Algorithms for finite abelian groups

Johannes Buchmann* Sachar Paulus†

14th July 1993

Abstract

We study the complexity of the basic computational problems in a finite abelian group; i.e. we prove upper bounds for the number of operations necessary to compute the order of an element, discrete logarithms, the order of the group, the structure of the group, roots of an element.

1 The results

Let G be a finite abelian group, $|G| = h$; elements of G are given in some representation which has the following properties:

1. for $\alpha, \beta \in G$ we can compute $-\alpha$ and $\alpha + \beta$,
2. the "test on neutral element" 0_G is known; i.e. for a given element α we can decide whether $\alpha = 0_G$ or not and
3. we can choose elements γ of G "at random":

$$p(\gamma) = \frac{1}{h^{1+o(1)}} \qquad \left(\text{i.e. } p(\gamma) = \frac{1}{h^{1+e_\gamma}}, |e_\gamma| < f(h), \lim_{h\to\infty} f(h) = 0\right).$$

Typical examples of groups for which such a representation exists (that is, to which we can apply our results) are:

- the multiplicative group of a finite field,
- the set of points of an elliptic curve over a finite field,

*Fachbereich Informatik, Universitaet des Saarlandes, Germany
†Institut fuer experimentelle Mathematik, Universitaet - GH - Essen, Germany

- the class group of an imaginary quadratic number field.

The first two operations are called **group operations**, the last one **choose operation**. We will look at the number of operations necessary for "administrative tasks" (like sorting, searching etc.) separately and count them as **arithmetical operations**.

Assuming for all finite abelian groups G that $p(\gamma) = \frac{1}{h}$ for all $\gamma \in G$, our results are:

- The order of an element $\gamma \in G$ can be computed with
  - $O\left(\sqrt{\operatorname{ord}\gamma}\right)$ group operations and
  - $O\left(\sqrt{\operatorname{ord}\gamma} \cdot \log(\operatorname{ord}\gamma)\right)$ arithmetical operations.
- The discrete logarithm problem in G can be solved with
  - $O\left(\sqrt{h}\right)$ group operations and
  - $O\left(\sqrt{h} \cdot \log h\right)$ arithmetical operations.
- The order and the structure of G can be probabilistically computed with
  - $O\left(\log h \cdot \log \frac{1}{c}\right)$ choose operations,
  - $O\left(\sqrt{h} \cdot \log h \cdot \log \frac{1}{c}\right)$ group operations,
  - $O\left(\sqrt{h} \cdot \log^2 h \cdot \log \frac{1}{c}\right)$ arithmetical operations and
  - probability $1 - c$.
- Knowing the structure and a basis of G, one can decide whether an element of G has an n-th root and if so, one of them can be computed with
  - $O\left(\sqrt{h}\right)$ group operations and
  - $O\left(\sqrt{h} \cdot \log h\right)$ arithmetical operations.
- Let $h = \prod\limits_{i=1}^{m} p_i^{l_i}$. A root of an element of G can be probabilistically computed with
  - $O\left(\left(\sum\limits_{i=1}^{m} \frac{1}{P_i}\right) \cdot \log \frac{1}{c}\right)$ choose operations,
  - $O\left(\left(\sum\limits_{i=1}^{m} \frac{1}{P_i} \cdot \sqrt{p_i^{l_i}}\right) \cdot \log \frac{1}{c}\right)$ group operations,
  - $O\left(\left(\sum\limits_{i=1}^{m} \frac{1}{P_i} \cdot \sqrt{p_i^{l_i}} \cdot \log p_i^{l_i}\right) \cdot \log \frac{1}{c}\right)$ arithmetical operations and
  - probability $1 - c$,

  where the P_i's are coefficients depending on the structure of the p_i-subgroup of G.

Remark:

1. In the original work [Pa92], we treated a more complicated mathematical model of the representation of the elements of a group. So we could analyze all algorithms in a more general way; e.g. the real complexity of a choose operation is involved in the results. This might bring more information, but one can't see what happens at first sight. So we simplified the model in order to increase the ease of reading.

