Theoretical Computer Science 412 (2011) 4110–4122

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Linear and sublinear time algorithms for the basis of abelian groups[☆]

Li Chen[a,1], Bin Fu[b,*]

[a] Department of Computer Science, University of District of Columbia, Washington, DC 20008, USA

[b] Department of Computer Science, University of Texas-Pan American, Edinburg, TX 78539, USA

ARTICLE INFO

Keywords:
Abelian group
Randomization
Decomposition
Basis of Abelian group

ABSTRACT

It is well known that every finite abelian group G can be represented as a direct product of cyclic groups: $G \cong G_1 \times G_2 \times \cdots \times G_t$, where each G_i is a cyclic group of order p^j for some prime p and integer $j \geq 1$. If a_i generates the cyclic group of G_i, $i = 1, 2, \ldots, t$, then the elements $a_1, a_2, \ldots, a_t$ are called a basis of G. We show a randomized algorithm such that given a set of generators $M = \{x_1, \ldots, x_k\}$ for an abelian group G and the prime factorization of order $\mathrm{ord}(x_i)$ $(i = 1, \ldots, k)$, it computes a basis of G in $O(|M|(\log n)^2 + \sum_{i=1}^{t} n_i p_i^{n_i/2})$ time, where $n = |G|$ has prime factorization $p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$ (which is not a part of input). This generalizes Buchmann and Schmidt's algorithm that takes $O(|M|\sqrt{|G|})$ time. In another model, all elements in an abelian group are put into a list as a part of input. We obtain an $O(n)$ time deterministic algorithm and a sublinear time randomized algorithm for computing a basis of an abelian group.

1. Introduction

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) [9]. The set of generators with exactly one from each of those cyclic groups forms a basis of the abelian group. Because a basis of an abelian group fully determines its structure, which is the nondecreasing orders of the elements in a basis, finding a basis is crucial in computing the general properties for abelian groups. The orders of all elements in a basis form the invariant structure of an abelian group. There is a long line of research about the algorithm for determining group isomorphism (e.g. [14,8,12,13,16,20,10,6,11]). Two abelian groups are isomorphic if and only if they have the same structure.

For finding a basis of abelian group, Chen [4] showed an $O(n^2)$ time algorithm for finding a basis of an abelian group G given all elements and size of G as input. An abelian group is often represented by a set of generators in the field of computational group theory (e.g., [18]) as a set of generators costs a small amount of memory. The algorithm for the basis of the abelian group with a set of generators as input was developed by Buchmann, et al. [2], Teske [19], and Buchmann and Schmidt [3] with the fastest proven time $O(m\sqrt{|G|})$. The methods for computing the order for one element in a group are also connected with computing the abelian basis, which was also reported in [2,17].

We show a randomized algorithm such that given a set of generators $M = \{x_1, \ldots, x_k\}$ for an abelian group G and the prime factorization of order $\mathrm{ord}(x_i)$ $(i = 1, \ldots, k)$, it computes a basis of G in $O(|M|(\log n)^2 + \sum_{i=1}^{t} n_i p_i^{n_i/2})$ time, where $n = |G|$ has prime factorization $p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$ (which is not a part of input). This implies an algorithm such that given an abelian group G represented by a set of generators $M = \{x_1, \ldots, x_k\}$ without their orders information, it computes a basis

☆ This research is supported in part by NSF Early Career Award 0845376.

* Corresponding author. Tel.: +1 956 381 3635; fax: +1 956 384 5099.
E-mail addresses: lchen@udc.edu (L. Chen), binfu@cs.panam.edu (B. Fu).

1 Tel.: +1 202 274 6301.

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2010.06.011

of G in $O(|M|(\log n)^2 + \sum_{i=1}^{t} n_i p_i^{n_i/2} + (\sum_{i=1}^{t} \sqrt{\mathrm{ord}(x_i)}))$ time. This improves Buchmann and Schmidt's algorithm that takes $O(|M|\sqrt{|G|})$ time.

In the model of all elements in an abelian group being put into a list as a part of input, we derive an $O(\sum_{i=1}^{t} n_i \min(p_i^{n_i/2}, p_i^{n_i-1}) + \sum_{i=1}^{t} n_i \log n)$-time randomized algorithm to compute a basis of abelian group G of order n with factorization $n = p_1^{n_1} \cdots p_t^{n_t}$, which is also a part of the input. It implies an $O(n^{1/2} \sum_{i=1}^{t} n_i)$-time randomized algorithm to compute a basis of an abelian group G of order n. It also implies that if n is an integer in $\{1, 2, \ldots, m\} - G(m, c)$, then a basis of an abelian group of order n can be computed in $O((\log n)^{c+1})$-time, where c is any positive constant and $G(m, c)$ is a subset of the small fraction of integers in $\{1, 2, \ldots, m\}$ with $\frac{|G(m,c)|}{m} = O(\frac{1}{(\log m)^{c/2}})$ for every integer m. We show an algorithm such that given a set of generators $M = \{x_1, \ldots, x_k\}$ for an abelian group G and the prime factorizations of orders $\mathrm{ord}(x_i)$ ($i = 1, \ldots, k$), it computes a basis of G in $O(|M|(\sum_{i=1}^{t} p_i^{n_i/2}))$ time, where $n = |G|$ has prime factorization $p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$ (which is not a part of input). We also obtain an $O(n)$-time deterministic algorithm for computing a basis of an abelian group with n elements. The existing algorithms need $O(n^2)$ time by Chen and $O(n^{1.5})$ time by Buchmann and Schmidt.

In Section 4, we give a randomized algorithm to compute a basis of an abelian group given a set of generators as input. In Section 5, we give a randomized algorithm to compute a basis of an abelian group given the entire group as input. In Section 6, we give a deterministic algorithm to compute a basis of an abelian group given the entire group as input. We consider Theorems 7 and 17 as two main theorems of this paper. In all algorithms, the multiplication table of an abelian group is accessed as a black box and no inverse operation is used.

2. Notations

For two positive integers x and y, (x, y) represents the greatest common divisor (GCD) between them. For a set A, $|A|$ denotes 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 smallest 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 contains 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 $\langle a \rangle$ to be the subgroup of G generated by the element a (in other words, $\langle a \rangle = \{e, a, a^2, \ldots, a^{\mathrm{ord}(a)-1}\}$). Let A and B be two subsets of group G, define $AB = A \cdot B = A \circ B = \{ab | a \in A \text{ and } b \in B\}$. We use $\cong$ to represent the isomorphism between two groups.

A group G is an *abelian* group if $ab = ba$ for every pair of elements $a, b \in G$. Assume that G is an abelian group with elements $g_1, g_2, \ldots, g_n$. For each element $g_i \in G$, it corresponds to an index i. According to the theory of abelian group, a finite abelian group G of n elements can be represented as $G = G(p_1^{n_1}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t}) \cong G(p_1^{n_1}) \times G(p_2^{n_2}) \times \cdots \times G(p_t^{n_t})$, 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 [9]). We also use the notation G_{p_i} to represent the subgroup of G with order $p_i^{n_i}$. Any abelian group G of order p^m can be represented by $G = G(p^{m_1}) \circ G(p^{m_2}) \circ \cdots \circ G(p^{m_k}) \cong G(p^{m_1}) \times G(p^{m_2}) \times \cdots \times G(p^{m_k})$, 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, $a_1, a_2, \ldots, a_k$ from the abelian group G, denote $\langle a_1, a_2, \ldots, a_k \rangle$ to be the set of all elements in G *generated* by $a_1, \ldots, a_k$. In other words, $\langle a_1, a_2, \ldots, a_k \rangle = \langle a_1 \rangle \langle a_2 \rangle \cdots \langle a_k \rangle$. An element $a \in G$ is *independent of* $a_1, a_2, \ldots, a_k$ in G if $a \ne e$ and $\langle a_1, a_2, \ldots, a_k \rangle \cap \langle a \rangle = \{e\}$. If $G = \langle a_1, a_2, \ldots, a_k \rangle$, then $\{a_1, a_2, \ldots, a_k\}$ is called a set of *generators* of G. If X is a set of elements in G, we also use $\langle X \rangle$ to represent the subgroup generated by set X.

