# Lietišķie algoritmi – 4. mājas darbs: Atrisinājumi

**1.uzdevums: RSA algoritms.** Bobs vēlas izveidot savu privātās/publiskās atslēgas pāri $(n, e)$, kur $n = p \cdot q$ ir reizinājums diviem mazākajiem 17-ciparu pirmskaitļiem (un $p < q$), bet kā publisko kāpinātāju viņš grib izvēlēties skaitli $e = 2^{16} + 1$.

(a) Kādi ir pirmskaitļi $p$ un $q$ un to reizinājums $n$? (Lielu pirmskaitļu pārbaudīšanai var izmantot esošas bibliotēkas, kas implementē Rabina-Millera varbūtisko pirmskaitļu pārbaudi, piemēram Python funkciju `sympy.isprime`.)

(b) Alise grib nosūtīt ziņojumu $m = 100$, izmantojot publisko RSA kriptoatslēgu - pāri $(n, e)$. Kāds ir viņas iešifrētais ziņojums?

(c) Cik reizināšanas darbības pēc $n$ moduļa Alisei jāveic, lai iešifrētu $m$?

(d) Kāda ir Boba izmantotā privātā kriptoatslēga $d$, kurai ir spēkā $e \cdot d \equiv 1$ pēc $\varphi(n)$ moduļa?

(e) Cik reizināšanas darbības pēc $n$ moduļa Bobam jāveic, lai atšifrētu Alises iešifrēto ziņojumu?

**(a)** Mazākais 17-ciparu skaitlis ir $10^{16}$ (viens vieninieks un sešpadsmit nulles). Pitona scenārijs $p, q$ (mazāko 17-ciparu pirmskaitļu atrašanai):

```
import sympy
primes = list()
n = 10**16
while len(primes) < 2:
    n += 1
    if sympy.isprime(n):
        primes.append(n)
p,q = primes[0],primes[1]
print('(p,q)=({},{})'.format(p,q))
```

Iegūstam, ka $p = 10000000000000061$ un $q = 10000000000000069$. To reizinājums

$$n = pq = 100000000000001300000000000004209.$$

**(b)** Alise nosūta skaitli $m^e \pmod n$. Mūsu gadījumā $m = 100$, $e = 2^{16} + 1$, $n = pq$. Lai kāpinātu pakāpē $e$, izmantojam nevis atkārtotu reizināšanu ciklā (kas prasītu ļoti daudz laika), bet gan sešpadsmit reizes kāpinām skaitli $m = 100$ kvadrātā un pēc tam vēlreiz piereizinām ar $m$. To panāk ar šādu scenārija turpinājumu:

```
n = p*q
m = 100
encrypt = m
# Raise to the power 2^16 - repeat 16 times
for _ in range(16):
    encrypt = (encrypt*encrypt) % n
# Raise power 2^16 to one higher: 2^16+1
encrypt = (encrypt*m) % n
print('Alice sends {}'.format(encrypt))
```

Iegūtais kriptoteksts ir šāds:

$$m^e \pmod n = 149016353215310820194680959321 67.$$

**(c)** Iepriekš aprakstītajā aprēķinā Alise veica $16 + 1 = 17$ reizināšanas darbības: 16 reizes reizināja skaitli pašu ar sevi (un atrada atlikumu), pēdējā solī vēlreiz piereizināja ar $m = 100$.

**(d)** Privāta kriptoatslēga, ko izmanto Bobs ir $d = e^{-1} \pmod{(p-1)(q-1)}$. Šeit $e = 2^{16} + 1$; tātad jāatrod skaitlis $d$, kuru piereizinot ar $e$ iegūsim atlikumu 1, dalot ar $(p-1)(q-1)$.

```
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

e = 2**16 + 1
priv = (p-1)*(q-1)
d = modinv(e,priv)
print('d = {}'.format(d))
```

Iegūstam šādu atslēgas vērtību:

$$d = 936173459267296112595938172368 33$$

Atšifrēšanas pareizuma pārbaudei var izmantot Pitona iebūvēto funkciju `pow(x,y,m)`, kas kāpina $x$ pakāpē $y$ pēc $m$ moduļa:

```
decrypt = pow(encrypt,d,n)
print('decrypted = {}'.format(decrypt))
```

**(e)** Iepriekšējā solī atšifrēšana (kāpināšana lielajā pakāpē $d$) notika ar iebūvētu Pitona funkciju `pow(x,y,m)`. Ja mums pašiem tā būtu efektīvi jāuzprogrammē, kāpinātāju $d$ pārveidojam divnieku skaitīšanas sistēmā (var izmantot Pitona iebūvēto funkciju `bin(...)`, bet bināro pierakstu var ātri iegūt arī, atkārtoti dalot ar divnieku):

```
print('d_bin = {}'.format(bin(d)))
```

