l9-hnp-sage

April 23, 2025

1 Hidden number problem

1.0.1 Globálne parametre

[17]:
```
bits = 20   # počet bitov n
lbits = 4   # počet bitov, ktoré budeme poznať v každej rovnici
R = 2^(bits - lbits) # hranica pre r hodnoty
m = 12     # počet vzoriek pre HNP
n = random_prime(2^bits, lbound = 2^(bits-1))
print(f'n = {n}')
alpha = randint(0, n-1)
print(f'alpha = {alpha}')
```

```
n = 921931
alpha = 826483
```

Funkcie pre generovanie vzoriek, vytvorenie mriežky pre CVP a SVP

[18]:
```
def gen_samples(m):
    a, t, r = [], [], []
    for _ in range(m):
        ti = randint(1, n-1)
        ai = (ti * alpha) % n
        ri = ai % R
        ai = ai - ri
        a.append(ai)
        t.append(ti)
        r.append(ri)
    return (a, t, r)

def make_latticeCVP(m):
    rows = []
    for i in range(0, m):
        row = vector(QQ, m + 1)
        row.set(i, QQ(n))
        rows.append(row)
    rows.append(t + [1/n])
    return matrix(QQ, rows)
```

```
def make_latticeSVP(tmpR, m):
    rows = []
    for i in range(0, m):
        row = vector(QQ, m + 2)
        row.set(i, QQ(n))
        rows.append(row)
    rows.append(t + [tmpR/n, 0])
    rows.append(-vector(QQ, a + [0, m*tmpR]))
    return matrix(QQ, rows)

def approximate_closest_vector(basis, v):
    """Returns an approximate CVP solution using Babai's nearest plane␣
 ↪algorithm.
       implementácia prebraná z https://kel.bz/post/hnp/
    """
    BL = basis.LLL()
    G, _ = BL.gram_schmidt()
    _, nn = BL.dimensions()
    small = vector(ZZ, v)
    for i in reversed(range(nn)):
        c = QQ(small * G[i]) / QQ(G[i] * G[i])
        c = c.round()
        small -= BL[i] * c
    return (v - small).coefficients()
```

1.0.2 Riešenie HNP pomocou CVP

[19]:
```
a, t, r = gen_samples(m)
```

[20]:
```
print(f'alpha = {alpha}')
M = make_latticeCVP(m)
for mi in M:
    print(mi)
v = approximate_closest_vector(M, vector(QQ,a + [0]))
print('approx. CVP:\n', v)
calc_alpha = (v[-1] * n) % n
print(f'calculated alpha = {calc_alpha}')
print(calc_alpha == alpha)
```

```
alpha = 826483
(921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0)
```

```
(0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0)
(626411, 776762, 390425, 822230, 849378, 568115, 10666, 158044, 710982, 509236,
298349, 370625, 1/921931)
approx. CVP:
 [314015, 389713, 87551, 89266, 415003, 697038, 685387, 573241, 609043, 488454,
711307, 921332, -95448/921931]
calculated alpha = 826483
True
```

Experiment s rôznym počtom rovníc

[21]:
```
for mm in range(2,16):
    a, t, r = gen_samples(mm)
    M = make_latticeCVP(mm)
    v = approximate_closest_vector(M, vector(QQ,a + [0]))
    #print(M)
    #print(v)
    calc_alpha = (v[-1] * n) % n
    print(f'mm = {mm:2}\tsuccess? {calc_alpha == alpha}')
```

```
mm =  2 success? False
mm =  3 success? False
mm =  4 success? False
mm =  5 success? False
mm =  6 success? False
mm =  7 success? True
mm =  8 success? True
mm =  9 success? True
mm = 10 success? True
mm = 11 success? True
mm = 12 success? True
mm = 13 success? True
mm = 14 success? True
mm = 15 success? True
```