The elements $a_1, a_2, \ldots, a_k$ from the abelian group G are *independent* if a_i is independent of $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k$ for every i with $1 \le i \le k$. A basis of G is a set of independent elements $a_1, \ldots, a_k$ that can generate all elements of G (in other words, $G = \langle a_1, a_2, \ldots, a_k \rangle$).

3. Overview of our methods

For an abelian group G with $n = p_1^{n_1} \times p_2^{n_2} \cdots \times p_t^{n_t}$ elements, it can be decomposed into product $G(p_1^{n_1}) \circ G(p_2^{n_2}) \circ \cdots \circ G(p_t^{n_t}) \cong G(p_1^{n_1}) \times G(p_2^{n_2}) \times \cdots \times G(p_t^{n_t})$, where each $G(p_i^{n_i})$ is a subgroup of G of order $p_i^{n_i}$. The problem for finding a basis of G is converted into the problem for finding a basis of every subgroup $G(p_i^{n_i})$ $i = 1, 2, \ldots, t$. The union of those basis for $G(p_i^{n_i})$ ($i = 1, 2, \ldots, t$) forms a basis of G. This decomposition method is used in every algorithm of this paper.

4. Randomized algorithm for basis via generators

An abelian group is often represented by a set of generators. The set of generators for a group is usually much less than the order of a group. It is important to find the algorithm for computing a basis of abelian group represented by a set of generators. The randomized algorithms in this paper belong to Monte Carlo algorithms [1], which have a small probability to output error results.

Let $B = \{b_1, \ldots, b_k\}$ be a set of basis for an abelian group G of size p^m (p is a prime) and assume that $\mathrm{ord}(b_1) \leq \mathrm{ord}(b_2) \leq \cdots \leq \mathrm{ord}(b_k)$. The *structure* of G is defined by $\langle \mathrm{ord}(b_1), \mathrm{ord}(b_2), \ldots, \mathrm{ord}(b_k)\rangle$. We note that the structure of an abelian group is invariant, but its basis is not unique.

The theorem of Buchmann and Schmidt [3] is used in our algorithm for finding a basis of abelian group. The following Theorem 1 follows from Lemma 3.1 and Theorem 3.4 in [3].

Theorem 1 ([3]). *There exists an $O(m\sqrt{|G|})$ time algorithm such that given a set of generators of order m for an abelian group G of order p^t for some prime number p and integer $t \geq 1$, the algorithm returns a basis and the structure of G in $O(m\sqrt{|G|})$ steps.*

Theorem 2 ([3]). *There exists an algorithm such that given an element g of an abelian group G, it returns $\mathrm{ord}(g)$ in $O(\sqrt{\mathrm{ord}(g)})$ steps.*

Lemma 3. *Assume G is an abelian group of order 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 $(\mathrm{ord}(a), m_1) = 1$, 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 $i = 1, \ldots, t$.*

Proof. It is easy to verify that G' is subgroup of G. Assume $a_1, \ldots, a_{s_1}, b_1, \ldots, b_{s_2}$ are the elements in a basis of G such that $\mathrm{ord}(a_i)|m_1$ for $i = 1, \ldots, s_1$ and $\mathrm{ord}(b_j)|m_2$ for $j = 1, \ldots, s_2$. It is easy to see that $a_i^{m_1} = e$ for $i = 1, \ldots, s_1$ and $b_j^{m_1} \neq e$ for $j = 1, \ldots, s_2$. For each b_j, $\langle b_j\rangle = \langle b_j^{m_1}\rangle$ since $(m_1, m_2) = 1$ and $\mathrm{ord}(b_j)|m_2$. Assume that $x = a^{m_1}$ and $y = a'^{m_1}$. Both x and y belong to G''. Let us 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 $b_1^{m_1}, \ldots, b_{s_2}^{m_1}$ that is the same as the group generated by $b_1, \ldots, b_{s_2}$. Therefore, G'' is of order m_2. On the other hand, G' has basis of elements $a_1, \ldots, a_{s_1}$ and is of order 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 $(\mathrm{ord}(a), m_1) = 1$, $\langle a^{m_1}\rangle = \langle a\rangle$ and $a^{m_1} \neq e$. So, we have $a^{m_1} \in G''$, which implies that $a \in \langle a\rangle = \langle a^{m_1}\rangle \subseteq G''$. Part (2) follows from part (1). □

Lemma 4. *Let $M = \{x_1, \ldots, x_k\}$ be a set of generators for an abelian group G. Assume that $|G| = n = p_1^{n_1} \cdots p_t^{n_t}$ is the prime factorization of the order of G. Let $m_i = \max\{t_i : p_i^{t_i} | \mathrm{ord}(x_j)$ for some x_j in $M\}$ and $u_i = \prod_{v \neq i} p_v^{m_v}$ for $i = 1, \ldots, t$. Let $M_i = \{x_1^{u_i}, \ldots, x_k^{u_i}\}$. Then M_i is a set of generators for $G(p_i^{n_i})$.*

Proof. For each $x_j^{u_i} \in M_i$, we have $(x_j^{u_i})^{p_i^{n_i}} = e$. Therefore, all elements of M_i are in $G(p_i^{n_i})$ (by Lemma 3). Let g be an arbitrary element in $G(p_i^{n_i})$. By Lemma 3, $g^{p_i^{n_i}} = e$. Since M is a set of generators for G, let $g = x_1^{z_1} \cdots x_k^{z_k}$. Since the greatest common divisor $(u_i, p_i^{n_i}) = 1$, there exist two integers y_1 and y_2 such that $y_1 u_i + y_2 p_i^{n_i} = 1$. We have that

$$g = g^{y_1 u_i + y_2 p_i^{n_i}} = g^{y_1 u_i} g^{y_2 p_i^{n_i}} = g^{y_1 u_i} = (x_1^{z_1} \cdots x_k^{z_k})^{y_1 u_i} = (x_1^{u_i})^{z_1 y_1} \cdots (x_k^{u_i})^{z_k y_1}.$$

We just show that g can be generated by the elements in M_i. Therefore, M_i is a set of generator for $G(p_i^{n_i})$. □

Let $X = \{x_1, \ldots, x_k\}$ be a set elements in a group G. Define a p-random product $x_1^{a_1} \cdots x_k^{a_k}$, where $a_1, \ldots, a_k$ are independent random integers in $[0, p-1]$.

Lemma 5. *Let G' be a proper subgroup of an abelian group $G = \langle x_1, \ldots, x_k\rangle$ of order p^m for some prime p. Let g be a p-random product of $\{x_1, \ldots, x_k\}$. Then $\Pr(g \in G') \leq \frac{1}{p}$.*

Proof. Since $G' \neq G$, let i be the least index such that $x_i \notin G'$. Consider $g = x_1^{a_1} \cdots x_{i-1}^{a_{i-1}} x_i^{a_i} x_{i+1}^{a_{i+1}} \cdots x_k^{a_k}$. Let $u = x_1^{a_1} \cdots x_{i-1}^{a_{i-1}}$ and $v = x_{i+1}^{a_{i+1}} \cdots x_k^{a_k}$. For any fixed u and v, there exists at most one integer $a_i \in [0, p-1]$ such that $u x_i^{a_i} v \in G'$. Assume that there exist $a_i' < a_i'' \in [0, p-1]$ such that $u x_i^{a_i'} v \in G'$ and $u x_i^{a_i''} v \in G'$. We have that $x_i^{a_i'' - a_i'} \in G'$ since G is an abelian group. Let $\mathrm{ord}(x_i) = p^s$. There exists an integer j such that $j(a_i'' - a_i') = 1 (\mathrm{mod}\ p^s)$ since $a_i'' - a_i' \in (0, p-1]$. Clearly, $x_i^{a_i'' - a_i'} \in G'$ implies $x_i = x_i^{j(a_i'' - a_i')} \in G'$. A contradiction. Therefore, with probability at most $\frac{1}{p}$, $g \in G'$. □

Lemma 6. *There exists a randomized algorithm such that given a set of generators $M = \{x_1, x_2, \ldots, x_k\}$ for a finite abelian p-group G, prime p, and integer $h \geq 1$, it computes a basis for G in $O(|M| hr \log p + (r + h) p^{r/2})$ time with probability at most p^{-h} to fail, where $|G| = p^r$ (which is not a part of input).*

Proof. We have the algorithm Randomly-Find-Basis-for-p-Group to find a basis for a p-group.