Iegūtais skaitlis ir (skaitļa pierakstā ir 107 binārie cipari, no tiem 55 ir vieninieki).

$$d = (1001001110 \ldots 0101100001)_2.$$

Kāpināšanai šādā pakāpē, izmantojot *Exponentiation by Squaring* metodi, vajag $107 - 1 = 106$ reizes kāpināt kvadrātā un arī $55 - 1 = 54$ reizes piereizināt kāpināmo skaitli. Pavisam tātad 160 reizināšanas.

**Piezīme.** Ievērosim, ka Alisei, kāpinot pakāpē $e = 2^{16} + 1$, vajadzēja tikai 17 reizināšanas, bet atšifrēšanai jeb kāpināšanai pakāpē $d$ vajag 160 reizināšanas. Asimetriskiem algoritmiem iešifrēšana un atšifrēšana var būtiski atšķirties laika sarežģītības ziņā. Mūsu gadījumā publiskā atslēga $e = 2^{16} + 1$ tika izraudzīta tā, lai iešifrēt varētu īpaši ātri.

**2.uzdevums: Afīnās mērogošanas metode LP uzdevumā.** Dots LP uzdevums: Maksimizēt $2x_1 + 3x_2$, kur

$$\begin{cases} x_1 - 2x_2 \leq 4, \\ x_1 + x_2 \leq 18, \\ x_2 \leq 10, \\ x_1, x_2 \geq 0. \end{cases}$$

Aprēķināt un attēlot koordinātu plaknē šī LP uzdevuma pirmos 2 tuvinājumus $X(1)$ un $X(2)$ kā 2-dimensionālus vektorus, izmantojot afīnās skalēšanas metodi.
Izvēlētais sākumpunkts $X(0) = (5, 5)$ (t.i. sākumpunkta koordinātes ir $x_1 = 5$ un $x_2 = 5$). Gan $X(1)$, gan $X(2)$ abas koordinātes atbildē noapaļot līdz 5 cipariem aiz komata. Soļa garums abos gadījumos: $\beta = 0.96$. (Vektoru un matricu operācijām var izmantot Pitona bibliotēkas.)

**3.uzdevums: KMP un BM algoritmi.** Virknē `947892879487` meklējam apakšstringu `9487`.
(a) Atrast Knuta-Morisa-Prata algoritmam vajadzīgo prefiksu funkciju $\pi$.
(b) Atrast Bojera-Mūra algoritmam vajadzīgo labo sufiksu tabulu un sliktā simbola tabulu.

**(a)** Izveidojam tabuliņu prefiksu funkcijai:

| $j$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $\pi(j)$ | 0 | 0 | 0 | 0 |

**(b)** Izveidojam sliktā simbola tabulu – atzīmējam tabuliņā pēdējo indeksu, ar kuru simbols ietilpst paraugā $P =$ `9487` vai $-1$, ja simbola tur nav vispār. Šeit $*$ apzīmē visus citus simbolus, izņemot tabulā ierakstītos.

| $x$ | `4` | `7` | `8` | `9` | $*$ |
|---|---|---|---|---|---|
| $\lambda(x)$ | 1 | 3 | 2 | 0 | -1 |

**4.uzdevums: Bojera-Mūra algoritms.** Dots teksts $T =$ `abcabbcabcbcabababababcbcab` un meklējamais paraugs $P =$ `abcbcab`.
(a) Uzrakstīt Bojera-Mūra algoritmā lietotās tabulas apakšstringam $P$.
(b) Nodemonstrēt Bojera-Mūra darbību pa soļiem, meklējot paraugu $P$ dotajā tekstā $T$.

**(a)** Sliktā simbola tabula (tikai 3 burti, jo citu tekstā un paraugā nav):

| $x$ | `a` | `b` | `c` |
|---|---|---|---|
| $\lambda(x)$ | 5 | 6 | 4 |

Labā sufiksa tabulu aprēķināsim, vispirms atrodot prefiksu funkcijas $\pi$ un $\pi'$ paraugam $P =$ `abcbcab` un reversajam paraugam $P' =$ `bacbcba` (no otra gala uzrakstītam paraugam).

| $\ell$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $\pi(\ell)$ | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| $\pi'(\ell)$ | 0 | 0 | 0 | 1 | 0 | 1 | 2 |
| $\ell - \pi'(\ell)$ | 1 | 2 | 3 | 3 | 5 | 5 | 5 |