2 The Shanks-Algorithm

The computation of the order of an element γ of G is done by the well-known Baby-Step-Giant-Step-Algorithm [Sh70] of Shanks: Suppose the order of γ is bounded by L; then compute

$$\gamma_1 = 1\gamma, \ldots, \gamma_{\lceil\sqrt{L}\rceil} = \left\lceil\sqrt{L}\right\rceil \gamma;$$

if γ_i is equal to 0_G, output i (these are the Baby-Steps; we will include $\gamma_0 = 0_G$). If not, denote $\gamma_{\lceil\sqrt{L}\rceil}$ by γ^1 and compute

$$\gamma^2 = 2\gamma^1, \ldots, \gamma^{\lfloor\sqrt{L}\rfloor+1} = \left(\left\lfloor\sqrt{L}\right\rfloor + 1\right)\gamma^1$$

(the Giant-Steps). For each γ^i check whether it is equal to some γ_j. If so, the order of γ is $i \cdot \left\lceil\sqrt{L}\right\rceil - j$. It can be seen immediately that the algorithm needs $O(\sqrt{L})$ group operations and (for fast searching in the list of Baby-Steps) $O(\sqrt{L} \cdot \log L)$ arithmetical operations.

Now suppose that we don't know an upper bound for the order of γ. In this case, we apply the above algorithm with $L = L_0$ for a suitable L_0 (i.e. a good estimate for $\operatorname{ord}\gamma$). If it is successful, we have won. Otherwise set $L = 2 \cdot L$ and start a new round using the γ^i and γ_j already computed. Clearly, the new algorithm will always terminate. Looking at the number of operations needed, assume that the algorithm terminates in the k-th round, $k > 0$; i.e.

$$L_0 \cdot 2^{k-1} < \operatorname{ord}\gamma \leq L_0 \cdot 2^k \quad \Longleftrightarrow \quad k = \lceil \log \operatorname{ord}\gamma - \log L_0 \rceil .$$

The algorithm needs $O\left(\sqrt{L_0 \cdot 2^j}\right)$ group operations in the j-th round, so the total number of group operations is

$$\begin{aligned}
\sum_{j=0}^{k} O\left(\sqrt{L_0 \cdot 2^j}\right) &= O\left(\sum_{j=0}^{k} \sqrt{L_0} \cdot \left(\sqrt{2}\right)^j\right) \\
&= O\left(\sqrt{L_0} \cdot \frac{\left(\sqrt{2}\right)^{k+1} - 1}{\sqrt{2} - 1}\right) \\
&= O\left(\sqrt{L_0} \cdot 2^{\frac{\log \operatorname{ord}\gamma - \log L_0}{2}}\right) \\
&= O\left(\sqrt{\operatorname{ord}\gamma}\right).
\end{aligned}$$

The number of arithmetical operations can be determined in a similar way and so we have

Theorem 1 *Let G be a finite abelian group, $\gamma \in G$. The order* $\operatorname{ord}\gamma$ *of γ can be computed with*

- $O\left(\sqrt{\operatorname{ord}\gamma}\right)$ *group operations and*
- $O\left(\sqrt{\operatorname{ord}\gamma} \cdot \log \operatorname{ord}\gamma\right)$ *arithmetical operations.*

Remarks:

1. If an interval $[\![L_l, L_u]\!]$ of length l for the order of γ is known, we can modify the Baby-Step-Giant-Step-Algorithm to only search through the interval. This Algorithm requires $O(\sqrt{l}+\log L_l)$ group operations and $O(\sqrt{l}\cdot\log l)$ arithmetical operations.
2. Insead of doubling L in the above algorithm, we can square it. The resulting complexity is much less intuitive; but comparing the two complexity functions (one can't tell which one is asymptotically smaller) leads us to the point of view that the ease of implementation (i.e. the constants) would be important for the decision between them.
3. One can think of a parallel version of the Baby-Step-Giant-Step-Algorithm with group operation conut of $O(\log L)$; this is reached by the trivial parallelization process using
$$O\left(\frac{\sqrt{L}}{\log L}\right)$$
processors.

3 Minimal relations

Let γ, β be two elements of a finite abelian group G. We are interested in

1. the existence of $x \in \mathbb{N}$ such that $\beta = x\gamma$,
2. finding x, if it exists.

x is called the **discrete logarithm** of β to the base γ. To decide the existence of a discrete logarithm and, if it exists, to compute it, is called the **discrete logarithm problem**.