Algorithm Randomly-Find-Basis-for-p-Group

Input: prime p, a set of generators $x_1, \ldots, x_k$ of a finite abelian group G of order p^m (p^m is not a part of input), and a parameter h.

Output: a basis of G

Steps:

Let $A_0 = \{e\}$ (only contains the identity).

Let $B_0 = \{e\}$
Let $S_0 = \langle e\rangle$ (the structure for the group with one element).
$i = 0$.
Repeat
$i = i + 1$.
Generate h p-random products $a_1, \ldots, a_h$ of M.
Let $A_i = B_{i-1} \cup \{a_1, \ldots, a_h\}$.
Let B_i be a basis of $\langle A_i\rangle$ and S_i be the structure of $\langle A_i\rangle$ by the Algorithm in Theorem 1.
Until $S_i = S_{i-1}$.
Output B_{i-1} as a basis of G.
End of Algorithm

We prove that the algorithm has a small probability failing to return a basis of G. Assume that the subgroup $\langle A\rangle$ is not equal to G. By Lemma 5, for a p-random product g of M, the probability is at most $\frac{1}{p}$ that $g \in \langle A\rangle$. Therefore, for h p-random elements $a_1, \ldots, a_h$, the probability that all $a_1, \ldots, a_h$ are in $\langle A\rangle$ is at most p^{-h}. We have that the probability at most p^{-h} that the algorithm stops before returning a basis of G.

Each cycle in the loop of the algorithm is indexed by the variable i. Since G is of order p^r, the order $|\langle B_i\rangle|$ of subgroup $\langle B_i\rangle$ of G is p^{m_i} for some integer m_i. A basis of G contains at most r elements since $|G| = p^r$. Therefore, $|B_i| \leq r$. It takes $O(|M|\log p)$ time to generate one p-random product. The time spent in cycle i is $O(|M|h\log p + (|B_i| + h)\sqrt{|\langle B_i\rangle|})$. The loop is repeated at most r times since $\langle B_{i-1}\rangle \neq \langle B_i\rangle$. Assume the algorithm stops when $i = i_0$. The total time is $O(\sum_{i=1}^{i_0}(|M|h\log p + (|B_i| + h)\sqrt{|\langle B_i\rangle|})$. Since $\langle B_0\rangle \neq \langle B_1\rangle \neq \cdots \neq \langle B_{i_0}\rangle$, we have that $0 = m_0 < m_1 < \cdots < m_{i_0} \leq r$. We have $\sum_{i=1}^{i_0}((|B_i| + h)\sqrt{|\langle B_i\rangle|}) \leq \sum_{i=1}^{r}((r + h)\sqrt{p^i} = (r + h)\frac{(\sqrt{p})^{r+1}-1}{\sqrt{p}-1}$. The total time is $O(|M|hr\log p + (r + h)p^{r/2})$. □

Theorem 7. *Let ϵ be a small constant greater than 0. Then there exists a randomized algorithm such that given a set of generators $M = \{x_1, x_2, \ldots, x_k\}$ for a finite abelian group G and the prime factorization for the order $\mathrm{ord}(x_i)$ of every x_i $(i = 1, \ldots, k)$, it computes a basis for G in $O((|M|(\log n)^2 + \sum_{i=1}^{t} n_i p_i^{n_i/2}))$ time and has probability at most ϵ to fail, where $n = |G|$ has prime factorization $p_1^{n_1}p_2^{n_2}\cdots p_t^{n_t}$ (which is not a part of input) with $p_1 < p_2 < \cdots < p_t$.*

Proof. Our algorithm to find a basis of G is decomposed into finding a basis of every p-group of G. The union of every basis among all p-subgroups of G is a basis of G. Let h be a constant such that $\frac{1}{(h-1)2^{h-1}} \leq \epsilon$.

Algorithm Randomly-Find-Basis-By-Generators
Input: a set of generators $x_1, \ldots, x_k$ of a finite abelian group G and the prime factorization for every $\mathrm{ord}(x_i)$ $(i = 1, \ldots, k)$.
Output: a basis of G
Steps:
Let $p_1, \ldots, p_t$ be all of the prime numbers p_j with $p_j|\mathrm{ord}(x_i)$ for some i in $\{1, 2, \ldots, k\}$.
For $i = 1$ to t let $v_i = \max\{p_i^{t_i} : p_i^{t_i}|\mathrm{ord}(x_j)$ for some x_j in $M\}$.
Let $u = v_1v_2\cdots v_t$.
For $i = 1$ to t let $u_i = \frac{u}{v_i}$.
For $i = 1$ to t let $M_i = \{x_1^{u_i}, \ldots, x_k^{u_i}\}$.
For $i = 1$ to t
Let B_i be a basis of $\langle M_i\rangle$ by the Algorithm in Lemma 6 with input p_i, M_i, and h.
Output $B_1 \cup B_2 \cup \cdots \cup B_t$ as a basis of G.
End of Algorithm

By Lemma 4, M_i is a set of generator for G_{p_i}. By Lemma 6, the probability is at most p_i^{-h} that B_i is not a basis of G_{p_i}. The probability failing to output a basis of G is at most $\sum_{i=1}^{t} p_i^{-h} < \sum_{i=p_1}^{\infty}\frac{1}{i^h} \leq \int_{p_1}^{\infty}\frac{1}{x^h}d_x \leq \frac{1}{(h-1)p_1^{h-1}} \leq \epsilon$ since h is selected with $\frac{1}{(h-1)2^{h-1}} \leq \epsilon$. By Lemma 4, $B_1 \cup B_2 \cup \cdots \cup B_t$ is a basis of G.

Since the prime factorization of the order $\mathrm{ord}(x_i)$ for $i = 1, \ldots, k$ is a part input, it takes $O(|M|t)$ time to compute one v_i. It takes $O(|M|t^2) = O(|M|(\log n)^2)$ time to compute $v_1, \ldots, v_t$. It takes $O(t)$ time to compute u and $u_1, \ldots, u_t$.

The time for computing each element in M_i is $O(\log n)$ since $u_i \leq n$ and computing the power function (x^n) takes $O(\log n)$ time. It takes $O(|M|\log n)$ time to generate one set M_i and $O(|M|t\log n) = O(|M|(\log n)^2)$ time to generate all $M_1, \ldots, M_t$. By Lemma 6, the computational time for computing each basis of $\langle M_i\rangle$ is $O(|M_i|n_ih\log p_i + (n_i + h)p_i^{n_i/2})$. The total time is $O((|M|(\log n)^2 + (\sum_{i=1}^{t} n_ip_i^{n_i/2})))$ since h is a constant, $|M_i| = |M|$, and $\sum_{i=1}^{t} n_i = O(\log n)$. □

The fastest-known fully proven deterministic algorithm for integer factorization is the Pollard–Strassen method, which is stated in Theorem 8.

Theorem 8 *([15,7]). There exists an $2^{O((\log n)^{1/3}(\log\log n)^{2/3})}$ time algorithm to factorize any integer n.*

We have Theorem 9 to compute a basis of an abelian group only given a set of generators. Some additional time is needed to compute the orders of elements among generators.

Theorem 9. *There exists a randomized algorithm such that given a set of generators $M = \{x_1, x_2, \ldots, x_k\}$ for a finite abelian group G of order n, it computes a basis for G in $O(|M|(\log n)^2 + \sum_{i=1}^t n_i p_i^{n_i/2} + \sum_{i=1}^t \sqrt{\mathrm{ord}(x_i)})$ time, where n has prime factorization $p_1^{n_1}p_2^{n_2}\cdots p_t^{n_t}$ (which is not a part of input).*

Proof. By Theorem 2, we can find $\mathrm{ord}(x_i)$ for $i = 1, \ldots, k$ in $O(\sum_{i=1}^k \sqrt{\mathrm{ord}(x_i)})$ time. Apply the algorithm of Theorem 8 to factorize an integer $j = \mathrm{ord}(x_i)$ with $2^{O((\log j)^{1/3}(\log\log j)^{2/3})} = O(\sqrt{j})$ time for $i = 1, \ldots, k$. Apply Theorem 7 to get a basis of G. The total time is $O(|M|(\log n)^2 + \sum_{i=1}^t n_i p_i^{n_i/2} + \sum_{i=1}^t \sqrt{\mathrm{ord}(x_i)})$. □

We have Theorem 10 to compute a basis of an abelian group only given a set of generators and their orders. Some additional time is needed to factorize the orders of elements among generators.

Theorem 10. *There exists a randomized algorithm such that given a set of generators $M = \{x_1, x_2, \ldots, x_k\}$ and their orders for a finite abelian group G of order n, it computes a basis for G in $O(|M|(\log n)^2 + \sum_{i=1}^t n_i p_i^{n_i/2} + |M|2^{O((\log n)^{1/3}(\log\log n)^{2/3})})$ time, where n has prime factorization $p_1^{n_1}p_2^{n_2}\cdots p_t^{n_t}$, which is not a part of input.*

Proof. By Theorem 8, we need $|M|2^{O((\log n)^{1/3}(\log\log n)^{2/3})}$ time to factorize the orders of all elements in M. Use Theorem 7 to get a basis of G. □

5. Sublinear time algorithm with entire group as input

In this section, we present a sublinear time randomized algorithm for finding a basis of a finite abelian group. The input contains a list that holds all the elements of an abelian group. We first show how to convert a random element from G to its subgroup $G(p_i^{n_i})$ in Lemma 11.

Lemma 11. *Let $n = p_1^{n_1}\cdots p_t^{n_t}$ and G be an abelian group of n elements. Assume $m_i = \frac{n}{p_i^{n_i}}$ for $i = 1, \ldots, t$. 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 $p_i^{n_i}$ 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 $b_{i,1}, b_{i,2}, \ldots, b_{i,k_i}$ form a basis of $G(p_i^{n_i})$, i.e. $G(p_i^{n_i}) = \langle b_{i,1}\rangle \circ \cdots \circ \langle b_{i,k_i}\rangle$. Assume a is 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 $\prod_{j\neq i} G(p_j^{n_j})$. For every two integers $x \neq y \in [0, p_i^{n_i} - 1]$, $m_i x \neq m_i y \pmod{p_i^{n_i}}$ (Otherwise, $m_i x = m_i y \pmod{p_i^{n_i}}$ implies $x = y$ because $(m_i, p_i) = 1$.) Thus, the list of numbers $m_i \cdot 0 \pmod{p_i^s}$, $m_i \cdot 1 \pmod{p_i^s}, \ldots, m_i(p_i^s - 1) \pmod{p_i^s}$ is a permutation of $0, 1, \ldots, p_i^s - 1$ for any integer $s \geq 1$. Thus, 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 $m_i c_{i,j} = c' (\mathrm{mod}\, \mathrm{ord}(b_{i,j}))$ 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 $G(p_i^{n_i})$. □

Lemma 12. *Let G be a group of order p^r. Then the probability is at most $\frac{2}{p^h \ln p}$ that a set of $r + 2h\log h + 9h$ random elements from G cannot generate G.*

Proof. For every subgroup G' of G, if $|G'| = p^s$, then the probability is p^{s-r} that a random element of G is in G'. We use this fact to construct a series of subgroups $G_0 = \langle e\rangle \subseteq G_1 \subseteq G_2 \subseteq \cdots \subseteq G_{r'}$ with $r' \leq r$. Each G_i is $\langle H_i\rangle$, where H_i is a set of random elements from G and we have the chain $H_0 = \{e\} \subset H_1 \subset H_2 \subset \cdots \subset H_{r'}$, which shows that H_{i+1} is extended from H_i by adding some additional random elements to H_i.

Let G_i be a subgroup generated by some elements $G_i = \langle H_i\rangle$. If $|G_i| = p^s \leq p^{r-h}$, then add one more random element to H_i to form H_{i+1}. With probability at most p^{s-r}, the new element is in G_i. Let a be the random element to be added to H_i. Therefore, $H_{i+1} = H_i \cup \{a\}$, $G_{i+1} = \langle H_{i+1}\rangle$, and the probability is at most p^{s-r} that $G_i = G_{i+1}$.

Now assume that $|G_i| > p^{r-h}$. We add new elements according to size of G_i. Let $|G_i| = p^s$. We have $r - s < h$ since $p^s = |G_i| > p^{r-h}$. We will construct at most $h - 1$ extensions (from $G_i = \langle H_i\rangle$ to $G_{i+1} = \langle H_{i+1}\rangle$). It is easy to see that $[1, h] \subseteq \cup_{k=0}^{\lfloor \log h\rfloor}(\frac{h}{2^{k+1}}, \frac{h}{2^k}]$. For $0 < r - s < h$, there exists an integer $k \in [0, \lfloor \ln h\rfloor]$ such that $r - s \in (\frac{h}{2^{k+1}}, \frac{h}{2^k}]$. If $r - s$ is in the range $(\frac{h}{2^k}, \frac{h}{2^{k+1}}]$, then in order to form H_{i+1}, we add $2 \cdot 2^{k+1}$ new random elements to H_i. Then the probability is at most $(p^{s-r})^{2^{k+2}} \leq (p^{-\frac{h}{2^{k+1}}})^{2^{k+2}} \leq \frac{1}{p^{2h}}$ that all of the $2 \cdot 2^{k+1}$ new elements are in G_i. Thus, with probability at most $\frac{1}{p^{2h}}$ that $G_i = G_{i+1}$.

Let i_0 be the least integer i with $|G_i| > p^{r-h}$. The number of random elements used in H_{i_0-1} is at most $r-h$ since one element is increased from H_{i-1} to H_i for $i < i_0$.

Let $j = \lfloor \ln h \rfloor$. The number of integers in $(\frac{h}{2^{k+1}}, \frac{h}{2^k}]$ is at most $\frac{h}{2^k} - \frac{h}{2^{k+1}} + 1 = \frac{h}{2^{k+1}} + 1$. For $i \geq i_0$, H_{i+1} is increased by $2 \cdot 2^{k+1}$ new random elements from H_i, where $|G_i| = p^s$ with $r-s \in (\frac{h}{2^{k+1}}, \frac{h}{2^k}]$. For all extensions from H_i to H_{i+1} after $i \geq i_0$, we need at most $((h-\frac{h}{2}+1)\cdot 4 + (\frac{h}{2}-\frac{h}{4}+1)\cdot 8 + \cdots + (\frac{h}{2^j}-\frac{h}{2^{j+1}}+1)\cdot 2 \cdot 2^{j+1}) = (\sum_{i=0}^{j} 2h + \sum_{i=0}^{j} 2^{i+2}) \leq 2h(\ln h+1) + 8h = 2h\ln h + 10h$ elements. The total number of random elements used is at most $(r-h) + (2h\ln h + 10h) = r + 2h\ln h + 9h$.

The probability that $G_i = G_{i+1}$ for some $i < i_0$ is at most $\sum_{i=h}^{\infty} \frac{1}{p^i}$. The probability $G_i = G_{i+1}$ for some $i \geq i_0$ is at most $\frac{(h-1)}{p^{2h}}$. The probability that $r + 2h\ln h + 9h$ random elements of G are not generators for G is at most $\sum_{i=h}^{\infty} \frac{1}{p^i} + \frac{(h-1)}{p^{2h}} \leq 2\sum_{i=h}^{\infty} \frac{1}{p^i} \leq 2\int_h^{\infty} \frac{1}{p^x} d_x \leq \frac{2}{p^h \ln p}$. $\square$

Theorem 13. *Let h be an integer parameter. There exists a randomized algorithm such that given an abelian group G of order n with $n = p_1^{n_1} \cdots p_t^{n_t}$ with $p_1 < p_2 < \cdots < p_t$, the algorithm computes a basis of G in $O(\sum_{i=1}^{t}(n_i + h\log h)\min(p_i^{n_i/2}, p_i^{n_i-1}) + \sum_{i=1}^{t}(n_i + h\log h)\log n)$ running time and has probability at most $\frac{2}{(h-1)p_1^{h-1}\ln p_1}$ to fail.*

Proof. It takes $O(\log n)$ steps to compute 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 11. Each $G(p_i^{n_i})$ needs $O(n_i + h\log h)$ random elements to find a basis by Lemma 12. Each $G(p_i^{n_i})$ needs $O((n_i + h\log h)\log n)$ time to convert the $O(n_i + h\log h)$ random elements from G to $G(p_i^{n_i})$. It takes $O(\sum_{i=1}^{t}(n_i + h\log h)\log n))$ time to convert random elements of G into the random elements in all subgroups $G(p_i^{n_i})$ for $i = 1, \ldots, t$. For $n = p_1^{n_1} \cdots p_t^{n_t}$, $\sum_{i=1}^{t} n_i \log p_i = \log n$.

