Practice 1: Getting familiar with FpyLLL

The goal of this lab is to familiarize yourself with basic routines implemented in the FPyLLL library [1]. Documentation is available at https://readthedocs.org/projects/fpylll/downloads/pdf/latest/

1 LLL

1. Generating a random q-ary matrix.

```
from fpylll import *
import copy
A = IntegerMatrix.random(30, "qary", k=15, bits=30)
print(A)
dim = A.nrows
print('dimension:', dim)
```

2. GSO object, norm computation, and LLL reduction

```
M = GSO.Mat(A, float_type="d") #float_type \in {'d', 'dd', 'qd','mpfr'}
print(M.get_r(0,0))
print(M.d) # lattice dimension
print(M.B) # input basis
M.update_gso() # compute all interesting matrices (R-factor, mu, ...)
print(M.get_r(1,1))
print(A[0].norm()**2) # first basis norm
print('before:', [M.get_r(i,i) for i in range(dim)]) # all r_ii

L = LLL.Reduction(M, delta=0.99, eta=0.501) # creating an LLL object
L() # run LLL
print('after:', [M.get_r(i,i) for i in range(dim)])
print(A)
```

3. Transformation matrix

```
A = IntegerMatrix.random(100, "qary", k=50, bits=30)
dim = A.nrows
Ac = copy.deepcopy(A) # copy of A
U = IntegerMatrix.identity(dim)
UinvT = IntegerMatrix.identity(dim)

M_transf = GSO.Mat(Ac, float_type="qd", U = U, UinvT = UinvT) #U -- transformation
                                                              matrix mapping A to LLL-reduced Ac
print(M_transf.inverse_transform_enabled) # LLL computes U

L = LLL.Reduction(M_transf, delta=0.99, eta=0.501)
L()
print(Ac[1])
print((U*A)[1])
```

2 Enumeration

```
A = IntegerMatrix.random(45, "qary", k=25, bits=30)
M = GSO.Mat(A)
L = LLL.Reduction(M, delta=0.99, eta=0.501)
L()
M.update_gso()

enum = Enumeration(M, strategy=EvaluatorStrategy.BEST_N_SOLUTIONS, sub_solutions=True) #
                                                initiate the object Enumeration
res = enum.enumerate(0, 45, 0.9*M.get_r(0, 0), 0) # launch enumeration
print(([int(res[0][1][i]) for i in range(len(res[0][1]))]))
#for a,b in enum.sub_solutions:
#    print(a, b)

print(enum.get_nodes()) # number of visited nodes
```

3 BKZ

Main source of the code: https://www.maths.ox.ac.uk/system/files/attachments/lab-02.pdf

1. Part 1: Launching BKZ

```
from fpylll import *
from fpylll.algorithms.bkz import BKZReduction
from math import log

from fpylll.algorithms.bkz2 import BKZReduction as BKZ2
import matplotlib.pyplot as plt

A = IntegerMatrix.random(60, "qary", k=30, bits=30)

bkz = BKZReduction(A) #instantiation class BKZReduction
print('before BKZ:', log(A[0].norm(),2))
bkz(BKZ.EasyParam(20, max_loops=8)) #block size  15 ; maxloops = number of bkz
                                                tours
print('after BKZ:', log(A[0].norm(),2))
bkz(BKZ.EasyParam(15, max_loops=8), tracer=True)
print(bkz.trace.get(("tour", 1)).report()) #for tracer=True



k = 40
flags = BKZ.AUTO_ABORT
#print('flags:', flags, BKZ.AUTO_ABORT, BKZ.MAX_LOOPS,BKZ.VERBOSE)
par = BKZ.Param(k, strategies=BKZ.DEFAULT_STRATEGY, max_loops=8, flags=flags) #
                                                instantiation class of BKZ parameters
#bkz = BKZ2(A) #one way to instantiate BKZ # or
#bkz = BKZ2(GSO.Mat(A)) # another way to instantiate BKZ
bkz = BKZ2(LLL.Reduction(GSO.Mat(A))) #third way to instantiate BKZ
_ = bkz(par) # run BKZ with flags
print(log(A[0].norm(),2))
```

2. Part II: tracing how $r_{i,i}$'s change during BKZ

```
A = IntegerMatrix.random(80, "qary", k=40, bits=45)
k = 42
tours = 3

LLL.reduction(A)
```

```
M = GSO.Mat(A)
M.update_gso()
d = A.nrows

colours = ["#4D4D4D", "#5DA5DA", "#FAA43A", "#60BD68", "#F17CB0", "#B2912F", "#
                                                    B276B2", "#DECF3F", "#F15854"]
norms = [[log(M.get_r(i,i)) for i in range(d)]]
plt.plot(norms[0],label="lll", color=colours[0])

par = BKZ.Param(block_size=k,strategies=BKZ.DEFAULT_STRATEGY)
bkz = BKZ2(M)

for i in range(tours):
bkz.tour(par)
norms += [[log(M.get_r(j,j)) for j in range(d)]]
plt.plot(norms[i+1],label="tour %d"%i, color=colours[i+1])

legend = plt.legend(loc='upper center')
plt.show()
```

4 G6K

1. Launch SVP with sieving

```
    from fpylll import *
    from g6k import *
    from g6k.utils.stats import SieveTreeTracer
    from g6k.algorithms.bkz import pump_n_jump_bkz_tour
    from fpylll.util import gaussian_heuristic
    from math import sqrt

    n = 60
    A = IntegerMatrix.random(n, "qary", k=30, bits=30)
    g6k = Siever(A, seed=0x1337) # g6k has its own seed

    g6k.initialize_local(0, 0, n) # sieve the entire basis
    g6k(alg="bgj1") #alg \in {"nv", "gauss", "hk3"}
    v = g6k.best_lifts()[0][2]
    print('best lifts', g6k.best_lifts())
    w = A.multiply_left(v)  # w = v*A
    print('norms:', sum(w_**2 for w_ in w), A[0].norm()**2)
```

2. Simulate BKZ with sieving

```
n = 60
A = IntegerMatrix.random(n, "qary", k=30, bits=30)

g6k = Siever(A, seed=0x1337) # g6k has its own seed
g6k.initialize_local(0, 0, n)
sp = g6k.params.new()
sp.threads = 1
#g6k(alg="bgj1")

tracer = SieveTreeTracer(g6k, root_label=("bgj1"), start_clocks=True)
for b in (20, 30, 40, 50, 60): # progressive BKZ
  pump_n_jump_bkz_tour(g6k, tracer, b, pump_params={"down_sieve": True})
print(b, A[0].norm()**2)
```

Список литературы

[1] The FPLLL development team. fpylll, a Python wrapper for the fplll lattice reduction library, Version: 0.5.9, 2023. Available at https://github.com/fplll/fpylll