We can generalize this problem in the following way: Let $\gamma_1, \ldots, \gamma_m, \beta$ be elements of a finite abelian group G. Determine $x_1, \ldots, x_m, n \in \mathbb{N}$ such that

1. $n\beta = x_1\gamma_1 + \cdots + x_m\gamma_m$ and

2. n minimal.

$(x_1, \ldots, x_m, n)$ is called a **minimal relation** for $(\gamma_1, \ldots, \gamma_m, \beta)$. n is the order of β (more precisely $\overline{\beta}$) in the factor group $G/<\gamma_1, \ldots, \gamma_m>$, which we will denote by $\operatorname{ord}_{G/<\gamma_1,\ldots,\gamma_m>} \beta$, $x_1, \ldots, x_m$ are the coefficients of a linear combination of the elements generating the subgroup $0_{G/<\gamma_1,\ldots,\gamma_m>}$ and so there always exists a minimal relation.

Solving the discrete logarithm problem is a special case; if we have determined a minimal relation (x, n) for (γ, β), then there exists a discrete logarithm if $n = 1$ (then it is x) otherwise not. One can also understand the order n of a single element γ as a minimal relation (n) for (γ).

It is clear that the x_i can be bounded by $\operatorname{ord} \gamma_i$; we know better:

Proposition: *Let $n_i = \operatorname{ord}_{G/<\gamma_1,\ldots,\gamma_{i-1}>} \gamma_i$ for all $i = 1, \ldots, m$. Then there exists a minimal relation $(x_1, \ldots, x_m, n)$ for $(\gamma_1, \ldots, \gamma_m, \beta)$ with*

$$0 \leq x_i < n_i$$

for all $i = 1, \ldots, m$.

Proof: $n_i = \operatorname{ord}_{G/<\gamma_1,\ldots,\gamma_{i-1}>} \gamma_i$, so there exists a minimal relation $(x_{i,1}, \ldots, x_{i,i-1}, n_i)$ for $(\gamma_1, \ldots, \gamma_{i-1}, \gamma_i)$ for each i. Suppose we have a minimal relation $(x_1, \ldots, x_m, n)$ for $(\gamma_1, \ldots, \gamma_m, \beta)$ with $x_j < n_j$ for all $j > j_0$, $x_{j_0} \geq n_{j_0}$. Then divide x_{j_0} by n_{j_0}: $x_{j_0} = q \cdot n_{j_0} + x'_{j_0}$ and so

$$\left(x_1 + q \cdot x_{j_0,1}, \ldots, x_{j_0-1} + q \cdot x_{j_0,j_0-1}, x'_{j_0}, x_{j_0+1}, \ldots, x_m, m\right)$$

is a minimal relation for $(\gamma_1, \ldots, \gamma_m, \beta)$ with $x_j < n_j$ for $j \geq j_0$. Applying this procedure m times from x_m down to x_1 completes the proof. $\square$

As we have seen in the proof, the n_i are equal to the last entry of a minimal relation for $(\gamma_1, \ldots, \gamma_i)$. Consequently, we can compute a minimal relation for $(\gamma_1, \ldots, \gamma_m, \beta)$ by computing successive minimal relations for (γ_1), (γ_1, γ_2), ..., $(\gamma_1, \ldots, \gamma_m)$, $(\gamma_1, \ldots, \gamma_m, \beta)$ and then using the last entries of the already determined minimal relations as upper bounds for the new calculation.

Now suppose that we have an upper bound L for the order of G. The first step consists in determining the order n_1 of γ_1 by the Baby-Step-Giant-Step-Algorithm with complexity $O(\sqrt{L})$ group operations and $O(\sqrt{L} \cdot \log L)$ arithmetical operations, since $\operatorname{ord}_G \gamma_1 \leq h$. In one of the following steps, we want to determine $\operatorname{ord}_{G/<\gamma_1,\ldots,\gamma_{i-1}>} \gamma_i$; since

$$|< \gamma_1, \ldots, \gamma_{i-1} >| = \prod_{j=1}^{i-1} n_j,$$

an upper bound is therefore $\left\lfloor \frac{L}{\prod_{j=1}^{i-1} n_j} \right\rfloor$. Here we apply a modification of the Baby-Step-Giant-Step-Algorithm to determine a minimal relation using the known information *with the same complexity*, i.e. $O(\sqrt{L})$ group operations and $O(\sqrt{L} \cdot \log L)$ arithmetical operations.