Pēc tam pielabojam šīs vērtības:
(a) Vispirms piešķiram sufiksa funkcijai sākumvērtības $\gamma^*(j)$ (kuras vēlāk pielabosim): $\gamma(j) = m - \pi[m]$ (kur $m = 7$ ir parauga garums).
(b) Apstaigājam visus indeksus $j_\ell = m - \pi'[\ell]$ ($\ell \in \{1, \ldots, m\}$) - mūsu gadījumā tie ir $j_\ell = \{7, 7, 7, 6, 7, 6, 5\}$.
(c) Katram $j_\ell$ atrodam ja $\ell - \pi'[\ell] > \gamma[j_\ell]$, aizstājam $\gamma[j_\ell]$ ar $\ell - \pi'[\ell]$.
(d) Ja kādam $\ell$ izrādās, ka $\ell - \pi'[\ell] > \gamma[j_\ell]$, aizstāj $\gamma[j_\ell]$ ar $\ell - \pi'[\ell]$.

| $j$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **(a)** $\gamma^*(j) = m - \pi(m)$ | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| **(b)** $j_\ell$ | $j_1, j_2, j_3, j_5$ | $j_4, j_6$ | $j_7$ | | | | | |
| **(c)** $\ell - \pi'[\ell]$ | $1, 2, 3, 5$ | $3, 5$ | 5 | | | | | |
| **(d)** $\gamma(j)$ | 1 | 3 | 5 | 5 | 5 | 5 | 5 | 5 |

**5.uzdevums: I-iespēja (atzīmei 10).** Vispārināt Rabina-Karpa algoritmu, lai atrastu kvadrātveida paraugu $m \times m$ divdimensionālā simbolu masīvā ar izmēru $n \times n$, kur $n > m$. (Meklējamo paraugu var bīdīt pa horizontāli un vertikāli, bet to nedrīkst pagriezt.)

(a) Aprakstīt algoritmu (ar skaidri definētiem soļiem), kas atrod visas parauga atrašanās vietas divdimensionālajā $n \times n$ masīvā kā pozīciju pārus $(s_x, s_y)$, kur $s_x$ ir nobīde pa horizontāli un $s_y$ ir nobīde pa vertikāli.

(b) Pamatot, ka Jūsu algoritms atrod izvada visas vietas, kur paraugs atrodams.

(c) Atrast mazāko laika sarežģītību visu atrašanās vietu izvadei.

**(a)** Apzīmējam meklējamā kvadrātveida parauga burtus ar divdimensiju masīva elementiem: $P[i][j]$, kur $0 \leq i, j < m$ un $P[i][j]$ ir simbols, kas atrodas $i$-tajā rindiņā un $j$-tajā kolonnā. Gan rindiņas, gan kolonnas numurējam ar skaitļiem no 0 līdz $m-1$. Divdimensiju tekstā, kurā veicam meklēšanu, tieši tāpat apzīmējam burtus ar $T[i][j]$, kur $i, j$ mainās no 0 līdz $n-1$.

- Izvēlamies konstantes $d$ un $q$ - polinomā ievietojamo vērtību $x = d$ un atlikumu moduli $q$.
- Katrā parauga $P$ rindiņā izrēķina polinoma atlikumu, ja polinomā ievieto dalot ar konstantu moduli $q$:

$$R[i] = P[i][0] \cdot d^{n-1} + P[i][1] \cdot d^{n-2} + P[i][2] \cdot d^{n-3} + \ldots + P[i][m-2] \cdot d^1 + P[i][m-1] =$$

$$= (\ldots(\ldots((P[i][0] \cdot d + P[i][1]) \cdot d + P[i][2]) \cdot d + P[i][3])\ldots) \cdot d + P[i][m-1].$$

  Šeit izmantojam Hornera shēmu - katrā solī pieskaitām kārtējo $P$ vērtību un piereizinām ar $d$. Turklāt gan saskaitīšanu, gan reizināšanu šajos polinomu vērtību aprēķinos veicam pēc $q$ moduļa; tādējādi izvairāmies no ļoti lieliem skaitļiem.
- No vērtībām $R[i]$ būvē nākamo polinomu (arī to rēķina ar Hornera shēmu tāpat kā iepriekš):

$$^* = R[0] \cdot d^{n-1} + R[1] \cdot d^{n-2} + R[2] \cdot d^{n-3} + \ldots + R[m-2] \cdot d^1 + R[m-1]$$

Katrā matricas $n \times n$ rindiņā aprēķinām