Theoretical Computer Science Stack Exchange is a question and answer site for theoretical computer scientists and researchers in related fields. Join them; it only takes a minute:

Sign up

Here's how it works:

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

Complexity of Membership-Testing for finite abelian groups

Consider the following abelian-subgroup membership-testing **problem**.

> **Inputs:**
>
> 1. A finite abelian group $G = \mathbb{Z}_{d_1} \times \mathbb{Z}_{d_1} \ldots \times \mathbb{Z}_{d_m}$ with arbitrary-large d_i.
> 2. A generating-set $\{h_1, \ldots, h_n\}$ of a subgroup $H \subset G$.
> 3. An element $b \in G$.
>
> **Output:** 'yes' if $b \in H$ and 'no' elsewhere'.

Question: Can this problem be solved *efficiently* in a classical computer? I consider an algorithm efficient if it uses $O(\text{polylog}|G|)$ time and memory resources in the usual sense of classical Turing machines. Notice that we can assume $n = O(\log|G|)$ for any subgroup H. The *input-size* of this problem is $\lceil \log|G| \rceil$.

A bit of motivation. Intuitively it looks like the problem can be tackled with algorithms to solve linear systems of congruences or linear diophantine equations (read below). However, it seems that there are different notions of computational efficiency used in the context of computations with integers, such as: strongly versus weakly polynomial time, algebraic versus bit complexity. I am not an expert on these definitions and I can not find a reference that clearly settles down this question.

> **Update:** the answer to the problem is "yes".
>
> - In a late answer, I proposed a method based on Smith normal forms which is efficient for any group with the prescribed form.
> - An answer by Blondin shows that in the particular case where all d_i are of the form $d_i = N_i^{e_i}$ and N_i, e_i are "tiny integers" then the problem belongs to $\mathrm{NC}^3 \subset \mathrm{P}$. Tiny integers are exponentially small with the input size: $O(\log\log|A|)$.
>
> In my answer I used "orthogonal subgroups" to solve this problem, but I believe this is not necessary. I will try to provide a more direct answer in the future based a row Echelon forms method I am reading.

Some possible approaches

The problem is closely related to solving linear system of congruences and/or linear diophantine equations. I briefly summarise these connection for the sake of completion.

Take A to be the matrix whose columns are the elements of the generating set $\{h_1, \ldots, h_n\}$. The following system of equations

$$Ax^T = \begin{pmatrix} h_1(1) & h_2(1) & \ldots & h_n(1) \\ h_1(2) & h_2(2) & \ldots & h_n(2) \\ \vdots & \vdots & \cdots & \vdots \\ h_1(m) & h_2(m) & \ldots & h_n(m) \end{pmatrix} \begin{pmatrix} x(1) \\ x(2) \\ \vdots \\ x(n) \end{pmatrix} = \begin{pmatrix} b(1) \\ b(2) \\ \vdots \\ b(m) \end{pmatrix} \begin{matrix} \mod d_1 \\ \mod d_2 \\ \vdots \\ \mod d_m \end{matrix}$$

has a solution if and only if $b \in H$.

If all cyclic factors have the same dimension $d = d_i$ there is an algorithm based on Smith normal forms that solves the problem in polynomial time. In this case, an efficient algorithm from [1] finds the Smith normal form of A: it returns a diagonal matrix D and two invertible matrices U and V such that $D = UAV$. This reduced the problem to solving the equivalent system system $DY = Ub \mod d$ with D diagonal. We can decide efficiently if the system has a solution using the Euclidean algorithm.

The above example suggest that the problem can be solved efficiently using similar techniques in the general case. We can try to solve the system doing modular operations, or by turning the system into a larger system of linear diophantine equations. Some possible techniques to approach the problem that I can think of are:

1. Computing the Smith normal forms of A.