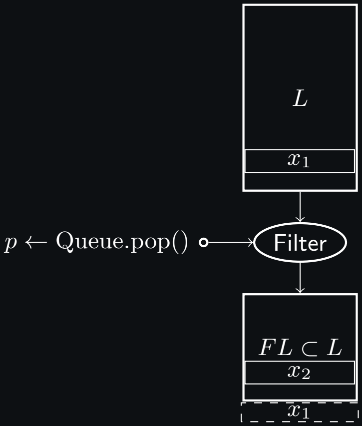
Details on Sieving Routines

December 18, 2017

What is implemented

We implemented basic 2- and 3-Gauss sieve.
An iteration of 3-sieve :



- if $||p \pm x_1 \pm x_2|| \leq ||p||$
  Update p, restart
- if $||p \pm x_1 \pm x_2|| \leq ||x_1||$
  Update x_1
  push x_1 into Queue
- else
  Append x_2 into FL

Some features of the (single-threaded) implementation

- We do keep the main list sorted
- We use standard std containers: list for L, queue for Q, vector for FL
- Lattice vectors are stored in $\mathbb{Z}$-basis, other representations (e.g., via Gram-Schmidt) are possible

Main sieving routines

```
class Sieve // in SieveJoint.h
{
        ...
        void run();
        void run_2_sieve(); //calls sieve_2_iteration
        void run_3_sieve(); //calls sieve_3_iteration

        //in SieveST2.cpp
        void sieve_2_iteration (Point &p);

        //in SieveST3.cpp
        void sieve_3_iteration (Point &p);
}
```

More details on the main list

Main list is an std::list<STNode> that stores *Bit approximations* to actual points

```
class STNode // in GaussListBitApprox.h
{
 bitset<sim_hash_len> bit_approximations[sim_hash_num];
 ExactPoint *ptr_to_exact;
}
```

The following parameters seem optimal for $n \in \{60 \ldots 80\}$:

```
// in Typedefs.h
static size_t constexpr sim_hash_len = 64;
static size_t constexpr sim_hash_num = 2;
```

What are the Bit approximations

```
// in BlockOrthogonalSimHash.h
```

Bit approximations are *sketches* of lattice vectors:

$$O(n) : \begin{cases} P - \text{ a random permutation matrix} \\ D = \text{diag}(\pm 1) \\ WH - \text{Walsh-Hadamard transform} \end{cases}$$

To create a sketch for $x \in L$

$$(WH_i \cdot D_i \cdot P_i)^{\otimes} \underbrace{[\, x \mid x \mid \ldots \mid \,]}_{\text{sim\_hash\_len}}$$

What are the Bit approximations

```
// in BlockOrthogonalSimHash.h
```

Bit approximations are *sketches* of lattice vectors:

$$O(n) : \begin{cases} P - \text{ a random permutation matrix} \\ D = \text{diag}(\pm 1) \\ WH - \text{Walsh-Hadamard transform} \end{cases}$$

To create a sketch for $x \in L$

$$\mathrm{sketch}_i = \mathrm{if}\Big((WH_i \cdot D_i \cdot P_i)^{\otimes} \underbrace{[\, x \mid x \mid \ldots \mid \,]}_{\text{sim\_hash\_len}} \Big) > 0$$

For random pair (x, y), we expect

$$(\text{sketch}(x) \oplus \text{sketch}(y)).\text{popcount}() = \text{sim\_hash\_len}/2$$

For close (x, y), we expect

$$(\text{sketch}(x) \oplus \text{sketch}(y)).\text{popcount}() \ll \text{sim\_hash\_len}/2$$

Some runtimes

Termination conditions are set as $||s|| \leq \mathrm{const} \cdot \sqrt{n}(\det L)^{1/n}$

	$k=3$	$k=2$
$n = 66$ + BKZ-32	47 sec	118 sec
$n = 68$ + BKZ-33	178 sec	391 sec
$n = 70$ + BKZ-35	283 sec	

For $k = 2$, 40 % of time is spent on approximate sc. prod
For $k = 3$, 20 % of time is spent on approximate sc. prod, 50 % on exact

ToDos

1. Merge into fplll-upstream (help needed!)
2. Implement progressive sieving
3. Implement 'Dimensions for free'
4. A better Sampler (e.g., ideas from Random Sampling)
5. Multi-threaded Gauss Sieve (partially done)
6. LSH (tried but seems worse in practice than bit approximations)
7. Inside BKZ: re-use the list