If $n_i = 1$, we just select an nonidentity element to be the basis for $G(p_i^{n_i})$. If $n_i > 1$, by Theorem 1, each $G(p_i^{n_i})$ needs $O((n_i + h\log h)p_i^{n_i/2})$ time to find a basis for $G(p_i^{n_i})$. The time spend for computing a basis of $G(p_i^{n_i})$ is $O((n_i + h\log h)\min(p_i^{n_i/2}, p_i^{n_i-1}))$. The sum of time for all $G(p_i^{n_i})$s to find basis is $O(\sum_{i=1}^{t}(n_i + h\log h)\min(p_i^{n_i/2}, p_i^{n_i-1}))$. The total time for the entire algorithm is equal to the time for generating random elements for k subgroups $G(p_i^{n_i})$ and the time for computing a basis of every $G(p_i^{n_i})$ $(i = 1, \ldots, t)$. Thus, the total time can be expressed as $O(\sum_{i=1}^{t}(n_i + h\log h)\min(p_i^{n_i/2}, p_i^{n_i-1}) + \sum_{i=1}^{t}(n_i + h\log h)\log n)$.

By Lemma 12, the probability is at most $\frac{2}{p_i^h \ln p_i}$ that we cannot get a set of generators for $G(p_i^{n_i})$ by selecting $O(n_i + h\log h)$ random elements in $G(p_i^{n_i})$. The total probability to fail is $\sum_{i=1}^{t} \frac{2}{p_i^h \ln p_i} \leq \frac{2}{\ln p_1}\sum_{i=1}^{t} \frac{1}{p_i^h} \leq \frac{2}{\ln p_1}\int_{p_1}^{\infty} \frac{1}{x^h} d_x = \frac{2}{(h-1)p_1^{h-1}\ln p_1}$. $\square$

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

Theorem 14. $\frac{H(m,c)}{m} = O(\frac{1}{(\log m)^{c/2}})$ *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 those numbers in the range $I = [\frac{m}{(\log m)^{c/2}}, m]$. It is easy to see that 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 $(\log m)^{c/2} \cdot \frac{4m}{(\log m)^c} \leq \frac{4m}{(\log m)^{c/2}}$ integers $n \in I$ to have $p^t | n$ with $p^t \geq \frac{(\log m)^c}{2}$.

Let us consider the cases $p^t | n$ for $p > (\log m)^{c/2}$ and $t \geq 2$. We ignore the case $t = 1$ because $p^{1-1} = 1$, which has no impact for $F(n) \geq (\log n)^c$. 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 $\frac{2m}{((\log m)^{c/2})^2} + \frac{2m}{(1+(\log m)^{c/2})^2} + \frac{2m}{(2+(\log m)^{c/2})^2} + \cdots < \frac{2m}{((\log m)^{c/2})^2} + \frac{2m}{((\log m)^{c/2})(1+(\log m)^{c/2})} + \frac{2m}{((1+(\log m)^{c/2})(2+(\log m)^{c/2})} + \cdots \leq \frac{2m}{((\log m)^{c/2})^2} + \frac{2m}{(\log m)^{c/2}} < \frac{4m}{(\log m)^{c/2}}$. Combining the cases above, we have $\frac{H(m,c)}{m} = O(\frac{1}{(\log m)^{c/2}})$. $\square$

Theorems 13 and 14 imply the following theorem:

Theorem 15. *There exists a randomized algorithm such that if n is in $[1, m] - G(m, c)$, then a basis of an abelian group of order n whose prime factorization is also part of the input can be computed in $O((\log n)^{\frac{c}{2}+3}\log\log n)$-time, where c is an arbitrary 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.*

Theorems 13 and 14 imply the following theorem:

Theorem 16. *There exists a randomized algorithm such that if n is in $[1, m] - G(m, c)$, then a basis of an abelian group of order n whose prime factorization is also part of the input can be computed in $O((\log n)^{c+1})$-time, where $c \geq 1$ is an arbitrary 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.*

Proof. Select a constant h such that $\frac{2}{(h-1)2^{h-1}\ln 2} < 0.1$. For prime factorization $n = p_1^{n_1} \cdots p_t^{n_t}$, $\sum_{i=1}^{t} n_i = O(\log n)$. Apply Theorems 13 and 14. $\square$

6. Deterministic algorithm with entire group as input

We also develop deterministic algorithms to compute a basis of an abelian group. Our $O(n)$ time algorithm needs the results of Kavitha [10,11].

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

6.1. Proof for $O(n)$ time algorithm

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

The second part finds a basis of each group $G(p_i^{n_i})$. The algorithm has several stages and each stage finds a member of basis at a time for $G(p_i^{n_i})$. Assume that $b_1, \ldots, b_h$, which satisfy $\mathrm{ord}(b_1) \geq \mathrm{ord}(b_2) \geq \cdots \geq \mathrm{ord}(b_h)$, are the elements of a basis of the abelian group $G(p^u)$. We will find another set of a basis $a_1, \ldots, a_h$. The element a_1 is selected among all elements in $G(p^u)$ such that a_1 has the largest order $\mathrm{ord}(a_1)$. Therefore, $\mathrm{ord}(a_1) = \mathrm{ord}(b_1)$. Assume that $a_1, \ldots, a_k$ have been obtained such that $\mathrm{ord}(a_1) = \mathrm{ord}(b_1), \ldots, \mathrm{ord}(a_k) = \mathrm{ord}(b_k)$. We show that it is always possible to find another a_{k+1} such that $(\langle a_1\rangle \cdots \langle a_k\rangle) \cap \langle a_{k+1}\rangle = \{e\}$ and $\mathrm{ord}(a_{k+1}) = \mathrm{ord}(b_{k+1})$. The possibility of such an extension is shown at Lemmas 18 and 20. 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 $a_1, a_2, \ldots, a_k$ and $\mathrm{ord}(a) \leq \mathrm{ord}(a_k)$. 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.

We show a linear time algorithm by using a result of Kavitha [10]. For an integer n, it can be factorized into product of primer numbers in $O(\sqrt{n}(\log n)^2)$ time by the brute force method. Both this section and Section 6.2 spend at least linear time for computing a basis of an abelian group. Therefore, we always assume that the primer factorization of n, which is the order of input abelian group, is known in the two sections.

In this section, we give some basic lemmas that show how to extend a partial basis for an abelian group of order p^u to a full basis. The following lemma is from Chen's early work [4]. Its proof, which was written in Chinese, is translated and refined here.