We now explain the modified algorithm. We want to determine a minimal relation $(x_1, \ldots, x_m, n)$ for $(\gamma_1, \ldots, \gamma_m, \beta)$ with $0 \le x_i < n_i$ for $i = 1, \ldots, m$ and

$$0 \le n < \left\lfloor \frac{L}{\prod_{j=1}^{m} n_j} \right\rfloor = L_m$$

.

1. For each Baby-Step β_k $\left(1 \le k \le \left\lceil \sqrt{L_m} \right\rceil\right)$ we compute each element of the form
$$-\sum_{i=1}^{m} a_i \gamma_i + \beta_k \qquad (0 \le a_i \le \lceil \sqrt{n_i} \rceil)$$
and test whether it is equal to 0_G. In this case, we have found a minimal relation $(a_1, \ldots, a_m, k)$. If not, we sort these elements (according to an arbitrary order on the representations) such that logarithmic searching is possible.
2. For each Giant-Step β^l $\left(2 \le l \le \left\lfloor \sqrt{L_m} \right\rfloor + 1\right)$ we compute each element of the form
$$-\sum_{i=1}^{m} q_i \cdot \lceil \sqrt{n_i} \rceil \, \gamma_i + \beta^l \qquad (1 \le q_i \le \lceil \sqrt{n_i} \rceil + 1)$$
and test first if it is equal to 0_G (then we have found a minimal relation) and then if it is in the above list. If so,
$$\left(q_1 \cdot \lceil \sqrt{n_1} \rceil - a_1, \ldots, q_m \cdot \lceil \sqrt{n_m} \rceil - a_m, l \cdot \left\lceil \sqrt{L_m} \right\rceil - k\right)$$
is a minimal relation for $(\gamma_1, \ldots, \gamma_m, \beta)$.

It is a little bit technical to check that if L is an upper bound for the order of G, the algorithm always finds a minimal relation; we omit this here. The computing of the group elements above can be done in such a way that we need $O(1)$ group operations per element. So the total number of group operations is

$$O\left(\prod_{j=1}^{m} \left(\sqrt{n_j}\right) \cdot \sqrt{L_m}\right) = O\left(\prod_{j=1}^{m} \left(\sqrt{n_j}\right) \cdot \sqrt{\frac{L}{\prod_{j=1}^{m} n_j}}\right) = O\left(\sqrt{L}\right).$$

The total number of arithmetical operations can be determined in a similar way to be $O\left(\sqrt{L} \cdot \log L\right)$.

Applying the process of the previous section, we obtain a Baby-Step-Giant-Step-Algorithm which computes a minimal relation without needing an upper bound:

Theorem 2 *Let G be a finite abelian group, $\gamma_1, \ldots, \gamma_m, \beta \in G$. A minimal relation for $(\gamma_1, \ldots, \gamma_m, \beta)$ can be computed with*

- $O\left((m+1) \cdot \sqrt{h}\right)$ *group operations and*
- $O\left((m+1) \cdot \sqrt{h} \cdot \log h\right)$ *arithmetical operations.*

As application to the special case of discrete logarithms, we obtain the following result:

Theorem 3 *Let G be a finite abelian group, $\gamma, \beta \in G$. We can solve the discrete logarithm problem with*

- $O\left(\sqrt{h}\right)$ *group operations and*
- $O\left(\sqrt{h} \cdot \log h\right)$ *arithmetical operations.*

Remarks:

1. If the factorization of the order h of G is known, we can reduce the computations to p-groups, where determining the order of an element is very fast, because it must be a power of p. It can be shown that minimal relations in the p-subgroups can be combined to a minimal relation in G by applying the Chinese remainder algorithm to the components of the minimal relations.

2. We have analyzed this algorithm in general and obtained a very unintuitive theoretical result; nevertheless, it tells us to use it only in the case where h is a product of powers of small primes. See also [LeLe90].