1.0.3 Transformácia na SVP

Skúmame druhý najkratší vektor získaný pomocou LLL algoritmu

[22]:
```
print(f'alpha = {alpha}\nr = {r}')
a, t, r = gen_samples(m)
M = make_latticeSVP(R, m)
print('M ' + '-'*30)
for mi in M:
    print(mi)
B = M.LLL()
```

```
print('B ' + '-'*30)
for bi in B:
    print(bi)
calc_alpha = B[1][m]*n/R % n
print(f'calculated alpha = {calc_alpha}')
print(calc_alpha == alpha)
```

```
alpha = 826483
r = [43776, 44377, 24111, 24526, 42614, 353, 29344, 56767, 35676, 37232, 38226,
38394, 16299, 20309, 33793]
M ------------------------------
(921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0)
(875185, 740322, 515148, 453531, 658482, 595526, 670899, 693158, 641480, 477268,
6075, 564081, 65536/921931, 0)
(-524288, -65536, -393216, -589824, 0, -786432, -393216, -851968, -196608,
-131072, 0, -327680, 0, -786432)
B ------------------------------
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 0)
(-32177, 49158, 65745, 155687, -45042, 104190, -56434, 105719, -132202, 43773,
38084, 74755, -24008327168/921931, 0)
(-36405, -35467, 81690, 83226, -11163, 152807, -81900, -194672, 5061, -18810,
-85616, 50396, -25998786560/921931, 0)
(-48912, 129593, -3065, -80317, 21241, 9446, 114463, 72197, -103285, -163163,
118891, 125525, -4335009792/921931, 0)
(-53625, 53359, 42838, 1796, 114610, 100491, 70812, 34991, 26828, -224857,
-113198, -54920, 29915217920/921931, 0)
(-8277, -136230, 43644, 123081, 112333, 132899, 136364, -70623, 87076, -76071,
37818, -8219, 5192024064/921931, 0)
(114560, -71613, 147112, 42754, -51861, -222675, 18410, 31152, -116970, 75146,
-2818, 105292, 9975431168/921931, 0)
(178085, -163190, 110549, -72842, -116712, -29680, 62177, 88986, 78186, -69808,
-24741, -113320, 7071072256/921931, 0)
(-193156, -5667, 63819, 87771, -87642, 106817, -9473, 191092, 165351, -5215,
47822, -20827, -1362034688/921931, 0)
(82967, -234053, -47521, 42729, -54483, -213294, -101667, 105097, 92093, -60835,
59330, 75257, -27648196608/921931, 0)
(153008, -122816, 3858, -231580, 50868, 52179, -79708, -27409, -15877, -77067,
```

```
-148749, 202950, -2448228352/921931, 0)
(-132314, -56528, -4710, -145983, 214621, 89714, -94818, -43963, -182782, 41036,
66895, -32700, 20936261632/921931, 0)
(137681, 243104, 140814, 28415, 29986, 31955, -150656, -150166, 52806, 40369,
166884, 26334, 10056826880/921931, 0)
(63811, 3634, 28434, 53393, 12127, 25656, 44361, 59532, 23855, 47436, 47999,
39432, -6255280128/921931, -786432)
calculated alpha = 555593
False
```

Ak by sme sa neobmedzili len na druhý najkratší vektor ...

[23]:
```
for i, bi in enumerate(B):
    print(i+1, bi[m]*n/R % n)
```

```
1 0
2 555593
3 525221
4 855784
5 456470
6 79224
7 152213
8 107896
9 901148
10 500053
11 884574
12 319462
13 153455
14 826483
```

A takto to dopadne pre rôzne hodnoty R:

[24]:
```
for useR in [2^i for i in range(bits)]:
    #v = vector(QQ, r + [alpha*useR/n, useR])
    #print(f'v = {v}, v.norm() = {v.norm().n(12)}')
    M = make_latticeSVP(useR, m)
    B = M.LLL()
    print(f'useR = {useR}\t{B[1][m]*n/useR % n}') #\tB[1] = {B[1]} {B[1].norm().
 ↪n(12)}')
    #print(B[0][m]*n/useR)
```

```
useR = 1        0
useR = 2        0
useR = 4        0
useR = 8        0
useR = 16       0
useR = 32       95448
useR = 64       95448
useR = 128      95448
useR = 256      95448
```

```
useR = 512      95448
useR = 1024     826483
useR = 2048     826483
useR = 4096     826483
useR = 8192     826483
useR = 16384    826483
useR = 32768    555593
useR = 65536    555593
useR = 131072   555593
useR = 262144   555593
useR = 524288   2285
```

Pozorovanie: Nezaujímajú nás vektory, ktoré majú nulovú poslednú súradnicu.

