Electronic Colloquium on Computational Complexity, Report No. 52 (2007)

Linear and Sublinear Time Algorithms for the Basis of Abelian Groups

Li Chen*and Bin Fu†

Abstract

It is well known that every finite Abelian group G can be represented as a product of cyclic groups: $G = G_1 \times G_2 \times \cdots G_t$, where each G_i is a cyclic group of size p^j for some prime p and integer $j \geq 1$. If a_i is the generator of the cyclic group of G_i, $i = 1, 2, \cdots, t$, then the elements $a_1, a_2, \cdots, a_t$ are the basis of G. In this paper, we first obtain an $O(n)$-time deterministic algorithm for computing the basis of an Abelian group with n elements. This improves the previous $O(n^2)$ time algorithm found by Chen [1]. We then derive an $O((\sum_{i=1}^{t} p_i^{n_i-1} n_i^2 \log p_i)(\log n)(\log\log n))$-time randomized algorithm to compute the basis of Abelian group G of size n with factorization $n = p_1^{n_1} \cdots p_k^{n_t}$, which is also a part of the input. This shows that for a large number of cases, the basis of a finite Abelian group can be computed in sublinear time. For example, it implies an $O(n^{1-\frac{1}{d}}(\log n)^3 \log\log n)$-time randomized algorithm to compute the basis of an Abelian group G of size $n = p_1^{n_1} \cdots p_t^{n_t}$, where $d = \max\{n_i | i = 1, \cdots, t\}$. It is a sublinear time algorithm if $\max\{n_i | i = 1, \cdots, t\}$ is bounded by a constant. It also implies that if n is an integer in $[1, m] - G(m, c)$, then the basis of an Abelian group of size n can be computed in $O((\log n)^{c+3} \log\log n)$-time, where c is any positive constant and $G(m, c)$ is a subset of the small fraction of integers in $[1, m]$ with $\frac{|G(m,c)|}{m} = O(\frac{1}{(\log m)^{c/2}})$ for every integer m.

1. Introduction

The theory of groups is a fundamental theory of mathematics. Its applications can be found throughout entire mathematics and theoretical physics especially quantum mechanics. In recent years, interest in researching the computational complexity of groups has raised dramatically due to the ever-increasing significance of its relationship to quantum computing and its application in elliptic curve cryptography. Since the early developmental period of computational complexity, computer scientists have shown great interest in the study of groups.

Abelian groups are groups with commutative property. It is well known that a finite Abelian group can be decomposed to a direct product of cyclic groups with prime-power order (called cyclic p-groups) [8]. This fundamental theorem is also called the Kronecker decomposition theorem. In quantum computing, the hidden subgroup problem (HSP) greatly interests scientists. The finite Abelian case was first used to spectacular effect by Shor and Simon [18, 19]. "If the group G is Abelian, then it is possible to solve the HSP in polynomial time with bounded error on a quantum computer." A polynomial time algorithm in quantum computing means an algorithm whose running time is a polynomial of the logarithm of the size of the group. There is a polynomial time quantum algorithm for solving HSP over Abelian groups [2, 4, 10, 12, 13, 16, 18, 19]. This is important since the famous Shor's quantum factoring algorithm is one particular case.

On the other hand, finite Abelian groups have been used in elliptic curve cryptosystems that were introduced by Miller in 1986 [16]. This system is based on the discrete logarithm problem, which is as follows: given an element g in a finite group G (over an elliptic curve) and another element h in G, find an integer x such that $g^x = h$ [13]. The advantage of using elliptic curve crypto-systems over other public-key crypto-systems is that the elliptic curve system may lead to smaller key sizes and better performance with the similar level of security. Another application can be found in [4]

No polynomial-time algorithm has been found for determining if two general groups are isomorphic. The group isomorphism problem is related to the graph isomorphism problem and is also easier to solve than the graph isomorphism problem [15]. Hoffmann published a book in 1982 that presents interesting algebraic results that relate the graph isomorphism problem to the automorphism groups of the two graphs [6]. The development can be found in [11]. Tarjan [14] showed an $O(n^{\log n + O(1)})$ time algorithm for the group isomorphism problem. Savage [17] claimed the isomorphism between two Abelian groups can be checked in

*Address: Department of Computer Science, University of District of Columbia, Washington, DC 20008, USA, Email:lchen@udc.edu. Phone: 202-274-6301.

†Address: Department of Computer Science, University of Texas-Pan American, Edinburg, TX 78539, USA, Email: binfu@cs.panam.edu. Phone: 956-381-3635. Fax: 956-384-5099.

ISSN 1433-8092

$O(n^2)$ steps. Vita [20] improved it to $O(n)$ time for the Abelian p-group and $O(n \log n)$ time for Abelian group. Kavitha [9] showed that the Abelian group isomorphism problem can be computed in $O(n \log\log n)$ time. Garzon and Zalcstein [5] also discussed that the polynomial time algorithms for the isomorphism problem of Abelian groups.

Since the basis of an Abelian group fully determines its structure, finding the basis is crucial in computing the general properties for Abelian groups. Also, finding the basis of an Abelian group is the generalization of the integer factorization problem, one of the fundamental problems in computer science. For an integer $n > 0$, the set $\{0, 1, 2, \cdots, n-1\}$ with the addition (mod n) forms an Abelian group. An algorithm for finding the basis of an Abelian group can be converted into an algorithm for checking the isomorphism between two Abelian groups. Therefore, pursing efficient algorithms in the classical computing model for the basis of Abelian group has fundamental significance.

In this paper, we obtain an $O(n)$-time deterministic algorithm for computing the basis of an Abelian group with n elements. This improves the previous $O(n^2)$ time algorithm of Chen [1]. We derive a randomized algorithm that computes the basis of G, which has $n = p_1^{n_1} \cdots p_k^{n_k}$ elements, in $O((\sum_{i=1}^k p_i^{n_i-1} n_i^2 \log p_i)(\log n)(\log\log n))$ running time. It implies a randomized algorithm with $O(n^{1-\frac{1}{d}} (\log n)^3 \log\log n)$ running time so that given an Abelian group G of size $n = p_1^{n_1} \cdots p_k^{n_k}$, it computes the basis of G, where $d = \max\{n_i | i = 1, \cdots, k\}$. It also implies that if n is an integer in $[1, m] - G(m, c)$, then the basis of an Abelian group of size n can be computed in $O((\log n)^{c+3} \log\log n)$-time, where c is any positive constant and $G(m, c)$ is a subset of small fraction of integers in $[1, m]$ with $\frac{|G(m,c)|}{m} = O(\frac{1}{(\log m)^{c/2}})$ for every integer m. It is a sublinear time algorithm if $d = \max\{n_i | i = 1, \cdots, k\}$ is bounded by a constant. Since saving the multiplication table of a group of size n takes $O(n^2)$ space, the multiplication table of the Abelian group can be accessed as an oracle during the computation.

2. Notations

For two positive integers x and y, (x, y) represents the greatest common divisor (GCD) between them. For a set A, $|A|$ is the number of elements in A. For a real number x, $\lfloor x \rfloor$ is the largest integer $\le x$ and $\lceil x \rceil$ is the least integer $\ge x$. For two integers x and y, $x|y$ means that $y = xc$ for some integer c.

A *group* is a nonempty set G with a binary operation "$\cdot$" that is closed in set G and satisfies the following properties (for simplicity, "ab" represents "$a \cdot b$"): 1)for every three elements a, b and c in G, $a(bc) = (ab)c$; 2)there exists an identity element $e \in G$ such that $ae = ea = a$ for every $a \in G$; 3)for every element $a \in G$, there exists $a^{-1} \in G$ with $aa^{-1} = a^{-1}a = e$. A group G is *finite* if G has only finite elements. Let e be the identity element of G, i.e. $ae = a$ for each $a \in G$. For $a \in G$, $\mathrm{ord}(a)$, the order of a, is the least integer k such that $a^k = e$. For $a \in G$, define $[a]$ to be the subgroup of G generated by the element a (in other words, $[a] = \{e, a, a^2, \cdots, a^{\mathrm{ord}(a)-1}\}$). Let A and B be two subsets of group G, define $AB = A \cdot B = A \circ B = \{ab | a \in A \text{ and } b \in B\}$.

A group G is an *Abelian* group if $ab = ba$ for every two elements $a, b \in G$. Assume that G is an Abelian group with elements $g_1, g_2, \cdots, g_n$. For each element $g_i \in G$, it corresponds to an index i. A finite Abelian group G of n elements can be represented as $G = G(p_1^{n_1}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t})$, where $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$, $p_1 < p_2 < \cdots < p_t$ are the prime factors of n, and $G(p_i^{n_i})$ is a subgroup of G with $p_i^{n_i}$ elements (see [7]). We also use the notation G_{p_i} to represent the subgroup of G with size $p_i^{n_i}$. Any Abelian group G of size p^m can be represented by $G = G(p^{m_1}) \circ G(p^{m_2}) \circ \cdots \circ G(p^{m_k})$, where $m = \sum_{i=1}^k m_i$ and $1 \le m_1 \le m_2 \le \cdots \le m_k$. Notice that each $G(p^{m_i})$ is a cyclic group.

For, $a_1, a_2, \cdots, a_k$ from the Abelian group G, denote $[a_1, a_2, \cdots, a_k]$ to be the set of all elements in G *generated* by $a_1, \cdots, a_k$. In other words, $[a_1, a_2, \cdots, a_k] = [a_1][a_2] \cdots [a_k]$. An element $a \in G$ is *independent of* $a_1, a_2, \cdots, a_k$ in G if $[a_1, a_2, \cdots, a_k] \cap [a] = \{e\}$.

The elements $a_1, a_2, \cdots, a_k$ from the Abelian group G are *independent* if $[a_i] \cap (\prod_{j \in \{1,2,\cdots,k\}-i} [a_j]) = \{e\}$ for every i with $1 \le i \le k$. The basis of G consists of independent elements $a_1, \cdots, a_k$ that can generate all elements of G (in other words, $G = [a_1, a_2, \cdots, a_k]$).

3. Algorithm with $O(n \log n)$ Steps

The algorithm in this section has two parts. The first part decomposes an Abelian group into product $G(p_1^{n_1}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_k^{n_k})$. To find the subgroup of size $p_i^{n_i}$, it is converted to find the set of elements with the order of p_i-power.

The second part finds the basis of each group $G(p_i^{n_i})$. Assume that $b_1, \cdots, b_h$, which satisfy $\mathrm{ord}(b_1) \ge \mathrm{ord}(b_2) \ge \cdots \ge \mathrm{ord}(b_h)$, are the elements of a basis of the Abelian group $G(p^u)$. We will find another set of the basis $a_1, \cdots, a_h$. The element a_1 is selected among all elements in $G(p^u)$ such that a_1 has the largest order $\text{ord}(a_1)$. Therefore, $\text{ord}(a_1) = \text{ord}(b_1)$. Assume that $a_1, \cdots, a_k$ have been obtained such that $\text{ord}(a_1) = \text{ord}(b_1), \cdots, \text{ord}(a_k) = \text{ord}(b_k)$. We show that it is always possible to find another a_{k+1} such that $([a_1] \cdots [a_k]) \cap [a_{k+1}] = \{e\}$ and $\text{ord}(a_{k+1}) = \text{ord}(b_{k+1})$. The possibility of such an extension is shown at Lemma 4 and Lemma 6. We maintain a subset M of elements of $G(p^u)$ such that M consists of all elements $a \in G$ that are independent of $a_1, a_2, \cdots, a_k$ and $\text{ord}(a) \leq \text{ord}(a_k)$. We search for a_{k+1} from M by selecting the element with the highest order. After a_{k+1} is found, M will be updated.

In this section, we develop an $O(n \log n)$ time algorithm to compute the basis of a finite Abelian group. The algorithm and its proof are self-contained. In section 4, we improve this algorithm to be in linear time by using a result of Kavitha [9].

Lemma 1. *There exists an $O(n \log n)$ time algorithm such that given a group G of size n with the factorization $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$, it computes the order of all elements g with $\text{ord}(g) = p_i^j$ for some $p_i || G|$ and $j \geq 0$.*

Proof: Assume $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$ and $n_i \geq 1$ for $i = 1, 2, \cdots, t$. Given the multiplication table of G, with $O(\log m)$ steps, we can compute a^m. This can be done by a straightforward divide and conquer method with the recursion $a^m = a^{\frac{m}{2}} \cdot a^{\frac{m}{2}}$ if m is even or $a^m = a \cdot a^{\lfloor \frac{m}{2} \rfloor} \cdot a^{\lfloor \frac{m}{2} \rfloor}$ if m is odd.

For each prime factor p_i of n, compute a^{p_i} for each $a \in G$. Build the table T_i so that $T_i(a) = a^{p_i}$ for $a \in G$. The table T_i can be built in $O(n \log p_i)$ steps.

For each $a \in G$ and prime factor p_i of n, try to find the least integer j, which may not exist, such that $a^{p_i^j} = e$. It takes $O(n_i)$ steps by looking up the table T_i. For each p_i, trying all $a \in G$ takes $O(n(\log p_i + n_i))$ steps. Therefore, the total time is $O(n(\sum_{i=1}^{t} (\log p_i + n_i)) = O(n \log n)$. ∎

Lemma 2. *Assume G is an Abelian group of size n. We have the following two facts: 1) If $n = m_1 m_2$ with $(m_1, m_2) = 1$, $G' = \{a \in G | a^{m_1} = e\}$ and $G'' = \{a^{m_1} | a \in G\}$, then both G' and G are subgroups of G, $G = G' \circ G''$, $|G'| = m_1$ and $|G''| = m_2$. Furthermore, for every $a \in G$, if $(\text{ord}(a), m_1) = 1$, then $a \in G''$. 2)If $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$, then $G = G(p_1^{n_1}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t})$, where $G(p_i^{n_i}) = \{a \in G | a^{p_i^{n_i}} = e\}$ for $i = 1, \cdots, t$.*

Proof: It is easy to verify that G' is subgroup of G. Assume $a_1, \cdots, a_{s_1}, b_1, \cdots, b_{s_2}$ are the elements in a basis of G such that $\text{ord}(a_i) | m_1$ for $i = 1, \cdots, s_1$ and $\text{ord}(b_j) | m_2$ for $j = 1, \cdots, s_2$. It is easy to see that $a_i^{m_1} = e$ for $i = 1, \cdots, s_1$ and $b_j^{m_1} \neq e$ for $j = 1, \cdots, s_2$. For each b_j, $[b_j] = [b_j^{m_1}]$ since $(m_1, m_2) = 1$ and $\text{ord}(b_j) | m_2$. Assume that $x = a^{m_1}$ and $y = a'^{m_1}$. Both x and y belong to G''. Let's consider $xy = (aa')^{m_1}$. We still have $xy \in G''$. Thus, G'' is closed under multiplication. Since G'' is a subset of a finite group, G'' is a group. Therefore, G'' is a group generated by $b_1^{m_1}, \cdots, b_{s_2}^{m_1}$ that is the same as the group generated by $b_1, \cdots, b_{s_2}$. Therefore, G'' is of size m_2. On the other hand, G' has basis of elements $a_1, \cdots, a_{s_1}$ and is of size m_1. We also have that $G' \cap G'' = \{e\}$. It is easy to see that $G = G' \circ G''$. For $a \in G$ with $(\text{ord}(a), m_1) = 1$, $[a^{m_1}] = [a]$ and $a^{m_1} \neq e$. So, we have $a^{m_1} \in G''$, which implies that $a \in [a] = [a^{m_1}] \subseteq G''$. Part 2) follows from part 1). ∎

Lemma 3. *Assume G is a group of size $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$. Given the table of the orders of all elements $g \in G$ with $\text{ord}(g) = p_i^j$ for some p_i and $j \geq 0$, with $O(n)$ steps, G can be decomposed as the product of subgroups $G(p_1^{n_1}) \circ \cdots \circ G(p_t^{n_t})$.*

Proof: By Lemma 2, the elements of each $G(p_i^{n_i})$ consists of all elements of G with order p_i^j for some integer $j \geq 0$. Therefore, we have the following algorithm:

Compute the list of integers $p_1, p_1^2, \cdots, p_1^{n_1}, p_2, p_2^2, \cdots, p_2^{n_2}, \cdots, p_t, p_t^2, \cdots, p_t^{n_t}$. This can be done in $O(\log n)^2$ steps because $n_1 + n_2 + \cdots + n_t \leq \log n$. Also sort those integers $p_1, p_1^2, \cdots, p_1^{n_1}, p_2, p_2^2, \cdots, p_2^{n_2}, \cdots, p_t, p_t^2, \cdots, p_t^{n_t}$ by increasing order. It takes $(\log n)^2$ steps because bubble sorting those $\log n$ integers takes $O((\log n)^2)$ steps. Let $q_1 < q_2 \cdots < q_m$ be the list of integers sorted from $p_1, p_1^2, \cdots, p_1^{n_1}, p_2, p_2^2, \cdots, p_2^{n_2}, \cdots, p_t, p_t^2, \cdots, p_t^{n_t}$.

Set up the array A of n buckets. Put all elements of order k into bucket $A[k]$. Merge the buckets $A[p_i], A[p_i^2], \cdots, A[p_i^{n_i}]$ to obtain $G(p_i^{n_i})$. This can be done by scanning the array A from left to right once and fetching the elements from the array $A[\ ]$ at those positions $q_1 < q_2 \cdots < q_m$. ∎

Lemma 4 ([1]). *Let G be an Abelian group of size p^t for prime p and integer $t \geq 1$. Assume $a_1, a_2, \cdots, a_k$ are independent elements in G and b is also an elements in G with $\text{ord}(b) \leq \text{ord}(a_i)$ for $i = 1, \cdots, k$. Then there exists $b' \in [a_1, \cdots, a_k, b]$ with $\text{ord}(b') | \text{ord}(b)$ such that (1) $a_1, \cdots, a_k, b'$ are independent elements in G; (2) $[a_1, \cdots, a_k, b'] = [a_1, \cdots, a_k, b]$; and (3)$b'$ can be expressed as $b' = b \prod_{i=1}^{k} (a_i^{-t_i p^{\xi_i - \eta}})$, where η is the least integer that $b^{p^\eta} \in [a_1, \cdots, a_k]$.*

Proof: Let $\mathrm{ord}(a_i) = p^{n_i}$ and $\mathrm{ord}(b) = p^m$, $n_i \geq m$ for $i = 1, ..., k$. Let $[a_1, \cdots, a_k] \cap [b] = [c]$. We assume that $c \neq e$ (Otherwise, let $b' = b$ and finish the proof). Assume,

$$c = a_1^{t_1 p^{\xi_1}} \cdots a_k^{t_k p^{\xi_k}} = b^{hp^\eta}, \tag{1}$$

where $0 \leq t_i < p^{n_i - \xi_i}$ and ($t_i = 0$ or $(t_i, p) = 1$) for $i = 1, \cdots, k$ and $0 < h < p^{m-\eta}$ with $(h, p) = 1$ and $\eta < m$ (because $c \neq e$).

Since $(t_i, p) = 1$, the order of each $a_i^{t_i p^{\xi_i}}$ is $\frac{p^{n_i}}{p^{\xi_i}}$. The order of $a_1^{t_1 p^{\xi_1}} \cdots a_k^{t_k p^{\xi_k}}$ is $\max\{\frac{p^{n_i}}{p^{\xi_i}} | t_i \neq 0, \text{ and } i = 1, ..., k\}$. On the hand, the order of b^{hp^η} is $\frac{p^m}{p^\eta}$. So, we have $\max\{\frac{p^{n_i}}{p^{\xi_i}} | t_i \neq 0, \text{ and } i = 1, ..., k\} = \frac{p^m}{p^\eta}$. Therefore, $p^{n_i - \xi_i} \leq p^{m-\eta}$ for each $i = 1, \cdots, k$. So, we have $n_i - \xi_i \leq m - \eta$. Since $(h, p) = 1$, we have $[b^{hp^\eta}] = [b^{p^\eta}]$. Without loss of generality, we assume that $h = 1$. It is easy to see that η is the least integer such that $b^{p^\eta} \in [a_1, \cdots, a_k]$. We have $\xi_i \geq \eta + (n_i - m) \geq \eta$ for $i = 1, ..., k$. Let

$$b' = \prod_{i=1}^{k} (a_i^{-t_i p^{\xi_i - \eta}}) \cdot b. \tag{2}$$

Clearly, $b' \in \prod_{i=1}^k [a_i] \cdot [b]$. By (1) and the fact $h = 1$, $b^{p^\eta} = (\prod_{i=1}^k a_i^{t_i p^{\xi_i - \eta}})^{p^\eta}$. By (2), we have $b'^{p^\eta} = e$, which implies $\mathrm{ord}(b') | p^\eta$. We obtain the following:

$$[a_1, \cdots, a_k, b] = [a_1, \cdots, a_k, b'].$$

We now want to prove that $[a_1, \cdots, a_k] \cap [b'] = \{e\}$.

If, on the contrary, $[a_1, \cdots, a_k] \cap [b'] = [c']$ and $c' \neq e$. We assume $c' = b'^{p^{u\eta'}}$ for some u with $(u, p) = 1$. Since $[b'^{p^{u\eta'}}] = [b'^{p^{\eta'}}]$, let $u = 1$. There exist integers $s_i, \xi_i' (i = 1, \cdots, k)$ such that

$$c' = \prod_{i=1}^{k} a_i^{s_i p^{\xi_i'}} = b'^{p^{\eta'}} = \prod_{i=1}^{k} a_i^{-t_i p^{\xi_i - \eta + \eta'}} \cdot b^{p^{\eta'}}, \tag{3}$$

where $0 \leq \xi_i' < n$, $0 \leq \eta' < \eta$. If $\eta' \geq \eta$, we have $c' = e$ by (1), (2), and (3). This contradicts the assumption $c' \neq e$.

Since $c = b^{p^\eta} \neq e$, we have $b^{p^{\eta'}} \neq e$. Since $[a_1, \cdots, a_k] \cap [b] = [b^{p^\eta}]$ and $\eta > \eta'$, we have $b^{p^{\eta'}} \notin [a_1, \cdots, a_k] \cap [b]$. By (3),

$$b^{p^{\eta'}} = \prod_{i=1}^{k} a_i^{s_i p^{\xi_i'}} \cdot \prod_{i=1}^{k} a_i^{t_i p^{\xi_i - \eta + \eta'}} \tag{4}$$

By (4), we also have $b^{p^{\eta'}} \in [a_1, \cdots, a_k] \cap [b]$. This contradicts that η is the least integer such that $b^{p^\eta} \in [a_1, \cdots, a_k]$ (notice that $\eta' < \eta$). Thus, $[a_1, \cdots, a_k] \cap [b'] = \{e\}$. ∎

Definition 5. Assume that group G has basis $b_1, \cdots, b_t$ with $\mathrm{ord}(b_1) \geq \cdots \geq \mathrm{ord}(b_t)$.

- Assume that $a_1, \cdots, a_k$ and b are the same as those in Lemma 4. We use $b'(a_1, \cdots, a_k, b)$ to represent b' derived in the Lemma 4.
- Let $a_1, \cdots, a_k$ be the elements of G with $\mathrm{ord}(a_1) = \mathrm{ord}(b_1), \cdots, \mathrm{ord}(a_k) = \mathrm{ord}(b_k)$ and $(\prod_{i \neq j} [a_i]) \cap [a_j] = \{e\}$ for every $j = 1, \cdots, k$. Then $a_1, \cdots, a_k$ is called a *partial basis* of G. If $C(a_1, \cdots, a_k) = \{a \in G | [a_1, \cdots, a_k] \cap [a] = \{e\} \text{ and } \mathrm{ord}(a) \leq \mathrm{ord}(a_k)\}$, then $C(a_1, \cdots, a_k)$ is called a *complementary space* of the partial basis $a_1, \cdots, a_k$.

Lemma 6. *Let $a_1, \cdots, a_k$ be partial basis of the Abelian G with p^i elements for some prime p and integer $i \geq 0$. Then 1)G can be generated by $\{a_1, \cdots, a_k\} \cup C(a_1, \cdots, a_k)$; and 2)the partial basis $a_1, \cdots, a_k$ can be extended to another partial basis $a_1, \cdots, a_k, a_{k+1}$ with complementary space $C(a_1, \cdots, a_k, a_{k+1}) = \{a \in C(a_1, \cdots, a_k) | [a_1, \cdots, a_k, a_{k+1}] \cap [a] = \{e\}$ and $\mathrm{ord}(a) \leq \mathrm{ord}(a_{k+1})\}$, where a_{k+1} is the element of $C(a_1, \cdots, a_k)$ with the largest order* $\mathrm{ord}(a_{k+1})$.

Proof: Assume group G has the basis $b_1, \cdots, b_t$ with $\mathrm{ord}(b_1) \geq \cdots \geq \mathrm{ord}(b_t)$. 1) We prove it by using induction. It is trivial at the case $k = 0$. Assume that it is true at k. We consider the case at $k+1$. Let $a_1, \cdots, a_k, a_{k+1}$ be the elements of a partial basis of G. Let the $C(a_1, \cdots, a_k)$ be the complementary space for $a_1, \cdots, a_k$. By our assumption, G can be generated by $\{a_1, \cdots, a_k\} \cup C(a_1, \cdots, a_k)$. By the definition of partial basis (see Section 2), it is easy to see that $a_{k+1} \in C(a_1, \cdots, a_k)$. Select a'_{k+1} from $C(a_1, \cdots, a_k)$ such that $\mathrm{ord}(a'_{k+1}) = \max\{\mathrm{ord}(a) : a \in C(a_1, \cdots, a_k)\}$. By Lemma 4, $b'(a_1, \cdots, a_k, a'_{k+1}, b) \in C(a_1, \cdots, a_k, a'_{k+1})$ for each $b \in C(a_1, \cdots, a_k)$. We still have such a property that $\{a_1, \cdots, a_k, a'_{k+1}\} \cup C(a_1, \cdots, a_k, a'_{k+1})$ can generate G. So, $a_1, \cdots, a_k$ can be extended into basis of G: $a_1, \cdots, a_k, a'_{k+1}, \cdots, a'_{t'}$ with $\mathrm{ord}(a_1) \geq \mathrm{ord}(a_2) \geq \cdots \geq \mathrm{ord}(a_k) \geq \mathrm{ord}(a'_{k+1}) \geq \cdots \geq \mathrm{ord}(a_{t'})$ by repeating the method above. Since the decomposition of G has a unique structure (see Lemma 25), we have that $t = t'$, $\mathrm{ord}(a_1) = \mathrm{ord}(b_1), \cdots, \mathrm{ord}(a_k) = \mathrm{ord}(b_k), \mathrm{ord}(a'_{k+1}) = \mathrm{ord}(b_{k+1}), \cdots,$ and $\mathrm{ord}(a'_t) = \mathrm{ord}(b_t)$. Therefore, $\mathrm{ord}(a'_{k+1}) = \mathrm{ord}(b_{k+1}) = \mathrm{ord}(a_{k+1})$. Thus, we can select a_{k+1} instead of a'_{k+1} to extend the partial basis from $a_1, \cdots, a_k$ to $a_1, \cdots, a_k, a_{k+1}$.

2) Notice that $C(a_1, \cdots, a_k, a_{k+1}) \subseteq C(a_1, \cdots, a_k)$. It follows from the proof of 1). ∎

Lemma 7. *With $O(m)$ steps, one can compute a^p for all elements a of a G group with $m = p^i$ elements for some prime p and integer $i \geq 0$.*

Proof: Initially mark all elements of $G - \{e\}$ "unprocessed" and mark the unit element e "processed". We always select an unprocessed element $a \in G$ and compute a^p until all elements in G are processed. Compute a^p, which takes $O(\log p)$ steps, and its order $\mathrm{ord}(a) = p^j$ by trying $a^p, a^{p^2}, \cdots, a^{p^j}$, which takes $O(j^2 \log p) = O((\log p^j)^2)$ steps. Process a^k according to the order $k = 1, 2, \cdots, p^j$, compute $(a^k)^p = (a^p)^k$ in $O(p^j)$ steps and mark $a, a^2, \cdots, a^{p^j}$ "processed". For each k with $1 \leq k \leq p^j$ and $(k, p) = 1$, a^k is not processed before because the subgroups generated by a^k and a are the same (In other words, $[a^k] = [a]$). There are $p^j - p^{j-1} \geq \frac{p^j}{2}$ integers k in the interval $[1, p^j]$ to have $(k, p) = 1$. Therefore, we process at least $\frac{p^j}{2}$ new elements a^k in $O(p^j)$ steps by computing a^{kp} from a^p. Therefore, the total number of steps is $O(m)$. ∎

Lemma 8. *With $O(m)$ steps, one can compute $a^{\frac{\mathrm{ord}(a)}{p}}$ and $\log_p \mathrm{ord}(a)$ for all elements a of a group G with $m = p^i$ elements for some prime p and integer $i \geq 0$.*

Proof: We first prove that for any two elements $a, b \in G$, if $a^{p^j} = b$ for some $j \geq 0$ and $\mathrm{ord}(b) = p^t$ for some $t \geq 1$, then $\mathrm{ord}(a) = p^{j+t}$. Assume that $\mathrm{ord}(a) = p^s$. First we should notice the number j for $a^{p^j} = b$ is unique. Otherwise, $a^{p^k} \neq e$ for any integer k. This contradicts $\mathrm{ord}(a) | p^i$. Assume $a^{p^{j_1}} = a^{p^{j_2}} = b \neq e$ for some $j_1 < j_2$. Then we have $(a^{p^{j_1}})^{p^{j_2 - j_1}} = a^{p^{j_1}} \neq e$. The loop makes $a^{p^k} \neq e$ for every $k \geq 0$.

We have $a^{p^{j+t}} = (a^{p^j})^{p^t} = b^{p^t} = e$. Therefore, $s \leq j + t$. Since $a^{p^j} = b \neq e$ and $\mathrm{ord}(a) = p^s$, we have $j < s$. $b^{p^{s-j}} = (a^{p^j})^{s-j} = a^{p^s} = e$. Since $\mathrm{ord}(b) = p^t$, $t \leq s - j$ and $t + j \leq s$. Thus, we have $s = t + j$. Therefore, $\mathrm{ord}(a) = p^{j+t}$. This implies that if $a^{p^j} = b \neq e$ for some j, then $a^{\frac{\mathrm{ord}(a)}{p}} = b^{\frac{\mathrm{ord}(b)}{p}}$ and $\log_p(\mathrm{ord}(a)) = \log_p(\mathrm{ord}(b)) + j$. This fact is used in the algorithm design.

By Lemma 7, we can have a table P with $P(a) = a^p$ in $O(m)$ time. Assign flag -1 to each element in the group G in the first step. If an element a has its values $a^{\frac{\mathrm{ord}(a)}{p}}$ and $\log_p \mathrm{ord}(a)$ computed, its flag is changed to $+1$. We maintain the table that always has the property that if $a^{\frac{\mathrm{ord}(a)}{p}}$ and $\log_p \mathrm{ord}(a)$ are available (the flag of a is $+1$), then $b^{\frac{\mathrm{ord}(b)}{p}}$ and $\log_p \mathrm{ord}(b)$ are available for every $b = a^{p^j}$ for some $j > 0$. For an element b of order p^t, when computing $b^{\frac{\mathrm{ord}(b)}{p}} = b^{p^{t-1}}$, we also compute $b_i^{\frac{\mathrm{ord}(b_i)}{p}}$ and $\log_p \mathrm{ord}(b_i)$ for $b_i = b^{p^i}$ with $i = 1, 2, \cdots, t-1$ until it meets some b_i with flag $+1$. The element $b_i = b_{i-1}^p$ can be computed in $O(1)$ steps from b_{i-1} since table P is available. It is easy to see that such a property of the table is always maintained. Thus, the time is proportional to the number of elements with flag $+1$. The total time is $O(m)$. ∎

Assume the Abelian group G has p^j elements. By Lemma 8, we can set up an array $U[\,]$ of m buckets that each its position $U[g_i]$ contains all the elements a of G with $a^{\frac{\mathrm{ord}(a)}{p}} = g_i$. We also maintain a double linked list M that contains all of the elements of G with order from small to large in the first step.

Definition 9. Assume $a_1, a_2, \cdots, a_k, a_{k+1}$ are elements of an Abelian group G with p^t elements for some prime p and integer $t \geq 0$.

- Define $L(a_1, \cdots, a_k) = [a_1^{\frac{\mathrm{ord}(a_1)}{p}}, \cdots, a_k^{\frac{\mathrm{ord}(a_k)}{p}}] - \{e\}$.
- If $A = \{a_1, \cdots, a_k\}$, define $L(A) = L(a_1, \cdots, a_k)$.

Lemma 10. *Assume $a_1, a_2, \cdots, a_k, a_{k+1}$ are independent elements of G, which has p^t elements for some prime p and integer $t \geq 0$. Then 1) $L(a_1, \cdots, a_k, a_{k+1}) = L(a_1, \cdots, a_k) \cup (L(a_{k+1}) \cup (L(a_{k+1}) \circ L(a_1, \cdots, a_k)))$, and 2) $L(a_1, \cdots, a_k) \cap (L(a_{k+1}) \cup (L(a_{k+1}) \circ L(a_1, \cdots, a_k))) = \emptyset$.*

Proof: To prove 1) in the lemma, we just need to follow the definition of $L(\,)$. For 2), we use the condition $[a_{k+1}] \cap [a_1, a_2, \cdots, a_k] = \{e\}$ since $a_1, a_2, \cdots, a_k$ are independent (see the definition at Section 2). ∎

The procedure of obtaining L is shown in the following algorithm, which is also used to find the basis of the Abelian group of size power of a prime in Lemma 11.

Algorithm

Input:

an Abelian group G with size p^t, prime p and integer t,
a table T with $T(a) = a^{\frac{\mathrm{ord}(a)}{p}}$ for each $a \neq e$,
a table R with $R(a) = j$ if $\mathrm{ord}(a) = p^j$ for each $a \in G$,
an array of buckets U with $U(b) = \{a | T(a) = b\}$.
a double linked list M that contains all elements a of G with nondecreasing order by $\mathrm{ord}(a)$ (each element $a \in G$ has a pointer to the node N, which holds a, in M).

Output: the basis of G;

begin
  $L = \emptyset; B = \emptyset;$
  repeat
    select $a \in M$ with the largest $\mathrm{ord}(a)$ (a is at the end of the double linked list M);
    $B = B \cup \{a\}$;
    $L' = L(a) \cup (L(a) \circ L)$;
    for (each $b \in L'$) remove all elements in $U(b)$ from M;
    $L = L \cup L'$;
  unitl ($\sum_{a_j \in B} R(a_j) = t$);
  output the set B as the basis of G;
end

End of Algorithm

Lemma 11. *There is an $O(m)$ time algorithm for computing the basis of an G group with $m = p^t$ elements for some prime p and integer $t \geq 0$.*

Proof: The algorithm is described above the lemma. By Lemma 7, we obtain the orders of all elements of G in $O(m)$ time. With another $O(m)$ time for Bucket sorting (see [3]), we can set up the double linked list M that contains all elements a of G with nondecreasing order by $\mathrm{ord}(a)$. By Lemma 8, with $O(m)$ steps, we can obtain the table T and table R with $T(a) = a^{\frac{\mathrm{ord}(a)}{p}}$ and $R(a) = \log_p \mathrm{ord}(a)$ for each $a \neq e$ in G. With table R, we can obtain the array of buckets U with $U(b) = \{a | T(a) = b\}$ for each $b \in G$ in $O(m)$ steps by Bucket sorting. The tables T and R, bucket array U, and double linked list are used as the inputs of the algorithm.

For every element $b \in G$ with $b \neq e$, $\mathrm{ord}(b) \leq \min\{\mathrm{ord}(a_i) | i = 1, \cdots, k\}$, and $[a_1, \cdots, a_k] \cap [b] \neq \{e\}$ iff $b^{\frac{\mathrm{ord}(b)}{p}}$ is in $L(a_1, \cdots, a_k)$. When a new a_{k+1} is found, $L(a_1, a_2, \cdots, a_k)$ becomes to $L(a_1, a_2, \cdots, a_k, a_{k+1}) = L(a_1, a_2, \cdots, a_k) \cup (L(a_{k+1}) \cup L(a_{k+1}) \circ L(a_1, a_2, \cdots, a_k))$. For each new element $g_i \in L(a_{k+1}) \cup L(a_{k+1}) \circ L(a_1, a_2, \cdots, a_k) = L(a_1, a_2, \cdots, a_k, a_{k+1}) - L(a_1, a_2, \cdots, a_k)$ (see Lemma 10), we obtain the bucket $U[g_i]$ that contains all elements $a \in G$ with $a^{\frac{\mathrm{ord}(a)}{p}} = g_i$. Then remove all elements of $U[g_i]$ from the double linked list M. This makes M holds all elements of $C(a_1, \cdots, a_k, a_{k+1})$ (see Definition 5). Removing an element takes $O(1)$ time and each element is removed at most once. Therefore, the total time is $O(m)$. It is easy to check the correctness of the algorithm by using Lemma 6. ∎

Theorem 12. *There is an $O(n \log n)$ time algorithm for computing the basis of an Abelian G group with n elements.*

Proof: Assume $n = p_1^{n_1} \cdot p_2^{n_2} \cdot \cdots \cdot p_t^{n_t}$. By Lemma 1 and Lemma 3, the group G can be decomposed into product $G = G(p_1^{n_2}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t})$ in $O(n \log n)$ steps. By Lemma 11, the basis of each $G(p_i^{n_i})$ $(i = 1, 2, \cdots, t)$ can be found in $O(p_i^{n_i})$ time. Thus, the total time is $O(n \log n) + O(\sum_{i=1}^{t} p^{n_i}) = O(n \log n)$. ∎

4. Algorithm in $O(n)$ Time

In this section, we improve the running time from $O(n \log n)$ to $O(n)$ by using a result of Kavitha [9]. We obtain a linear time group decomposition $G = G(p_1^{n_1}) \circ \cdots \circ G(p_t^{n_t})$, where the Abelian group G has n elements with $n = p_1^{n_1} \cdots p_t^{n_t}$. The technique we use here is the following: For an Abelian group G with $|G| = 2^{n_1} m_2$, where m_2 is an odd number. We derive a decomposition of $G = G_1 \circ G_2$ in linear time such that $|G_1| = 2^{n_1}$ and $|G_2| = m_2$. Then we apply Kavitha's theorem to decompose the group G_2. In order to derive the elements of G_2, we convert this problem into a search problem in a special directed graph that each of its nodes has one outgoing edge. The directed graph has all elements of G as its vertices. Vertex a has edge going to vertex b if $a^2 = b$. Each weakly connected component of such a directed graph has a unique directed cycle. We show that each node in the cycle can be added to G_2. Removing the cycle nodes, we obtain a set of directed trees. The nodes that have a path of length at least n_1 to a leaf node can be also added to the group G_2. Searching the directed graph takes $O(n)$ time. Combining with Kavitha's theorem, we obtain the $O(n)$ time decomposition for the graph G. Using the result of section 3, we obtain the $O(n)$ time algorithm for finding the basis.

Theorem 13 ([9]). *Given any group G of n elements, one can compute the orders of all elements in G in $O(n \log p)$ time, where p is the smallest prime non-divisor of n.*

An undirected graph $G = (E, V)$ consists a set of nodes V and a set of undirected edges E such that the two nodes of each edge in E belong to set V. A path of G is a series of nodes $v_1 v_2 \cdots v_k$ such that $(v_i, v_i + 1)$ is an edge of G for $i = 1, \cdots, k-1$. A undirected graph is connected if every pair of nodes is linked by a path. A graph $G_1 = (E_1, V_1)$ is a subgraph of $G = (E, V)$ if $E_1 \subseteq E$ and $V_1 \subseteq V$. A connected component of G is a (maximal) subgraph $G_1 = (E_1, V_1)$ of G such that G_1 is a connected subgraph and G does not have another connected subgraph $G_2 = (E_2, V_2)$ with $E_1 \subset E_2$ or $V_1 \subset V_2$.

A directed graph $G = (E, V)$ consists of a set of nodes V and a set of directed edges E such that each edge in E starts from one node in V and ends at another node in V. A path of G is a series of nodes $v_1 v_2 \cdots v_k$ such that $(v_i, v_i + 1)$ is a directed edge of G for $i = 1, \cdots, k-1$. A (directed) cycle of G is a directed path $v_1 v_2 \cdots v_k$ with $v_1 = v_k$. For a directed graph $G = (E, V)$, let $G = (E', V)$ be the undirected graph that E' is derived from E by converting each directed edge of E into undirected edge. A directed graph $G = (E, V)$ is weakly connected if $G = (E', V)$ is connected. A subgraph $G_1 = (E_1, V_1)$ of $G = (E, V)$ is a weakly connected component of G if (E_1', V_1) is a connected component of (E', V).

We need the following lemma that shows the structure of a directed graph that each of its nodes has exactly one edge leaving it. An example about such a kind of graphs is at Figure 1.

Lemma 14. *Assume that $G = (E, V)$ is a weakly connected directed graph such that each node has exactly one outgoing edge that leaves it (and may come back to the node itself). Then the directed graph $G = (E, V)$ has the following properties: 1) Its derived undirected graph $G' = (E', V)$ has exactly one cycle. 2)G has exactly one directed cycle. 3)Every node of G is either in the directed cycle or has a directed path to a node in the directed cycle. 4)For every node v of G, if v is not in the cycle of G, then there exists a node v' in the cycle of G such that every path from v to another node v'' in the cycle of G has to go through the node v'.*

Proof: Since each node of G has exactly one edge leaving it, the number of edges in G is the same as the number of nodes. Therefore, G' can be considered to be formed by adding one edge to a tree. Clearly, G' has exactly one cycle. Therefore, G has at most one directed cycle.

Now we prove G have at least one directed cycle. We pick up a node from G. Since each node of G has exactly one edge leaving it, follow the edge leaving the node to reach another node. We will eventually come back to the node that is visited before since G has a finite number of nodes. Therefore, G has at least one cycle. Therefore, G has exactly one directed cycle. This process also shows every node of G has a directed path to a node in the directed cycle.

Assume that v is a node of G and v is not in the cycle. Let v' be the first node that v has a path to v' and the path does not visit any other node in the cycle of G. Let e be the edge leaving v'. Clearly, $H = ((E - e)', V)$ is a tree. Therefore, for every node v'' in the cycle of G, every path in $(E - e, V)$ from v to v'' has to go through v'. It is still true when e is added back since e connects v'. ∎

Lemma 15. *There exists an $O(n)$ time algorithm such that given an Abelian group G of size n, prime $p|n$, and a table H with $H(a) = a^p$, it returns two subgroups $G' = \{a \in G | a^{p^{n_1}} = e\}$ and $G'' = \{a^{p^{n_1}} | a \in G\}$ such that $|G'| = p^{n_1}$, $|G''| = m_2$ and $G = G' \circ G''$, where $n = p^{n_1} m_2$ with $(p, m_2) = 1$.*

Proof: It is easy to see that G' can be derived in $O(n)$ time since we have the table H available. By Lemma 2, we have $G = G' \circ G''$. We focus on how to generate G'' below. For each element a, set up a flag that is initially assigned -1. In order to decompose the group G into $G' \circ G''$ with $|G'| = p^{n_1}$ and $G'' = m_2$, we use Lemma 2 to build up two subsets A and B of G, where $A = \{a \in G | a^{p^{n_1}} = e\}$ and $B = \{a^{p^{n_1}} | a \in G$ and $a^{p^{n_1}} \neq e\}$. Then let $G' = A$ and $G'' = B \cup \{e\}$.

During the construction, we have the table H where $H(a) = a^p$ for every $a \in G$. We compute a^{p^j} for $j = 1, 2, \cdots, n_1$. If $a^{p^j} = e$ for some least j with $1 \le j \le n_1$, put a into A and change the flag from -1 to 1.

It is easy to see we can obtain all elements of A in $O(n)$ steps. We design an algorithm to obtain B by working on the elements in $G - A$. We build up some trees for the elements in $V_0 = G - A$.

Algorithm

Input:

group G, its size n and p with $p|n$;

table $H(\ )$ with $H(a) = a^p$ for each $a \in G$;

Output: subgroup $\{a^{p^{n_1}} | a \in G\}$;

begin

for every $a \in V_0$ with $a^p = b$ (notice $H(a) = a^p$)

begin

let (a, b) be a directed edge from a to b;

end (for)

form a directed graph (E, V_0);

let $(E_1, V_1), (E_2, V_2), \cdots, (E_m, V_m)$ be the weakly connected components of (E, V_0);

for each (E_i, V_i) with $i = 1, 2, \cdots, m$

begin

find the loop L_i, and put all elements of the loop into the set B;

for each tree in $(E_i, V_i) - L_i$ compute the height of each node;

put all nodes of height at least n_1 into B;

end (for)

output B;

end

End of Algorithm

For each component of (E, V_0), each node has only one outgoing edge. It has at most one loop in the component (see Lemma 14 for the structure of such a directed graph). The height of a node in a subtree tree, which is derived from a weakly connected component by removing a directed cycle, is the length of longest path from a leaf to it. For each node v in the cycle, clearly, there is a path $v_0 v_1 \cdots v_{n_1}$ with $v_{n_1} = v$ (notice that all the other nodes $v_0, v_1, \cdots, v_{n_1 - 1}$ are also in the cycle). So, $v \in B$. If v is not in the cycle, $v \in B$ iff there is a path with length at least n_1 and the path ends v. Since each node has one outgoing edge, each node in the cycle has no edge going out the cycle. So, a node is in B iff it has height of at least n_1 or it is in a cycle. Therefore, the set B can be derived in $O(n)$ steps by using the depth first method to scan each tree. ▌

Lemma 16. *There is an $O(n)$ time algorithm such that given a group G of size n, it returns the decomposition $G(p_1^{n_1}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t})$, where n has the factorization $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$ and $G(p_i^{n_i})$ is the subgroup of size $p_i^{n_i}$ of G for $i = 1, 2, \cdots, t$.*

Proof: For $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$, assume that $p_1 < p_2 < \cdots < p_t$. We discuss the following two cases.

Case 1: $p_1 > 2$. In this case, 2 is the least prime that is not a divisor of n. By Theorem 13, we can find the order of all elements in $O(n \log p) = O(n)$ time since $p = 2$ here. By Lemma 3, we can obtain the group decomposition in $O(n)$ time.

Case 2: $p_1 = 2$. Apply Lemma 15, we have $G = G(2^{n_1}) \circ G'$. In the next stage, we decompose G' into the production of subgroups $G' = G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t})$. Since G' does not have the divisor 2, we come back to Case 1. Clearly, the total number of steps is $O(n)$. ▌

Theorem 17. *There is an $O(n)$ time algorithm for computing the basis of an Abelian group with n elements.*

Proof: The theorem follows from Lemma 16 and Lemma 11. ▌

5. Sublinear Time Algorithm for the Basis of Abelian Group

In this section, we present a sublinear time algorithm for finding the basis of a finite Abelian group. For $n = p_1^{n_1} \cdots p_k^{n_k}$, we derive a randomized algorithm with $O((\sum_{i=1}^{k} p_i^{n_i-1} n_i \log p_i)(\log n)(\log\log n))$ running time. For this sublinear time algorithm, we always assume that the Abelian group size n and the prime factorization of n are a part of the input.

We assume that G has n elements $a_1, \cdots, a_n$ and each a_i is represented by an integer. The integer representation has the advantage in that those elements have linear order and we can use B-tree to store them so that finding and inserting can be done in $O(\log n)$ steps. We first present a sublinear time algorithm for computing the basis of a G group, which has p^t elements for some integer $t \geq 1$ and prime p.

Algorithm
Input: an Abelian group G, its size p^t, p and t;
Output: a basis of G;

Phase 0:
  If $t = 0$ then output e as the basis for G and stop the algorithm;
  Else
    Let $H_0 = \{e\}$;
    Let $m = \left\lceil \frac{x+\log t}{\log p} \right\rceil$;
    Enter Phase 1;

Phase 1:
  Randomly select m elements $b_1, \cdots, b_m$ from G.
  Compute the orders of $b_1, \cdots, b_m$ (since each $\mathrm{ord}(b_i) = p^{\eta_i}$, we just save $\log_p(\mathrm{ord}(b_i)) = \eta_i$));
  Let $a_1' = b_j$ with the largest order $\mathrm{ord}(b_j) = \max\{\mathrm{ord}(b_i)(i = 1, \cdots, m)\}$;.
  Let $E_1 = \log_p(\mathrm{ord}(a_1'))$;
  If $(E_1 < t)$ then
  Begin
    $H_1 = \{a_1^j | j = 0, \cdots, \mathrm{ord}(a_1) - 1\}$ (Save all elements of H_1 in a B-tree);
    Enter Phase 2;
  End (then)
  Else output a_1' as the basis of G and stop the algorithm;

Phase $s + 1$:
  Assume that $a_1', \cdots, a_s'$ have been found at the Phases 1 to s;
  Randomly select m elements $b_1, \cdots, b_m$ from G.
  For each b_i (($i = 1, 2, \cdots, m$)
  Begin
    For each $g \in H_k$
    Begin
      Let $b = gb_i$;
      Compute the order p^u for b and the set $B = \{b, b^p, b^{p^2}, \cdots, b^{p^{u-1}}\}$;
      (save $\log_p(\mathrm{ord}(b)) = u$)
      If $(\mathrm{ord}(b) \leq \mathrm{ord}(a_k'))$ and $(B \cap H_s = \emptyset)$) Then
      Begin
        Let $b_i' = gb_i$;
        Goto L;
      End (if)
    End (for)
   L: Continue;
  End (for)
  Let $a_{s+1}' = b_j'$ which has the largest order $\mathrm{ord}(b_j') = \max\{\mathrm{ord}(b_i')(i = 1, \cdots, m)\}$;
  Let $E_{s+1} = E_s + \log_p(\mathrm{ord}(a_{s+1}'))$;
  If $(E_{s+1} < t)$ then
  Begin
    $H_{s+1} = \{a_{s+1}'^j h | h \in H_s \text{ and } j = 0, \cdots, \mathrm{ord}(a_{s+1}') - 1\}$
    (Save all elements of H_{s+1} in a B-tree);
    Enter the phase $s + 2$;
  End (Then)

Else output $a'_1, a'_2, \cdots, a'_{s+1}$ as the basis for G and stop the algorithm.

End of Algorithm

Lemma 18. *There exists a randomized algorithm such that given an Abelian group G of size p^t, p, $t \geq 0$, and integer $x > 0$, it runs in $O(p^{t-1}(x+\log t)t^2 \log p)$ steps, uses at most $\left\lceil \frac{x+\log t}{\log p} \right\rceil t$ random elements selected from G, and computes its basis with a failure probability at most $\frac{1}{2^x}$.*

Proof: Assume that $a_1, \cdots, a_k$ form a basis of G with orders $\mathrm{ord}(a_1) \geq \mathrm{ord}(a_2) \geq \cdots \geq \mathrm{ord}(a_k)$. Our algorithm finds a basis $\{a'_1, a'_2, \cdots, a'_k\}$ of G with $\mathrm{ord}(a'_1) = \mathrm{ord}(a_1), \mathrm{ord}(a'_2) = \mathrm{ord}(a_2), \cdots$, and $\mathrm{ord}(a'_k) = \mathrm{ord}(a_k)$.

If $m \geq \frac{x+\log t}{\log p}$, then $m \log p \geq x + \log t$, which implies that $\frac{1}{p^m} \leq \frac{1}{t 2^x}$. The algorithm is described right above the lemma.

Phase 0: If $t = 0$ then output e as the basis for G and stop the algorithm. Otherwise, let $H_0 = \{e\}$ and $m = \left\lceil \frac{x+\log t}{\log p} \right\rceil$. Then enter Phase 1. We set $m = \left\lceil \frac{x+\log t}{\log p} \right\rceil$ in the algorithm for the number of random sample elements selected from G to find one element in the basis of G in the coming phases.

Phase 1: Randomly select m elements $b_1, \cdots, b_m$ from G. Assume that $b_i = a_1^{\eta_i} c_i$, where $c_i \in [a_2, \cdots, a_k]$ and η_i is an integer in the interval $[0, \mathrm{ord}(a_1) - 1]$. If $(\eta_i, p) = 1$, then $\mathrm{ord}(b_i) = \mathrm{ord}(a_1)$ (notice that $\mathrm{ord}(a_1) = p^j$ for some integer $j \geq 1$ and $\mathrm{ord}(a_1) \geq \mathrm{ord}(a_2) \geq \cdots \geq \mathrm{ord}(a_k)$). Since b_i is a random element in G, η_i is a random number in $[0, \mathrm{ord}(a_1) - 1]$ and the probability is $\frac{1}{\mathrm{ord}(a_1)}$ that η_i is equal to any integer in $[0, \mathrm{ord}(a_1) - 1]$. Assume that $\mathrm{ord}(a_1) = p^{j_1}$. There are $p^{j_1 - 1}$ integers i in $[0, \mathrm{ord}(a_1) - 1]$ with $(i, p) \neq 1$. With probability at most $\frac{p^{j_1-1}}{p^{j_1}} = \frac{1}{p}$, $(\eta_i, p) \neq 1$. With probability at most $\frac{1}{p^m}$, $(\eta_i, p) \neq 1$ for every $i = 1, \cdots, m$. Therefore, with probability at most $\frac{1}{p^m}$, $\max\{\mathrm{ord}(b_1), \cdots, \mathrm{ord}(b_m)\} < \mathrm{ord}(a_1)$.

Computing the orders of $b_1, \cdots, b_m$ takes $O(m \cdot t^2 \log p)$ steps. This is because for each b_i $(i = 1, \cdots, m)$, we compute $b_i^p, b_i^{p^2}, \cdots, b_i^{p^t}$ and take $O(\log p^t)$ steps for each of them. Let $a'_1 = b_i$ with $\mathrm{ord}(b_i) = \max\{\mathrm{ord}(b_j) | j = 1, \cdots, m\}$. Clearly, with probability at most $\frac{1}{p^m}$, $\mathrm{ord}(a'_1) = \mathrm{ord}(a_1)$ is not true. At the end of this phase, the algorithm checks if $G = [a'_1]$, which is equivalent to $\mathrm{ord}(a'_1) = p^t$ or $\log_p \mathrm{ord}(a'_1) = t$. If not, it generates $H_1 = [a'_1]$ (all elements of H_1 are stored in a B-tree), and then enter Phase 2.

Phase $s + 1$: Assume that $a'_1, \cdots, a'_s$ have been obtained such that $\mathrm{ord}(a'_i) = \mathrm{ord}(a_i)$ for $i = 1, \cdots, s$, $a'_1, \cdots, a'_s$ are independent, and $H_s = [a'_1, \cdots, a'_s]$. We will find a'_{s+1} in this phase. By Lemma 6, there are $a''_{s+1}, \cdots, a''_k$ such that $a'_1, \cdots, a'_s, a''_{s+1}, \cdots, a''_k$ form the basis of G with $\mathrm{ord}(a'_1) = \mathrm{ord}(a_1), \mathrm{ord}(a'_2) = \mathrm{ord}(a_2), \cdots$, $\mathrm{ord}(a'_s) = \mathrm{ord}(a_s)$, $\mathrm{ord}(a''_{s+1}) = \mathrm{ord}(a_{s+1}), \cdots$, and $\mathrm{ord}(a''_k) = \mathrm{ord}(a_k)$. If b is a random element from G, then $b = a''^{\eta}_{s+1} c$ and η is a random integer in $[0, \mathrm{ord}(a''_{s+1}) - 1]$, where $c \in [a'_1, \cdots, a'_s, a''_{s+2}, a''_{s+3}, \cdots, a''_k]$ and η is an integer in $[0, \mathrm{ord}(a_{s+1}) - 1]$.

Randomly select m elements $b_1, \cdots, b_m$ from G. Let $b_i = a''^{\eta_i}_{s+1} c_i$ and η_i be a random integer in $[0, \mathrm{ord}(a''_{s+1}) - 1]$, where $c_i \in [a'_1, \cdots, a'_s, a''_{s+2}, a''_{s+3}, \cdots, a''_k]$. Similar to Phase 1, the probability is at most $\frac{1}{p^m}$ that $(\eta_i, p) \neq 1$ for every $i = 1, \cdots, m$.

For each b_u, we can always find another $g \in H_s$ such that $\mathrm{ord}(b_u g) \leq \mathrm{ord}(a'_s)$ and $([a'_1, \cdots, a'_s]) \cap [b_u g] = \{e\}$. This is because we can let $g = a'^{-j_1}_1 \cdots a'^{-j_s}_s$ when $b_u = a'^{j_1}_1 \cdots a'^{j_s}_s a''^{j_{s+1}}_{s+1} \cdots a''^{j_k}_k$.

Assume that $g \in [a'_1, \cdots, a'_s]$ and $[g b_u] \cap H_s = \{e\}$. Let $g b_u = \prod_{i=1}^{s} a'^{t_i p^{\xi_i}}_i \prod_{j=s+1}^{s} a''^{t_j p^{\xi_j}}_j$, where $(t_i, p) = 1$ for $i = 1, \cdots, k$. We claim that (a) $\max\{\mathrm{ord}(a'^{t_i p^{\xi_i}}_i) | i = 1, \cdots, s\} \leq \max\{\mathrm{ord}(a''^{t_j p^{\xi_j}}_j) | j = s+1, \cdots, k\}$; and (b) $\mathrm{ord}(g b_u) = \max\{\mathrm{ord}(a''^{t_j p^{\xi_j}}_j) | j = s+1, \cdots, k\}$. Assume (a) is not true. We have that $\max\{\mathrm{ord}(a'^{t_i p^{\xi_i}}_i) | i = 1, \cdots, s\} > \max\{\mathrm{ord}(a''^{t_j p^{\xi_j}}_j) | j = s+1, \cdots, k\}$. Let $\max\{\mathrm{ord}(a''^{t_j p^{\xi_j}}_j) | j = s+1, \cdots, k\} = p^{\xi}$. Then $(\prod_{j=s+1}^{s} a''^{t_j p^{\xi_j}}_j)^{p^{\xi}} = e$ and $(\prod_{i=1}^{s} a'^{t_i p^{\xi_i}}_i)^{p^{\xi}} \neq e$. Thus, $e \neq (g b_u)^{p^{\xi}} \in H_s$ (recall $H_s = [a_1, \cdots, a_s]$). This contradicts that $[g b_u] \cap H_s = \{e\}$. Therefore, (a) is true. (b) follows from (a).

If $b_i = a''^{\eta_i}_{s+1} c_i$ with $c_i \in [a'_1, \cdots, a'_s, a''_{s+2}, a''_{s+3}, \cdots, a''_k]$. We find $b'_i = g b_i$ such that $\mathrm{ord}(g b_i) \leq \mathrm{ord}(a'_s)$ and $[g b_i] \cap H_s = \{e\}$. The $s+1$-th element a'_{s+1} is selected to be b'_j with $\mathrm{ord}(b'_j) = \max\{\mathrm{ord}(b'_i)(i = 1, \cdots, m)\}$. If $(\eta_i, p) = 1$, then $\mathrm{ord}(b'_i) = \mathrm{ord}(a''_{s+1}) = \mathrm{ord}(a_{s+1})$. We already know that the probability is at most $\frac{1}{p^m}$ that $(\eta_i, p) \neq 1$ for every $i = 1, \cdots, m$. So, with probability at most $\frac{1}{p^m}$, $\mathrm{ord}(a'_{s+1}) \neq \mathrm{ord}(a_{s+1})$. At the end of this phase, the algorithm checks if $G = [a'_1, a'_2, \cdots, a'_s, a'_{s+1}]$, which is equivalent to that $\log_p(\mathrm{ord}(a'_1)) + \log_p(\mathrm{ord}(a'_2)) + \cdots + \log_p(\mathrm{ord}(a'_{s+1})) = t$. If not, it generates $H_{s+1} = H_s \circ [a'_{s+1}] = [a'_1, \cdots, a'_s, a'_{s+1}]$ (all elements of H_{s+1} are stored in a B-tree) and enter Phase $s + 2$.

Assume the algorithm stops at Phase $z + 1$. The basis generated by the algorithm is $a'_1, \cdots, a'_z, a'_{z+1}$. So, $H_1, H_2, \cdots, H_z$ have been generated with $H_1 \subset H_2 \subset \cdots \subset H_z \subset G$. The size of H_z is strictly less than that of the group G. It is easy to see that $H_i = [a'_1, a'_2, \cdots, a'_i]$, which is the subgroup generated by the part of elements that have been found from Phase 1 to Phase i for $(i = 1, \cdots, z)$. Thus, $|H_i| = \prod_{j=1}^{i} \mathrm{ord}(a'_j)$. The algorithm stops at Phase $z+1$, which has found the full basis $a'_1, \cdots, a'_{z+1}$ and it does not generate H_{z+1} any more. This is why we use less time than linear. It is easy to see that $|H_z| \le p^{t-1}$ and $|H_{y-1}| \le \frac{|H_y|}{p}$ for $y = z, z-1, \cdots, 2, 1$. We have $|H_z| + |H_{z-1}| + \cdots + |H_1| = O(p^{t-1} + \cdots + p^2 + p) = O(\frac{p^t-1}{p-1}) = O(p^{t-1})$.

For each b_i, it takes $|H_s|$ steps to generate all $b = gb_i$ for all $g \in H_s$. For each $b = gb_i$, it takes $O(t \log p^t) = O(t^2 \log p)$ steps to compute its order $\mathrm{ord}(b) = p^u$ and the set $B = \{b, b^p, b^{p^2}, \cdots, b^{p^{u-1}}\}$ in the algorithm. It is easy to see that $H_s \cap [b] = \{e\}$ if and only if $H_s \cap B = \emptyset$. It takes another $O(t \log p^t) = O(t^2 \log p)$ steps for checking if $B \cap H_s = \emptyset$, which needs to use at most t finding operations to a B-tree with at most p^t elements. It takes $O(|H_s| t^2 \log p)$ steps to compute one b'_i. So, it takes $O(|H_s| m t^2 \log p)$ steps to compute all b'_is for $i = 1, 2, \cdots, m$.

The total running time is $O(|H_1| + \cdots + |H_z|) m t^2 \log p) = O(\frac{p^{t-1}(x+\log t) t^2 \log p}{\log p}) = O(p^{t-1}(x + \log t) t^2 \log p)$.

With probability at most $\frac{1}{p^m}$, one phase fails. There are at most t phases since each phase generates a new element in the basis and the group has size p^t. The total probability of failure is at most $\frac{t}{p^m} \le 2^x$ by the setting of m. ∎

Lemma 19. *Assume $n = p_1^{n_1} \cdots p_k^{n_k}$ and G is an Abelian group of n elements. Let $m_i = \frac{n}{p_i^{n_i}}$ for $i = 1, \cdots, k$. If a is a random element of G that with probability $\frac{1}{|G|}$, a is equal to b for each $b \in G$, then a^{m_i} is a random element of $G(p_i^{n_i})$, the subgroup of G with p^{n_i} elements, such that with probability $\frac{1}{p_i^{n_i}}$, a^{m_i} is b for any $b \in G(p_i^{n_i})$*

Proof: Let $b_{i,j}$ ($j = 1, \cdots, k_i$) be the basis of $G(p_i^{n_i})$, i.e. $G(p_i^{n_i}) = [b_{i,1}] \circ \cdots \circ [b_{i,k}]$. Let a be a random element in G. Let $a = (\prod_{j=1}^{k_i} b_{i,j}^{c_{i,j}}) a'$, where a' is an element in $\prod_{j \ne i} G(p_j^{n_j})$. For every two integers $x \ne y \in [0, p_i^{n_i} - 1]$, $m_i x \ne m_i y (\bmod\ p_i^{n_i})$ (Otherwise, $m_i x = m_i y (\bmod\ p_i^{n_i})$ implies $x = y$ because $(m_i, p_i) = 1$). So, the list of numbers $m_i \cdot 0 (\bmod\ o(p_i^t)), m_i \cdot 1 (\bmod\ o(p_i^t)), \cdots, m_i(p_i^t - 1)(\bmod\ o(p_i^t))$ is a permutation of $0, 1, \cdots, p_i^t - 1$. So, if $c_{i,j}$ is a random integer in the range $[0, \mathrm{ord}(b_{i,j}) - 1]$ such that with probability $\frac{1}{\mathrm{ord}(b_{i,j})}$, $c_{i,j} = c'$ for each $c' \in [0, \mathrm{ord}(b_{i,j}) - 1]$, then the probability is also $\frac{1}{\mathrm{ord}(b_{i,j})}$ that $m_i c_{i,j} = c'$ for each $c' \in [0, \mathrm{ord}(b_{i,j}) - 1]$. Therefore, $a^{m_i} = ((\prod_{j=1}^{k_i} b_{i,j}^{c_{i,j}}) a')^{m_i} = \prod_{j=1}^{k_i} b_{i,j}^{m_i c_{i,j}}$, which is a random element in $G(p_i^{n_i})$. ∎

Theorem 20. *There exists a randomized algorithm such that given an Abelian group G of size n with $n = p_1^{n_1} \cdots p_k^{n_k}$, the algorithm computes the basis of G in $O((\sum_{i=1}^k p_i^{n_i - 1} n_i^2 \log p_i)(\log n) \log \log n)$ running time.*

Proof: Let $x = 3 \log \log n$. Then $\frac{1}{2^x} < \frac{1}{(\log n)^2}$. It takes $O(\log n)$ steps for computing a^{m_i} for an element $a \in G$, where $m_i = \frac{n}{p_i^{n_i}}$. Each random element of G can be converted into a random element of $G(p_i^{n_i})$ by Lemma 19. Each $G(p_i^{n_i})$ needs $O(x + \log n_i) n_i$ random elements by Lemma 18. Each $G(p_i^{n_i})$ needs $O((x + \log n_i) n_i \log n)$ time to convert the $(x + \log n_i) n_i$ random elements from G to $G(p_i^{n_i})$. It takes $O(\sum_{i=1}^k (x + \log n_i) n_i \log n))$ time to convert random elements of G into the random elements in all subgroups $G(p_i^{n_i})$ for $i = 1, \cdots, k$. For $n = p_1^{n_1} \cdots p_k^{n_k}$, $\sum_{i=1}^k n_i \log p_i = \log n$. Furthermore, $x + \log n_i = O(\log \log n)$. By Lemma 18, the sum of time for all $G(p_i^{n_i})$s to find basis is $O((\sum_{i=1}^k p_i^{n_i - 1} (\log \log n + \log n_i) n_i^2 \log p_i)(\log n)) = O((\sum_{i=1}^k p_i^{n_i - 1} n_i^2 \log p_i)(\log n) \log \log n)$

It follows from Lemma 19 and Lemma 18. The group size n has at most $\log n$ prime factors. Since each $G(p_i^{n_i})$ has a probability at most $\frac{1}{2^x}$ of failure, the total probability of failure is at most $\frac{\log n}{2^x} \le \frac{1}{\log n}$. ∎

Corollary 21. *There exists a randomized algorithm with $O(n^{1-\frac{1}{d}} (\log n)^3 (\log \log n))$ running time that given an Abelian group G of size $n = p_1^{n_1} \cdots p_k^{n_k}$, it computes the basis of G, where $d = \max\{n_i | i = 1, \cdots, k\}$.*

Proof: For $n = p_1^{n_1} \cdots p_k^{n_k}$, $\sum_{i=1}^k n_i \log p_i = \log n$ and $p_i^{n_i - 1} \le n^{1-\frac{1}{d}}$. By Theorem 20, the running time is $O((\sum_{i=1}^k p_i^{n_i - 1} n_i^2 \log p_i)(\log n) \log \log n) = O(n^{1-\frac{1}{d}} (\log n)^3 (\log \log n))$. ∎

Definition 22. For an integer n, define $F(n) = \max\{p^{i-1} | p^i | n,\ p^{i+1} \not| n,\ i \ge 1$, and p is a prime $\}$. Define $G(m, c)$ as the set of all integers in $[1, m]$ with $F(n) \ge (\log n)^c$ and $H(m, c) = |G(m, c)|$.

Theorem 23. $\frac{H(m,c)}{m} = O(\frac{1}{(\log m)^{c/2}})$ *for every constant $c > 0$.*

Proof: $H(m,c)$ is the number of integers in $G(m,c)$, which is a subset of integers in $[1,m]$. We discuss the three cases.

The number of integers in the interval $[1, \frac{m}{(\log m)^{c/2}}]$ is at most $\frac{m}{(\log m)^{c/2}}$. We only consider the numbers in the range $I = [\frac{m}{(\log m)^{c/2}}, m]$. It is easy to see for every integer $n \in I$, $2(\log n)^c \geq (\log m)^c$ for all large m since c is fixed. We consider each number $n \in I$ such that $p^t | n$ with $p^t \geq \frac{(\log m)^c}{2}$ for some prime p.

For each prime number $p \in [2, (\log m)^{c/2}]$, let t be the least integer with $p^t \geq \frac{(\log m)^c}{2}$. We count the number of integers $n \in I$ such that $p^u | n$ for some $u \geq t$. The number is at most $\frac{m}{p^t} + \frac{m}{p^{t+1}} + \cdots \leq \frac{m}{p^t}(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots) \leq \frac{2m}{p^t} \leq \frac{4m}{(\log m)^c}$. Therefore, it has at most $(\log m)^{c/2} \cdot \frac{2m}{(\log m)^c} \leq \frac{4m}{(\log m)^{c/2}}$ integers $n \in I$ to have $p^t | n$ with $p^t \geq \frac{(\log m)^c}{2}$.

Let's consider the cases $p^t | n$ for $p > (\log m)^{c/2}$ and $t \geq 2$. The number of integers $n \in I$ for a fixed p with $p^2 | n$ is at most $\frac{m}{p^2} + \frac{m}{p^3} + \cdots \leq \frac{2m}{p^2}$. The total number of integers $n \in I$ that have $p^2 | n$ for some prime number $p > (\log m)^{c/2}$ is at most $\frac{2m}{(1+(\log m)^{c/2})^2} + \frac{2m}{(2+(\log m)^{c/2})^2} + \cdots < \frac{2m}{((\log m)^{c/2})(1+(\log m)^{c/2})} + \frac{2m}{((1+(\log m)^{c/2})(2+(\log m)^{c/2})^2} + \cdots \leq \frac{2m}{(\log m)^{c/2}}$. Combining the cases above, we have $\frac{H(m,c)}{m} = O(\frac{1}{(\log m)^{c/2}})$. ∎

Theorem 20 and Theorem 23 imply the following theorem:

Theorem 24. *There exists a randomized algorithm such that if n is in $[1,m] - G(m,c)$, then the basis of an Abelian group of size n whose prime factorization is also part of the input can be computed in $O((\log n)^{c+3} \log\log n)$-time, where c is any positive constant and $G(m,c)$ is a subset of integers in $[1,m]$ with $\frac{|G(m,c)|}{m} = O(\frac{1}{(\log m)^{c/2}})$ for each integer m.*

6. Conclusion

In this paper, we obtained an $O(n)$-time deterministic algorithm for computing the basis of an Abelian group with n elements. We also show that there exists a randomized algorithm such that for each integer $n \in [1,m] - G(m,c)$, the basis of an Abelian group of size n can be computed in $(\log n)^{O(1)}$ time where c is a constant and m is any integer. The subset $G(m,c) \subset [1,m]$ only has a small fraction of integers in $[1,m]$. We also show that there exists a randomized algorithm with $O(n^{1-\frac{1}{d}} (\log n)^3 \log\log n)$ time complexity such that given an Abelian group G of size $n = p_1^{n_1} \cdots p_k^{n_k}$, the algorithm computes the basis of G, where $d = \max\{n_i | i = 1, \cdots, k\}$. This algorithm is a sublinear time algorithm if $d = \max\{n_i | i = 1, \cdots, k\}$ is bounded by a constant. An interesting open problem is to find an $O(n^{1-\epsilon})$-time algorithm to compute the basis of an Abelian group of n elements, where ϵ is a constant independent of n.

References

[1] L. Chen. Algorithms and their complexity analysis for some problems in finite group. *Journal of Sandong Normal University, in Chinese*, 2:27–33, 1984.

[2] K. Cheung and M. Mosca. Decomposing finite abelian groups. *Quantum Information and Computation*, 1:26–32, 2001.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, 2001.

[4] Y. G. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4):667–679, 1994.

[5] M. Garzon and Y. Zalcstein. On isomorphism testing of a class of 2-nilpoten groups. *Journal of Computer and System Sciences*, 42:237–248, 1991.

[6] C. M. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism.* Springer-Verlag, 1982.

[7] T. Hungerford. *Algebra.* Springer-Verlag, 1974.

[8] M. I. Kargapolov and J. I. Merzljako. *Fundamentals of the Theory of Groups.* Springer-Verlag, 1979.

[9] T. Kavitha. Efficient algorithms for abelian group isomorphism and related problems. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science, Lecture notes in computer science, 2914*, pages 277–288, 2003.

[10] A. Y. Kitaev. Quantum computations: Algorithms and error correction. *Russian Math. Surveys*, 52:1191, 1997.

[11] J. Köbler, U. Schöning, and J. Toran. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhouser, 1993.

[12] C. Lomont. The hidden subgroup problem -review and open problems. http://arxiv.org/abs/quant-ph/0411037, 2004.

[13] A. Menezes. Elliptic curve cryptosystems. *CryptoBytes*, 1:1–4, 1995.

[14] G. L. Miller. On the $n^{\log n}$ isomorphism technique. In *In proceedings of the tenth annual ACM symposium on theory of computing*, pages 128–142, 1978.

[15] G. L. Miller. Graph isomorphism, general remarks. *ournal of Computer and System Sciences*, 18:128–142, 1979.

[16] V. Miller. Uses of elliptic curves in cryptography. In *Advances in Cryptology CRYPTO'85, Lecture Notes in Computer Science*, pages 417–426, 1986.

[17] C. Savage. An $O(n^2)$ algorithm for abelian group isomorphism. Technical report, North Carolina State University, January 1980.

[18] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:14841509, 1997.

[19] D. R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:1474 1483, 1997.

[20] N. Vikas. An $O(n)$ algorithm for abelian p-group isomorphism and an $O(n \log n)$ algorithm for abelian group isomorphism. *Journal of Computer and System Sciences*, 53:1–9, 1996.

Figure 1: Each node has one outgoing edge

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/