4 Order and structure of a finite abelian group

Since $\prod_{j=1}^{m} n_j = |< \gamma_1, \ldots, \gamma_m >|$, we can probabilistically compute the order of a finite abelian group: We take elements at random $\gamma_i \in_R G$ and compute minimal relations $(x_{m+1,1}, \ldots, x_{m+1,m}, n_{m+1})$ for $(\gamma_1, \ldots, \gamma_m, \gamma_{m+1})$. If $n_{m+1} = 1$, then we know that γ_{m+1} lies in the subgroup generated by $(\gamma_1, \ldots, \gamma_m)$. In this case, we reject it, because it brings no further information. Otherwise, we set $H = H \cdot n_{m+1}$ (after the first round we set $H = n_1$). If we have rejected elements several times successively, we suppose that $\gamma_1, \cdots, \gamma_m$ already generate the whole group and so $h = H$. Since the largest possible subgroup has at most $\frac{h}{2}$ elements, the probability for having a bad result after k rejections is less than $\frac{1}{2^k}$. The number of choose operations is in this case less than $k \cdot m$. Since $m = O(\log h)$, we have

Theorem 4 *The order of a finite abelian group can be probabilistically computed with*

- $O\left(\log h \cdot \log \frac{1}{c}\right)$ *choose operations,*
- $O\left(\sqrt{h} \cdot \log h \cdot \log \frac{1}{c}\right)$ *group operations,*
- $O\left(\sqrt{h} \cdot \log^2 h \cdot \log \frac{1}{c}\right)$ *arithmetic operations and*
- *probability* $1 - c$.

We call the **structure** of a finite abelian group G the vector $(s_1, \ldots, s_m)$, $s_i \in \mathbb{N}$, with $s_i \mid s_{i+1}$ and $s_1 > 1$ such that

$$G \cong \mathbb{Z}/_{s_1\mathbb{Z}} \times \cdots \times \mathbb{Z}/_{s_m\mathbb{Z}}.$$

Let A be a regular quadratic matrix of rank n, S the corresponding matrix in Smith normal form, $(s_1, \ldots, s_m) = \operatorname{diag} S$ and $\mathcal{L}(A)$ the lattice created by A. Then we know

$$\mathbb{Z}^n/_{\mathcal{L}(A)} = \mathbb{Z}^n/_{\mathcal{L}(S)} \cong \mathbb{Z}/_{s_1\mathbb{Z}} \times \cdots \times \mathbb{Z}/_{s_m\mathbb{Z}}.$$

For $G =< \gamma_1, \ldots, \gamma_m >$, we have furthermore $G \cong \mathbb{Z}^n/\mathcal{L}(A)$, where A is the regular matrix whose columns $(x_{1,i}, \ldots, x_{i-1,i}, n_i, 0, \ldots, 0)^T$ contain minimal relations $(x_{1,i}, \ldots, x_{i-1,i}, n_i)$ for $(\gamma_1, \ldots, \gamma_i)$. So we obtain the structure of G by computing the Smith normal form of the matrix whose columns were computed in the previous algorithm and taking the non-1-entries in the diagonal. So we can, analogous to the order of a finite abelian group, probabilistically compute the structure with the same algorithm as before and then applying Smith-reduction. By *modular* Smith-reduction (modulo $h = \det A$), we can compute the Smith-normalform of A in $O(m^3 \cdot \log h) = O(\log^4 h) = O(\sqrt{h})$ arithmetical operations (see [DoKaTr87]). So we have

Theorem 5 *The structure of a finite abelian group can be probabilistically computed with*

- $O\left(\log h \cdot \log \frac{1}{c}\right)$ *choose operations,*
- $O\left(\sqrt{h} \cdot \log h \cdot \log \frac{1}{c}\right)$ *group operations,*
- $O\left(\sqrt{h} \cdot \log^2 h \cdot \log \frac{1}{c}\right)$ *arithmetic operations and*
- *probability* $1 - c$.