Takže zopakujeme výpočet pre SVP, ale z LLL bázy zoberieme najkratší vektor s nenulovou poslednou súradnicou.

[25]:
```
print(f'alpha = {alpha}\nr = {r}')
a, t, r = gen_samples(m)
M = make_latticeSVP(R, m)
print('M ' + '-'*30)
for mi in M:
    print(mi)
B = M.LLL()
print('B ' + '-'*30)
found = -1
for i, bi in enumerate(B):
    print(bi)
    if bi[-1] != 0 and found == -1:
        found = i
calc_alpha = B[found][m]*n/R % n
print(f'calculated alpha = {calc_alpha}')
print(calc_alpha == alpha)
```

```
alpha = 826483
r = [63811, 3634, 28434, 53393, 12127, 25656, 44361, 59532, 23855, 47436, 47999,
39432]
M ------------------------------
(921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0, 0)
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0)
```

```
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 921931, 0, 0)
(56680, 253799, 741191, 49735, 355960, 754785, 309055, 895896, 513591, 818777,
550509, 626979, 65536/921931, 0)
(-786432, 0, -65536, -786432, -196608, -655360, -327680, -327680, -589824,
-524288, -458752, -458752, 0, -786432)
B ------------------------------
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -65536, 0)
(9169, -47119, -124554, -177463, -142158, -76909, -34667, 73619, 16898, -22445,
70333, -71006, -13195739136/921931, 0)
(-74601, 70463, 147775, 96626, -28036, 22758, -4606, 97621, 21180, -90875,
17775, -210883, 6279593984/921931, 0)
(131929, 3543, 2537, 73392, -125275, 149241, 128514, -128508, -63334, 47570,
16058, 122970, -4497211392/921931, 0)
(-23859, 51120, 154481, 14442, -186924, 22961, -187105, -77444, -37938, -39630,
21701, -55544, -8560902144/921931, 0)
(108255, 41747, 109258, 63842, -120405, -19789, -75881, 25991, 199609, -36151,
157725, -74071, -5463015424/921931, 0)
(-149006, -67111, 145282, 53215, -22310, -28322, -2613, -22481, -17608, -123811,
-166256, 135968, 18319736832/921931, 0)
(-63949, 34425, -135380, 30826, -187556, -133053, -3454, -107338, 167603,
-102974, 38149, 34013, 25880821760/921931, 0)
(-163700, 91655, -122763, 100016, -126302, 126201, -149260, 53397, -75054,
-55364, 42786, -77224, 13708296192/921931, 0)
(88438, -29861, -106992, 97283, -43817, 4460, -117491, 77628, 72694, -132129,
-49990, -161319, -29013704704/921931, 0)
(22029, 226146, -30883, 98299, -90673, -67337, -15620, 125292, -23049, -172070,
37461, 59862, 28142927872/921931, 0)
(-29219, 1745, 68124, -13033, -37761, 29008, -10084, -219420, 40365, -307040,
84151, 7482, 6533349376/921931, 0)
(-126544, -230001, 102283, 25592, -47804, -305818, -83618, 2823, -248397,
-134957, 112123, -105449, 18189713408/921931, 0)
(33967, 52004, 33112, 51938, 56455, 2024, 16887, 56955, 13471, 17555, 15561,
34728, -6255280128/921931, -786432)
calculated alpha = 826483
True
```

[26]:
```python
for useR in [2^i for i in range(bits)]:
    M = make_latticeSVP(useR, m)
    B = M.LLL()
    valB = 0
    for bi in B:
        if bi[-1] != 0:
            valB = bi[m]
            break
    print(f'useR = {useR}\t{valB*n/useR % n}')
```

```
useR = 1        0
useR = 2        0
```

```
useR = 4         0
useR = 8         0
useR = 16        0
useR = 32        0
useR = 64        95448
useR = 128       95448
useR = 256       95448
useR = 512       826483
useR = 1024      826483
useR = 2048      826483
useR = 4096      826483
useR = 8192      826483
useR = 16384     826483
useR = 32768     826483
useR = 65536     826483
useR = 131072    826483
useR = 262144    826483
useR = 524288    826483
```

[ ]: