Quantum Computing: An Applied Approach

Chapter 4 Problems: Complexity Theory

1

This problem statement is somewhat misleading. The discrete log problem and the factoring problem are not isomorphic; factoring an integer N is not sufficient to solve the discrete log problem modulo N. Rather, computation of log modulo N is sufficient to factor N and calculation of the discrete log problem modulo p_i, where p_i are the prime factors of N, is sufficient to factorize N.

The following proof that these two problems are equivalent is adapted from the original publication by Eric Bach (Bach 1984). I recommend reading the original proof, as it is fairly concise and there is not much added here. (https://www2.eecs.berkeley.edu/Pubs/TechRpts/1984/CSD-84-186.pdf)

1.1 First Direction: Discrete Logarithms $\rightarrow$ Factoring

Let $\phi(n)$ be the Euler totient function. Let n be an integer that is not a power of a prime, with factorization $n = p_1^{c_1}...p_k^{c_k}$, so that the corresponding group decomposition via the Chinese Remainder Theorem: $\mathbb{Z}_n^* \equiv \mathbb{Z}_{p_1^{c_1}} \times ... \times \mathbb{Z}^*_{p_k^{c_k}}$ consists of direct factors of order $\phi_i = (p_i - 1)p_i^{c_i - 1}$ dividing $\lambda = \text{lcm}(\phi_1, .., \phi_k)$.

Using a theorem proven previously (Miller 1976), the solution $a^x \cong 1(\bmod n$ for $x \neq 0$ enables selection of some $a \in \mathbb{Z}_n^*$, such that with probability greater than $\frac{1}{2}$ a can be used to find a factor p_i of n in polynomial time.

Specifically, letting $K = \{a \in \mathbb{Z}_n^* : a^{\frac{\lambda}{2}} \cong \pm 1(\bmod n)\}$ and $\nu = \max\{k : 2^k|(p_1 - 1)\} \geq \max\{k : 2^k|(p_i - 1)\}$ for all i, letting a have order ϕ_1 and order $\frac{\phi_1}{2}$ modulo the other prime powers yields an $a \notin K$ by results from number theory.

Let α be the order of this a in $\mathbb{Z}_n^*$ and $\lambda = \alpha\beta_0$, $x = \alpha\beta_1 \cdot 2^\nu$ for some odd integers β_0, β_1.

Taking $k = \nu + 1$, this yields a solution $\text{gc}(n, (a^{\frac{x}{2^k}} + 1) \bmod n)$ as a proper factor of n, so that x can be computed quickly if x itself can be accessed in polynomial time.

1.2 Second Direction: Factoring $\rightarrow$ Discrete Logarithms

Select a prime $p_i \not| \phi(n)$. Then p satisfies $(a^p)^y \cong a(\text{mod } n)$. Substituting $x = py - 1$, $a^x \cong 1$ for some prime factor p_i.

Since $\mathbb{Z}_{p^c}^*$ is cyclic, it admits an isomorphic decomposition $\mathbb{Z}_{p^c}^* \cong \mathbb{Z}_{[}^* \times \mathbb{Z}_{p^{c-1}}^+$. Assume the projection onto $\mathbb{Z}_{p^{c-1}}^+$ can be computed in polynomial time, and compute x_2 such that $x_2\theta(a) \cong \theta(b)(\text{ mod } p^{c-1})$.

This yields a coupled set of congruities, $x \cong x_1(\text{mod } p{-}1)$ and $x \cong x_2(\text{mod } p^{c-1})$. that together satisfy $a^x \cong b(\text{mod } p^c)$.

To compute θ, take the (polynomial-time) isomorphism $\lambda : U \rightarrow \mathbb{Z}_{p^{c-1}}^+$ and set $\theta = \lambda \circ \pi$, where π is the prime-counting function.

Overall, this shows that given a solution $a^{x_1} \cong b(\text{mod } p_i)$, there is a polynomial-time approach to determine an x such that $a^x \cong b(\text{mod } p_i^c)$.

Finally, taking a prime factorization $n = p_1^{c_1}...p_r^{c_r}$ of n, with solutions x_i, x with $a^{x_i} \cong b(\text{mod } p_i)$ and $a^x \cong b(\text{mod } n)$, this shows that $a^{x_i} \cong b(\text{mod } p_i^{c_i})$.