Lemma 18 ([4]). *Let G be an abelian group of order p^t for prime p and integer $t \geq 1$. Assume $a_1, a_2, \ldots, a_k$ are independent elements in G and b is also an elements in G with $\mathrm{ord}(b) \leq \mathrm{ord}(a_i)$ for $i = 1, \ldots, k$. Then there exists $b' \in \langle a_1, \ldots, a_k, b\rangle$ with $\mathrm{ord}(b')|\mathrm{ord}(b)$ such that (1) $a_1, \ldots, a_k, b'$ are independent elements in G; (2) $\langle a_1, \ldots, a_k, b'\rangle = \langle a_1, \ldots, a_k, b\rangle$; 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 $b^{p^\eta} \in \langle a_1, \ldots, a_k\rangle$.*

Proof. Let $\mathrm{ord}(a_i) = p^{n_i}$ and $\mathrm{ord}(b) = p^m$, $n_i \geq m$ for $i = 1, \ldots, k$. Let $\langle a_1, \ldots, a_k\rangle \cap \langle b\rangle = \langle c\rangle$. 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 $i = 1, \ldots, k$ 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 $\max\{\frac{p^{n_i}}{p^{\xi_i}} | t_i \neq 0, \text{ and } i = 1, \ldots, k\}$. On the hand, the order of b^{hp^η} is $\frac{p^m}{p^\eta}$. Thus, we have $\max\{\frac{p^{n_i}}{p^{\xi_i}} | t_i \neq 0, \text{ and } i = 1, \ldots, k\} = \frac{p^m}{p^\eta}$. Therefore, $p^{n_i-\xi_i} \leq p^{m-\eta}$ for each $i = 1, \ldots, k$. Thus, we have $n_i - \xi_i \leq m - \eta$. Since $(h, p) = 1$, we have $\langle b^{hp^\eta}\rangle = \langle b^{p^\eta}\rangle$. Without loss of generality, we assume that $h = 1$. It is easy to see that η is the least integer such that $b^{p^\eta} \in \langle a_1, \ldots, a_k\rangle$. We have $\xi_i \geq \eta + (n_i - m) \geq \eta$ for $i = 1, \ldots, k$. Let

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

Clearly, $b' \in \prod_{i=1}^{k}\langle a_i\rangle \cdot \langle b\rangle$. 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:

$$\langle a_1, \ldots, a_k, b\rangle = \langle a_1, \ldots, a_k, b'\rangle.$$

We now want to prove that $\langle a_1, \ldots, a_k\rangle \cap \langle b'\rangle = \{e\}$.

If, on the contrary, $\langle a_1, \ldots, a_k\rangle \cap \langle b'\rangle = \langle c'\rangle$ and $c' \neq e$. We assume $c' = b'^{p^{u\eta'}}$ for some u with $(u, p) = 1$. Since $\langle b'^{p^{u\eta'}}\rangle = \langle b'^{p^{\eta'}}\rangle$, let $u = 1$. There exist integers s_i, ξ_i' $(i = 1, \ldots, k)$ 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 $0 \le \xi_i' < n$, $0 \le \eta' < \eta$. If $\eta' \ge \eta$, we have $c' = e$ by (1)–(3). This contradicts the assumption $c' \neq e$.

Since $c = b^{p^{\eta}} \neq e$, we have $b^{p^{\eta'}} \neq e$. Since $\langle a_1, \ldots, a_k\rangle \cap \langle b\rangle = \langle b^{p^{\eta}}\rangle$ and $\eta > \eta'$, we have $b^{p^{\eta'}} \notin \langle a_1, \ldots, a_k\rangle \cap \langle b\rangle$. 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 $b^{p^{\eta'}} \in \langle a_1, \ldots, a_k\rangle \cap \langle b\rangle$. This contradicts that η is the least integer such that $b^{p^{\eta}} \in \langle a_1, \ldots, a_k\rangle$ (notice that $\eta' < \eta$). Thus, $\langle a_1, \ldots, a_k\rangle \cap \langle b'\rangle = \{e\}$. $\square$

Definition 2. Assume that group G has basis $b_1, \ldots, b_t$ with $\mathrm{ord}(b_1) \ge \cdots \ge \mathrm{ord}(b_t)$.

- Assume that $a_1, \ldots, a_k$ and b are the same as those in Lemma 18. We use independent-extension$(a_1, \ldots, a_k, b)$ to represent b' derived in the Lemma 18 such that (1) $a_1, \ldots, a_k, b'$ are independent elements in G; and (2) $\langle a_1, \ldots, a_k, b'\rangle = \langle a_1, \ldots, a_k, b\rangle$.
- Let $a_1, \ldots, a_k$ be the elements of G with $\mathrm{ord}(a_1) = \mathrm{ord}(b_1), \ldots, \mathrm{ord}(a_k) = \mathrm{ord}(b_k)$ and $(\prod_{i\neq j}\langle a_i\rangle) \cap \langle a_j\rangle = \{e\}$ for every $j = 1, \ldots, k$. Then $a_1, \ldots, a_k$ is called a *partial basis* of G. If $C(a_1, \ldots, a_k) = \{a \in G | \langle a_1, \ldots, a_k\rangle \cap \langle a\rangle = \{e\}$ and $\mathrm{ord}(a) \le \mathrm{ord}(a_k)\}$, then $C(a_1, \ldots, a_k)$ is called a *complementary space* of the partial basis $a_1, \ldots, a_k$.

It is well known that the decomposition of abelian group is unique (see [9]). For the completeness purpose, we prove the following lemma.

Lemma 19. *Let G be an abelian group of order p^m for some prime p and integer m. Let $b_1, \ldots, b_t$ be a basis of G with $\mathrm{ord}(b_1) \ge \cdots \ge \mathrm{ord}(b_t)$ and $b_1', \ldots, b_{t'}'$ be another basis of G with $\mathrm{ord}(b_1') \ge \cdots \ge \mathrm{ord}(b_{t'}')$. Then $t = t'$ and $\mathrm{ord}(b_1) = \mathrm{ord}(b_1'), \ldots, \mathrm{ord}(b_t) = \mathrm{ord}(b_t')$.*

Proof. Assume that i be the least integer that $\mathrm{ord}(b_i) \neq \mathrm{ord}(b_i')$. Without loss of generality, we assume that $\mathrm{ord}(b_i) > \mathrm{ord}(b_i')$. Let $h = \mathrm{ord}(b_i')$. Consider the generators set $\{b_1^h, b_2^h, \ldots, b_t^h\}$, which generates a subgroup of G with $\prod_{j=1}^{i} p^{\mathrm{ord}(b_j)-h}$ elements. On the other hand, generator set $\{b_1'^h, b_2'^h, \ldots, b_{t'}'^h\}$, which generates a subgroup of G with $\prod_{j=1}^{i} p^{\mathrm{ord}(b_j')-h} = \prod_{j=1}^{i-1} p^{\mathrm{ord}(b_j')-h} = \prod_{j=1}^{i-1} p^{\mathrm{ord}(b_j)-h}$ elements. Both sets generate the subgroup $\{a^h : a \in G\}$. This is a contradiction. $\square$

Lemma 20. *Let $a_1, \ldots, a_k$ be partial basis of the abelian G with p^i elements for some prime p and integer $i \ge 0$. Then (1) G can be generated by $\{a_1, \ldots, a_k\} \cup C(a_1, \ldots, a_k)$; and (2) the partial basis $a_1, \ldots, a_k$ can be extended to another partial basis $a_1, \ldots, a_k, a_{k+1}$ with complementary space $C(a_1, \ldots, a_k, a_{k+1}) = \{a \in C(a_1, \ldots, a_k) | \langle a_1, \ldots, a_k, a_{k+1}\rangle \cap \langle a\rangle = \{e\}$ and $\mathrm{ord}(a) \le \mathrm{ord}(a_{k+1})\}$, and a_{k+1} is the element of $C(a_1, \ldots, a_k)$ having the largest order $\mathrm{ord}(a_{k+1})$.*

Proof. Assume group G has a basis $b_1, \ldots, b_t$ with $\mathrm{ord}(b_1) \ge \cdots \ge \mathrm{ord}(b_t)$. (1) We prove it by using induction. It is trivial in the case $k = 0$. Assume that it is true at k. We consider the case at $k + 1$. Let $a_1, \ldots, a_k, a_{k+1}$ be the elements of a partial basis of G. Let the $C(a_1, \ldots, a_k)$ be the complementary space for $a_1, \ldots, a_k$. By assumption, G can be generated by $\{a_1, \ldots, a_k\} \cup C(a_1, \ldots, a_k)$. By the definition of partial basis (see Section 2), it is easy to see that $a_{k+1} \in C(a_1, \ldots, a_k)$. Select a_{k+1}' from $C(a_1, \ldots, a_k)$ such that $\mathrm{ord}(a_{k+1}') = \max\{\mathrm{ord}(a) : a \in C(a_1, \ldots, a_k)\}$. By Lemma 18, independent-extension$(a_1, \ldots, a_k, a_{k+1}', b) \in C(a_1, \ldots, a_k, a_{k+1}')$ for each $b \in C(a_1, \ldots, a_k)$. We still have such a property that $\{a_1, \ldots, a_k, a_{k+1}'\} \cup C(a_1, \ldots, a_k, a_{k+1}')$ can generate G. Thus, $a_1, \ldots, a_k$ can be extended into a basis of G: $a_1, \ldots, a_k, a_{k+1}', \ldots, a_{t'}'$ with $\mathrm{ord}(a_1) \ge \mathrm{ord}(a_2) \ge \cdots \ge \mathrm{ord}(a_k) \ge \mathrm{ord}(a_{k+1}') \ge \cdots \ge \mathrm{ord}(a_{t'})$ by repeating the method above. Since the decomposition of G has a unique structure (see Lemma 19), we have that $t = t'$, $\mathrm{ord}(a_1) = \mathrm{ord}(b_1), \ldots, \mathrm{ord}(a_k) = \mathrm{ord}(b_k)$, $\mathrm{ord}(a_{k+1}') = \mathrm{ord}(b_{k+1}), \ldots,$ 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 $a_1, \ldots, a_k$ to $a_1, \ldots, a_k, a_{k+1}$. (2) Notice that $C(a_1, \ldots, a_k, a_{k+1}) \subseteq C(a_1, \ldots, a_k)$. It follows from the proof of (1). $\square$

