcija(s,a,b):
    return inverzni_slovar[((a**19)*(slovar[s]-b))%25]



# pretvorba tajnopisa_po_pet v čistopis_po_pet
def pretvorba_v_čistopis(ime_datoteke,a,b):
    f=open(ime_datoteke, encoding ='UTF-8')
    nova_datoteka=ime_datoteke[:-13]+"_čistopis.txt" # minus 13 črk _tajnopis.txt
    g=open(nova_datoteka,'w',encoding='utf-8')
    for vrstica in f:
        odkodirana_vrstica = ""
        for beseda in vrstica.split():
            odkodirana_beseda = ""
            for crka in beseda:
                odkodirana_crka=dekripcija(crka,a,b)
                odkodirana_beseda += odkodirana_crka
            odkodirana_vrstica += odkodirana_beseda+" "
        g.write(odkodirana_vrstica+'₩n')
    g.close()
    f.close()
```

datoteka: relativna_frekvenca_črk.py

```
import os
import matplotlib.pyplot as plt

''' Funkcija delež_črk vrne slovar, katerega ključi so črke slovenske abecede,
vrednosti pa deleži črk v slovenskem besedilu vsebovanem v datoteki ime_datoteke'''
def delež__črk (ime_datoteke):
    f=open(ime_datoteke, encoding ='UTF-8')
    delez = {'a': 0, 'b': 0, 'c': 0, 'č': 0, 'd': 0,
             'e': 0, 'f': 0, 'g': 0, 'h': 0, 'i': 0,
             'j': 0, 'k': 0, 'l': 0, 'm': 0, 'n': 0,
             'o': 0, 'p': 0, 'r': 0, 's': 0, 'š': 0,
             't': 0, 'u': 0, 'v': 0, 'z': 0, 'ž': 0}

    n=0 #števec črk besedila
    for vrstica in f:
        vrstica=vrstica.strip()   # odstrani '₩n'
        if vrstica:  # preveri, da ni prazna
            for beseda in vrstica.split():
                for crka in beseda.lower():
                    if not crka in delez:# crka je črka
                        continue
                    n += 1
                    delez[crka] += 1
                   # if crka not in delez:
                   #     delez[crka] = 0

    # na tej točki n pomeni število vseh črk besedila, delez['a'] pa število
    # pojavitev črke 'a' v besedilu
    for crka in delez:
        delez[crka]=round(delez[crka]*100/n,1)
    return delez
```

```
''' Narišemo graf slovarja, ki ga vrne prejšnja funkcija'''
def histogram(ime_datoteke):
    funkcija=delež_črk(ime_datoteke)
    originali = list(funkcija.keys())
    vrednosti = list(funkcija.values())
    # Ustvarjanje točk na grafu za vsak ključ-vrednost par
    # plt.plot(originali, vrednosti, marker='.', linestyle='-')
    plt.bar(originali, vrednosti, color='forestgreen')
    # Dodajanje vrednosti nad stolpce
    for i, vrednost in enumerate(vrednosti):
        plt.text(i, vrednost + 0.08, str(vrednost), ha='center', va='bottom')

    # Dodajanje naslova in oznak osi
    plt.title('Relativni deleži črk slovenske abecede v povprečnem besedilu')
    plt.xlabel('Črke slovenske abecede')
    plt.ylabel('Delež posamezne črke v besedilu v procentih')
    # Prikaz grafa
    plt.show()
```

```
'''funkcija prvih_nekaj (ime_datoteke,n) vrne seznam
največjih n vrednosti v slovarju '''
def prvih_nekaj (ime_datoteke,n):
    slovar=delež_črk (ime_datoteke)
    prvih_nekaj_kljuci = sorted(slovar, key=slovar.get, reverse=True)[:n]
    return(prvih_nekaj_kljuci)

''' Funkcija inverz(a) vrne inverz števila a po modulu 25 ali po modulu 5.
Če a ni tuj m, vrne 0.'''
def inverz(a,modul):
    b=a%modul
    if modul==25:
        inv=(b**19)%25
        if (b*inv)%25!=1:
            return 0
        else:
            return inv
    elif modul==5:
        inv=(b**3)%5
        if (b*inv)%5!=1:
            return 0
        else:
            return inv
    elif modul==5:
        inv=(b**3)%5
        if (b*inv)%5!=1:
            return 0
        else:
            return inv
```

datoteka: RSA_šifriranje