Remark:

1. If $G = \mathbb{Z}/s_1\mathbb{Z} \times \cdots \times \mathbb{Z}/s_m\mathbb{Z}$, then we call $(\epsilon_1, \ldots, \epsilon_m)$ with $\operatorname{ord} \epsilon_i = s_i$ and $G =< \epsilon_1, \ldots, \epsilon_m >$ a **basis** of G. Such a basis can be computed using the transformation matrices during the (non-modular) Smith-reduction. We suppose that one can compute it even using *modular* Smith-reduction.

2. The knowledge of the order of a finite abelian group doesn't lead us to a faster algorithm to determine the structure of the group. We can't even give a deterministic algorithm for this problem!

5 Roots in a finite abelian group

Let G be a finite abelian group, $\alpha \in G$ and $n \in \mathbb{N}$. We call $\gamma \in G$ with $n\gamma = \alpha$ an **n-th root** of α.

If the structure $(s_1, \ldots, s_m)$ and a basis $(\epsilon_1, \ldots, \epsilon_m)$ of G are known, then the problem of existence and computation of an n-th root of α is reduced to finite cyclic groups: we compute a minimal relation $(a_1, \ldots, a_m, 1)$ for $(\epsilon_1, \ldots, \epsilon_m, \alpha)$ using $(s_1, \ldots, s_m)$. Then there exists a n-th root of α if and only if there exists an $(n \bmod s_i)$-th root of $\overline{a_i}$ in $\mathbb{Z}/s_i\mathbb{Z}$ for each i. Furthermore, we can easily check the following fact:

Proposition: *Let $n, a \in [\![0, s-1]\!]$. The congruence*

$$n \cdot x \equiv a \bmod s$$

is solvable if and only if $\gcd(n, s) \mid a$. If so, there are $\gcd(n, s)$ different solutions x_i in $[\![0, s-1]\!]$ with

$$(*) \quad x_i \equiv x_j \bmod \frac{s}{\gcd(n,s)} \qquad (i, j \in [\![0, \gcd(n,s) - 1]\!]).$$

To compute one solution, let $d = \gcd(n, s)$ and $n' = \frac{n}{d}$. Then $\gcd(n', s) = 1$ and so compute n_0 such that $n_0 \cdot n' \equiv 1 \bmod s$ by the extended euclidean algorithm. Thus $n_0 \cdot \frac{a}{d}$ is a solution. All other solutions can be obtained by using $(*)$.

One solution to our general problem is then given by computing $x_1\epsilon_1 + \cdots + x_m\epsilon_m$. This can be done by fast multiplication (same idea as fast exponentiation) in $O(\log h)$ group operations.

Since the extended euclidean algorithm has the same complexity as simple multiplication ($O(\log^2 s)$ bit operations), we can say: once the minimal relations are computed, the number of arithmetical operations necessary to compute an n-th root of a in $\mathbb{Z}/s\mathbb{Z}$ is constant.

Theorem 6 *Let G be a finite abelian group, $n \in \mathbb{N}$. Knowing the structure and a basis of G, we can decide whether an element has an n-th root and, if so, compute one of them with*

- $O\left(\sqrt{h}\right)$ *group operations and*
- $O\left(\sqrt{h} \cdot \log h\right)$ *arithmetical operations.*

All t n-th roots of an element can then be computed with

- $O\left(t + \log h\right)$ *group operations and*
- $O(1)$ *arithmetical operations.*

Once we have computed the $\frac{s_i}{\gcd(n,s_i)}\epsilon_i$ by fast multiplication, we can obtain each of the $t = \prod_{i=1}^{m} \gcd(n, s_i)$ solutions with $O(1)$ group operations. This gives the second part of the theorem.

Remark:

1. Analogous to a remark in section 3, it could be much faster to use the Chinese remainder algorithm and solve the congruences in p-groups. Again, this is the case when all prime factors of h are small.

Our last topic is the computation of an n-th root of an element α of G knowing only the order h of the group G. This is done by computing successively a p-th root for each prime factor p of n.