Lemma 21. *Assume G is a group of order $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$. Given the table of the orders of all elements $g \in G$ with $\mathrm{ord}(g) = p_i^j$ for some p_i and $j \ge 0$, 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 3, 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 $p_1, p_1^2, \ldots, p_1^{n_1}, p_2, p_2^2, \ldots, p_2^{n_2}, \ldots, p_t, p_t^2, \ldots, p_t^{n_t}$. 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 $p_1, p_1^2, \ldots, p_1^{n_1}, p_2, p_2^2, \ldots, p_2^{n_2}, \ldots, p_t, p_t^2, \ldots, p_t^{n_t}$ 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 $p_1, p_1^2, \ldots, p_1^{n_1}, p_2, p_2^2, \ldots, p_2^{n_2}, \ldots, p_t, p_t^2, \ldots, p_t^{n_t}$.

Set up the array A of n buckets. Put all elements of order k into bucket $A[k]$. Merge the buckets $A[p_i], A[p_i^2], \ldots, A[p_i^{n_i}]$ 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 $q_1 < q_2 \cdots < q_m$. $\square$

Assume the abelian group G has p^j elements. By Lemma 24, 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 3. Assume $a_1, a_2, \ldots, a_k, a_{k+1}$ are elements of abelian group G with p^t elements for some prime p and integer $t \geq 0$.

- Define $L(a_1, \ldots, a_k) = \langle a_1^{\frac{\mathrm{ord}(a_1)}{p}}, \ldots, a_k^{\frac{\mathrm{ord}(a_k)}{p}} \rangle - \{e\}$.
- If $A = \{a_1, \ldots, a_k\}$, define $L(A) = L(a_1, \ldots, a_k)$.

Lemma 22. *Assume $a_1, a_2, \ldots, a_k, a_{k+1}$ are independent elements of G, which has p^t elements for some prime p and integer $t \geq 0$. Then (1) $L(a_1, \ldots, a_k, a_{k+1}) = L(a_1, \ldots, a_k) \cup (L(a_{k+1}) \cup (L(a_{k+1}) \circ L(a_1, \ldots, a_k)))$, and (2) $L(a_1, \ldots, a_k) \cap (L(a_{k+1}) \cup (L(a_{k+1}) \circ L(a_1, \ldots, a_k))) = \emptyset$.*

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

Lemma 23. *With $O(m)$ steps, one can compute a^p for all elements a of group G, where $|G| = m = p^i$ 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 $a^p, a^{p^2}, \ldots, a^{p^j}$, which takes $O(j^2 \log p) = O((\log p^j)^2)$ steps. Process a^k according to the order $k = 1, 2, \ldots, p^j$, compute $(a^k)^p = (a^p)^k$ in $O(p^j)$ steps and mark $a, a^2, \ldots, a^{p^j}$ "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, $\langle a^k \rangle = \langle a \rangle$). 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 $O(m)$. $\square$

Lemma 24. *With $O(m)$ steps, one can compute $a^{\frac{\mathrm{ord}(a)}{p}}$ and $\log_p \mathrm{ord}(a)$ for all elements a of group G with $|G| = m = p^i$ 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 23, 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 $i = 1, 2, \ldots, t-1$ 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 $O(m)$. $\square$

The procedure of obtaining L is shown in the following algorithm, which is also used to find a basis of the abelian group of order power of a prime in Lemma 25.

Algorithm A
Input:
an abelian group G with order 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: a 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;
$L = L \cup L'$;
until $(\sum_{a_j \in B} R(a_j) = t)$;
output the set B as a basis of G;
end
End of Algorithm A

Lemma 25. *There is an $O(m)$ time algorithm for computing a basis of an G group with $m = p^t$ elements for some prime p and integer $t \geq 0$.*

Proof. Algorithm A is described above the lemma. By Lemma 23, we can obtain the orders of all elements of G in $O(m)$ time. With another $O(m)$ time for Bucket sorting (see [5]), 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 24, 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 $b \neq e$, $\mathrm{ord}(b) \leq \min\{\mathrm{ord}(a_i)|i = 1, \ldots, k\}$, and $\langle a_1, \ldots, a_k\rangle \cap \langle b\rangle \neq \{e\}$ iff $b^{\frac{\mathrm{ord}(b)}{p}}$ is in $L(a_1, \ldots, a_k)$. When a new a_{k+1} is found, $L(a_1, a_2, \ldots, a_k)$ becomes to $L(a_1, a_2, \ldots, a_k, a_{k+1}) = L(a_1, a_2, \ldots, a_k) \cup (L(a_{k+1}) \cup L(a_{k+1}) \circ L(a_1, a_2, \ldots, a_k))$. For each new element $g_i \in L(a_{k+1}) \cup L(a_{k+1}) \circ L(a_1, a_2, \ldots, a_k) = L(a_1, a_2, \ldots, a_k, a_{k+1}) - L(a_1, a_2, \ldots, a_k)$ (see Lemma 22), 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 hold all elements of $C(a_1, \ldots, a_k, a_{k+1})$ (see Definition 2). 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 20. □

An $O(n)$ time algorithm for computing the orders of all elements in an abelian group G was recently reported by Kavitha [11]. The proof is more involved.

Theorem 26 ([11]). *Given any group G of n elements, one can compute the orders of all elements in G in $O(n)$ time.*

Theorem 27. *There is an $O(n)$ time algorithm for computing a basis of an abelian group with n elements.*

Proof. The theorem follows from Lemmas 21, 25, and Theorem 26. □

6.2. Second proof for $O(n)$ time algorithm

We give second $O(n)$ time algorithm by using a result of Kavitha [10]. It is slightly weaker than Theorem 26.

Theorem 28 ([10]). *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.*

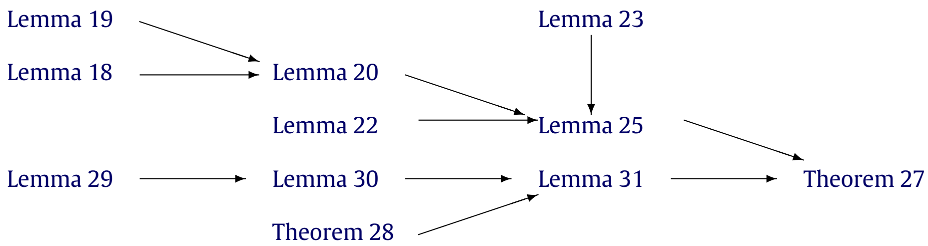

Fig. 1. Structure for proving Theorem 27.

Our second proof for Theorem 27 shows that it also follows from Lemmas 25 and 31, which is proved slightly later. Using Theorem 28 instead of Theorem 26, 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 $n = p_1^{n_1} \cdots p_t^{n_t}$. This provides a second proof of Theorem 27 without depending on Theorem 26. 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 Theorem 28 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 where 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 (Theorem 28), we obtain the $O(n)$ time decomposition for the graph G.