```
import random
import re
import razširjeni_Evklidov_algoritem as ra
import modularno_potenciranje as mp

'''naključen izračun ostanka e po modulu fi(p*q),ker sta p in q lihi praštevili'''
def ključ(p,q):
    fi_od_m=(p-1)*(q-1)
    d=0
    while(d==0):
        e = random.randint(1,fi_od_m-1)
        d = ra.inverz(e,fi_od_m)
    return e,d

'''numerični ekvivalenti črk slovenske abecede zapisani v obliki nizov'''
slovar={'a': '00', 'b': '01', 'c': '02', 'č': '03', 'd': '04',
            'e': '05', 'f': '06', 'g': '07', 'h': '08', 'i': '09',
            'j': '10', 'k': '11', 'l': '12', 'm': '13', 'n': '14',
            'o': '15', 'p': '16', 'r': '17', 's': '18', 'š': '19',
            't': '20', 'u': '21', 'v': '22', 'z': '23', 'ž': '24'}

'''slovarju priredimo njegov inverzni slovar'''
inverzni_slovar = {y:x for x,y in slovar.items()}

'''Pretvorba čistopisa v dolgi niz,pri čemer odstranimo vsa ločila in zamenjamo
   vse velike črke z malimi. Funkcija vrne ta dolgi niz. '''
def pretvori_čistopis_v_dolgi_niz(ime_datoteke):
    with open(ime_datoteke, 'r', encoding='utf-8') as datoteka:
        vsebina = datoteka.read() # Vsebina datoteke se prebere in shrani v
                                  # niz z imenom vsebina
        vsebina = vsebina.lower() # Spremeni vsebino datoteke v male črke
        vsebina = re.sub(r'₩W+', '', vsebina) # Odstrani ločila in presledke
    return vsebina
```

```
'''pretvorba čistopisa-niza v bloke iz štirih števk, ki so ločeni s presledki'''
def pretvori_dolgi_niz_v_bloke(niz):
    blok=""
    novi_niz=""
    for crka in niz:
        if blok=="":
            blok=slovar[crka]
        else:
            blok+=slovar[crka]
            novi_niz+=(blok+" ")
            blok=""
    if blok!="":
        novi_niz+=(blok+slovar['ž'])
    return novi_niz

'''pretvorba niza blokov iz štirih števk, ki so ločeni s presledki, v niz
šifriranih blokov iz štirih števk '''
def šifriraj_bloke(niz_bločnih_nizov, m, e):
    def f(x):
        return mp.modularna_potenca(x,e,m)
    # Razdeli niz na bloke štirih številk
    seznam_števil =[int(niz) for niz in niz_bločnih_nizov.split() ]
    # izračuna f na elementih seznama
    seznam_šifriranih_števil =[f(x) for x in seznam_števil]
    #elemente seznama pretvori v nize
    seznam_nizov=[str(x) for x in seznam_šifriranih_števil]
    # dopolni nize do dolžine 4 z ničlami na začetku
    seznam_4_nizov=[niz.zfill(4) if len(niz)<4 else niz for niz in seznam_nizov]
    niz = ' '.join(seznam_4_nizov)
    return niz
```

```
''' pretvori številske bloke v besede'''
def dekripcija(tajni_niz,m,d):
    def g(x):
        return mp.modularna_potenca(x,d,m)
    # Razdeli niz na bloke štirih številk
    seznam_števil =[int(niz) for niz in tajni_niz.split()]
    # izračuna g na elementih seznama
    seznam_odšifriranih_števil =[g(x) for x in seznam_števil]
    #elemente seznama pretvori v nize
    seznam_nizov=[str(x) for x in seznam_odšifriranih_števil]
    # dopolni nize do dolžine 4 z ničlami na začetku
    seznam_4_nizov=[niz.zfill(4) if len(niz)<4 else niz for niz in seznam_nizov]
    niz = ' '.join(seznam_4_nizov)

    seznam_2_nizov=[niz[i:i+2] for niz in seznam_4_nizov for i in
                    range(0, len(niz), 2)]
    seznam_črk=[inverzni_slovar[x] for x in seznam_2_nizov]
    dolgi_niz=''.join(seznam_črk)
    dolgi_niz_s_presledki_na_pet_crk=' '.join(dolgi_niz[i:i+5] for i in
                                        range(0, len(dolgi_niz), 5))
    return(dolgi_niz_s_presledki_na_pet_crk)
```