Let $h = p^l \cdot h_0$ with $\gcd(p, h_0) = 1$. Then compute h_0' with $h_0' \cdot h_0 \equiv -1 \bmod p$ by the extended euclidean algorithm. Let $\chi = \frac{h_0 \cdot h_0' + 1}{p}\alpha$ (compute it with fast multiplication), then we have $\mathrm{ord}(p\chi - \alpha) = \mathrm{ord}(h_0 \cdot h_0'\alpha) \mid p^l$. By this deterministic computation we have reduced the problem to the case of a p-group. Specially if $p \nmid h$, then $h_0 = h$ and so χ is a p-th root of α (we don't use the fact that p is a prime here, it also works for n with $\gcd(n, h) = 1$). This computation requires $O(\log h)$ group operations and $O(1)$ arithmetical operations.

Theorem 7 *Let G be a finite abelian group, $|G| = h$, $n \in \mathbb{N}$ with $\gcd(n, h) = 1$. An n-th root of an element of G can be computed with*

- $O\left(\log h\right)$ *group operations and*
- $O\left(1\right)$ *arithmetical operations.*

We will now show how to probabilistically determine p-th roots of an element α in a p-group. Let $\beta \in G$ such that there exists a minimal relation (x_0, x_1) for (α, β) with $p \nmid x_0$ and $x_1 > 1$. Then $\gcd(x_1 - x_0, \operatorname{ord} \alpha) = 1$ (because $x_1 = p^k$, $k \geq 1$ and $\alpha = p^l$), compute z_1, z_2 such that $z_1 \cdot (x_1 - x_0) + z_2 \cdot \operatorname{ord} \alpha = 1$ with the extended euclidean algorithm. Then

$$\begin{aligned} z_1 \cdot x_1(\alpha + \beta) &= z_1 \cdot x_1 \alpha + z_1 \cdot (-x_0 \alpha) \\ &= z_1 \cdot (x_1 - x_0)\alpha \\ &= (1 - z_2 \cdot \operatorname{ord} \alpha)\alpha \\ &= \alpha \end{aligned}$$

and so $\frac{z_1 \cdot x_1}{p}(\alpha + \beta)$ is a p-th root of α.

Clearly, the success of this algorithm depends on the existence of an element $\beta \in G$ with the above properties. We can choose an element at random for β and test the properties; after a certain number of choose operations (depending on the structure of G) with no success, we state that there is no p-th root of α. The analysis is not intuitive at all; we give here just the result:

Theorem 8 *Let G be a finite abelian group, $|G| = \prod_{i=1}^{m} p_i^{l_i}$. A root of an element of G can be probabilistically computed with*

- $O\left(\left(\sum_{i=1}^{m} \frac{1}{P_i}\right) \log \frac{1}{c}\right)$ *choose operations,*
- $O\left(\left(\sum_{i=1}^{m} \frac{1}{P_i} \cdot \sqrt{p_i^{l_i}}\right) \log \frac{1}{c}\right)$ *group operations,*
- $O\left(\left(\sum_{i=1}^{m} \frac{1}{P_i} \cdot \sqrt{p_i^{l_i}} \cdot \log p_i^{l_i}\right) \log \frac{1}{c}\right)$ *arithmetical operations and*
- *probability* $1 - c$

where the P_i's are coefficients depending on the structure of the p_i-subgroup of G.

Remark:

1. It would be interesting to use finite abelian groups G and n-th roots with $\gcd(n, |G|) \neq 1$ for cryptographic schemes, since the complexity of computing an n-th root is much larger than of solving the discrete logarithm problem: one first has to determine the order of G (this is already more complex than solving the discrete logarithm problem) and perhaps the structure and a basis before applying one of the methods above.

References

[DoKaTr87] P.D. Domich, R. Kannan and L.E. Trotter, *Hermite Normal Form Computation using Modulo Determinant Arithmetic*, Math. Op. Res. 12 (1987), S. 50–59.

[LeLe90] A.K. Lenstra, H.W. Lenstra, jr. *Algorithms in Number Theory* in J. van Leeuwen, *Handbook of theoretical computer science*, Elsevier Science Publishers B.V. 1990, S. 673–715.

[Sh70] D. Shanks, *Class Number, A Theory of Factorization and Genera,* Proceedings Symposium Pure Mathematics, Vol. 20, American Mathematical Society, 1970, S. 415–440.

[Pa92] S. Paulus, *Algorithmen fuer endliche abelsche Gruppen*, Diplomarbeit, Universitaet des Saarlandes, 1992.