Our linear time decomposition method using Theorem 28 is also technically interesting as it converts an algebraic problem into a searching problem in a directed graph that every node has exactly one outgoing edge. Using Theorem 28, our method is much simpler than that in [11] and can easily converted into a linear time algorithm for the abelian group isomorphism problem. The structure of the second proof for Theorem 27 is shown in Fig. 1.

An undirected graph $G = (V, E)$ 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 $i = 1, \ldots, k - 1$. A undirected graph is connected if every pair of nodes is linked by a path. A graph $G_1 = (V_1, E_1)$ is a subgraph of $G = (V, E)$ if $E_1 \subseteq E$ and $V_1 \subseteq V$. A connected component of G is a (maximal) subgraph $G_1 = (V_1, E_1)$ of G such that G_1 is a connected subgraph and G does not have another connected subgraph $G_2 = (V_2, E_2)$ with $E_1 \subset E_2$ or $V_1 \subset V_2$.

A directed graph $G = (V, E)$ 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 $i = 1, \ldots, k - 1$. 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 $G = (V, E)$, let $G = (V, E')$ be the undirected graph that E' is derived from E by converting each directed edge of E into undirected edge. A directed graph $G = (V, E)$ is weakly connected if $G = (V, E')$ is connected. A subgraph $G_1 = (V_1, E_1)$ of $G = (V, E)$ is a weakly connected component of G if (V_1, E_1') is a connected component of (V, E').

We need the following lemma that shows the structure of a special kind directed graph in which each of its nodes has exactly one outgoing edge.

Lemma 29. *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 $G = (V, E)$ has the following properties:* (1) *Its derived undirected graph $G' = (V, E')$ 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 must 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 that 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 that every node of G has a directed path linking 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 such 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, $H = (V, (E - e)')$ is a tree. Therefore, for every node v'' in the cycle of G, every path in $(V, E - e)$ from v to v'' has to go through v'. It is still true when e is added back since e connects v'. $\square$

Lemma 30. *There exists an $O(n)$ time algorithm such that given an abelian group G of order 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 3, 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 3 to build up two subsets A and B of G, where $A = \{a \in G | a^{p^{n_1}} = e\}$ and $B = \{a^{p^{n_1}} | a \in G \text{ and } a^{p^{n_1}} \neq e\}$. Then let $G' = A$ and $G'' = B \cup \{e\}$.

During this construction, we have the table H such that $H(a) = a^p$ for every $a \in G$. We compute a^{p^j} for $j = 1, 2, \ldots, n_1$. 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 B
Input:
  group G, its order 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 (V_0, E);
  let $(E_1, V_1), (E_2, V_2), \ldots, (E_m, V_m)$ be the weakly connected components
  of (E, V_0);
  for each (V_i, E_i) with $i = 1, 2, \ldots, m$
  begin
    find the loop L_i, and put all elements of the loop into the set B;
    for each tree in $(V_i, E_i) - L_i$ 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 B

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 29 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 $v_0, v_1, \ldots, v_{n_1 - 1}$ are also in the cycle). Thus, $v \in B$. 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. Thus, 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 31. *There is an $O(n)$ time algorithm such that given a group G of order 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 order $p_i^{n_i}$ of G for $i = 1, 2, \ldots, t$.*

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 28, we can find the order of all elements in $O(n \log p) = O(n)$ time since $p = 2$ here. By Lemma 21, we can obtain the group decomposition in $O(n)$ time.
Case 2: $p_1 = 2$. Apply Lemma 30, 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 $O(n)$. □

Now we have the second proof about our linear time algorithm to compute a basis of an abelian group.

Theorem 32. *There is an $O(n)$ time algorithm for computing the basis of an abelian group with n elements.*

Proof. The theorem follows from Lemmas 31 and 25. □

6.3. Self-contained proof for an $O(n \log n)$ time algorithm

In this section, we develop an $O(n \log n)$ time algorithm to compute a basis of a finite abelian group. The algorithm and its proof are self-contained so that it can help the readers to understand our method.

Lemma 33 ([20]). *There exists an $O(n \log n)$ time algorithm such that given a group G of order n, it computes the order of all elements g with $\mathrm{ord}(g) = p_i^j$ for some $p_i||G|$ and $j \geq 0$.*

Proof. Assume that n has the primer factorization $n = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t}$ and $n_i \geq 1$ for $i = 1, 2, \ldots, t$. 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 $O(n(\sum_{i=1}^{t}(\log p_i + n_i)) = O(n \log n)$. □

Theorem 34. *There is an $O(n \log n)$ time algorithm for computing a 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 Lemmas 33 and 21, 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 25, a basis of each $G(p_i^{n_i})$ ($i = 1, 2, \ldots, t$) can be found in $O(p_i^{n_i})$ time. Thus, the total time is $O(n \log n) + O(\sum_{i=1}^{t} p^{n_i}) = O(n \log n)$. □

7. Further research and open problem

An interesting problem of further research is if there exists an $(\log n)^{O(1)}$ randomized time algorithm to find the basis of an abelian group of size $n = p^r$ for some prime p. The positive answer implies that there exists an $(\log n)^{O(1)}$ time algorithm to find a basis of an abelian group with known prime factorization for its size. Our algorithm only shows that the time is $(\log n)^{O(1)}$ for most of abelian groups.

Acknowledgements

We would like to thank Eric Allender, Igor Shparlinski, and Hong Zhu for their help and comments on an earlier version of this paper.

References

[1] L. Babai, Randomization in group algorithms: conceptual questions, in: L. Finklelstein, W.M. Kantor (Eds.), Groups and Computation. II, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 28, 1997, pp. 1–17.
[2] J. Buchmann, M.J. Jacobson Jr., E. Teske, On some computational problems in finite abelian groups, Mathematics of Computation 66 (1997) 1663–1687.
[3] J. Buchmann, A. Schmidt, Computing the structure of a finite abelian group, Mathematics of Computation 74 (2005) 2017–2026.
[4] L. Chen, Algorithms and their complexity analysis for some problems in finite group, Journal of Sandong Normal University 2 (1984) 27–33 (in Chinese).
[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second edition, The MIT Press, 2001.
[6] M. Garzon, Y. Zalcstein, On isomorphism testing of a class of 2-nilpoten groups, Journal of Computer and System Sciences 42 (1991) 237–248.
[7] K. Hardy, J.B. Muskat, K.S. Williams, A deterministic algorithm for solving $n = fu^2 + gv^2$ in coprime integers u and v, Mathematics of Computation 55 (1990) 327–343.
[8] C.M. Hoffmann, Group-Theoretic Algorithms and Graph Isomorphism, Springer-Verlag, 1982.
[9] T. Hungerford, Algebra, Springer-Verlag, 1974.
[10] T. Kavitha, Efficient algorithms for abelian group isomorphism and related problems, in: Proceedings of Foundations of Software Technology and Theoretical Computer Science, in: Lecture Notes in Computer Science, vol. 2914, 2003, pp. 277–288.
[11] T. Kavitha, Linear time algorithms for abelian group isomorphism and related problem, Journal of Computer and System Sciences 73 (2007) 986–996.
[12] J. Köbler, U. Schöning, J. Toran, The Graph Isomorphism Problem: Its Structural Complexity, Birkhaüser, 1993.
[13] G.L. Miller, On the $n^{\log n}$ isomorphism technique, in: Proceedings of the tenth annual ACM symposium on theory of computing, 1978, pp. 128–142.
[14] G.L. Miller, Graph isomorphism, general remarks, Journal of Computer and System Sciences 18 (1979) 128–142.
[15] C. Pomerance, Analysis and comparison of some integer factorization algorithms, in: H.W. Lenstra, R. Tijdeman (Eds.), Computational Methods in Number Theory, Part 1, Mathematisch Centrum, Amsterdam, Netherlands, 1982, pp. 89–139.
[16] C. Savage, An $O(n^2)$ algorithm for abelian group isomorphism, Technical report, North Carolina State University, January 1980.
[17] D. Shanks, Class number, a theory of factorization and genera, Proceedings of Symposia in Pure Mathematics 20 (1971) 414–440.
[18] C. Sims, Computation with Finitely Presented Groups, Cambridge University Press, 1994.
[19] E. Teske, A space efficient algorithm for group structure computation, Mathematics of Computation 67 (1998) 1637–1663.
[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 (1996) 1–9.