The last step is to compute the order of a modulo some prime power of n, with solution:

$$\Pi_{q|\phi(P), q \text{ prime}} q^{\nu_q(\text{order}(a))} \tag{1}$$

where $\nu_q(\text{order}(a)) = \min\{k : a^{\frac{\phi(P)}{q^k}} \cong 1(\text{ mod } P)\}$.

This shows that, given a factorization of $n = \Pi p_i^{c_i}$, a polynomial-time algorithm can be used to solve $a^x \cong b(\text{ mod } n$ for integers a and b, and vice-versa.

2

This question is also misleading; the graph isomorphism problem is in fact in the complexity class NP. For the sake of this problem, I assume that Hidary is asking to discuss the consequences and existing indications regarding whether or not it is NP-complete.

Schöning demonstrated that the graph isomorphism problem lies in the low hierarchy of NP , so that the problem ”is not NP-complete...unless the polynomial-time hierarchy collapses to some finite level” (Schöning 1987). On the other hand, many particular cases of the graph isomorphism problem can be solved in polynomial time, including that for trees, planar graphs, interval graphs, permutation graphs, circulant graphs, and bounded-parameter graphs. However, it is unclear whether these results necessarily generalize in the case of an arbitrary graph.

3

Ewin Tang demonstrated a faster classical solution to the recommendation problem, inspired by an efficient quantum solution.

Recommendation Problem: The recommendation problem asks to approximately determine an appropriate feature for a given user, where m users and n features are given as rows and columns, respectively, in a matrix $A \in \mathbb{C}^{m \times n}$.

Effectively, the problem can be posed as sampling from a row D_i, where D is an approximate version of A. It corresponds roughly to picking a feature for a given user based on a table of information, where D corresponds to a lower-dimensional version of that information.

3.1 (a)

Tang's strategy is essentially to use a dequantized version of the phase estimation algorithm in quantum computing. Tang takes into account sample and query access to input as an additional step, making it possible to compare state preparation in the quantum case with a similar step in the classical case and draw relatively similar time complexities.

Specifically, Tang frames a quantum algorithm as a problem with some set of C input states $|\phi_1\rangle, ..., |\phi_C\rangle$ and output $|\psi\rangle$ (possibly equivalent to scalar λ to some precision), and defines a dequantization operation that yields a corresponding classical algorithm. The 'dequantized' classical algorithm takes as input versions of the input state with sample and query access, SQ(ϕ_1),...,SQ(ϕ_C) and yields output in the form SQ(ψ) or λ.

To 'dequantize' the quantum solution to the recommendation problem, Tang uses a dequantized version of low-rank approximation. This mirrors the use of phase estimation in the quantum algorithm, but offers a significant speedup over the classical solution.

Low-rank approximation takes some $A \in \mathbb{C}^{m \times n}$, SQ($A$), a threshold k, and an error parameter ϵ and outputs a low-rank approximation of A with number of queries polynomial in k and ϵ^{-1}. This provides the crucical step needed to convert A into D in the statement of the recommendation problem.

Specifically, D can be expressed in terms of A as $D = AS^Y \hat{U}\hat{\Sigma}^{-1}(\hat{\Sigma}^{-1})^T \hat{U}^T S$. Sampling from this matrix $D_i = A_i S^T M S$ can be performed by approximating $A_i S^T$ using another 'dequantized' algorithm, this time inner product protocols.

Inner product protocols accept some $x, y \in \mathbb{C}^n$, SQ(x) and Q(y) and estimates $\langle x, y\rangle$ up to some error, with some probability of success.

$A_i S^T M$ can be computed with standard matrix-vector multiplication, and $A_i S^T \hat{U}\hat{U}^T S$ can be evaluated with a final 'dequantized' algorithm, the thin matrix-vector product.

The **thin matrix-vector product** accepts an input $V \in \mathbb{C}^{n \times k}$ and $w \in \mathbb{C}^k$ and approximates SQ(VW) with queries polynomial in k, taking SQ($V^\dagger$) and Q(w) as input.

Σ follows as an output of low-rank approximation, and as a diagonal matrix can be efficiently multiplied, making it possible to sample. Overall, the preceding steps make it possible to sample from an approximation of D efficiently.

3.2 (b)

Certainly, it is often difficult to prove that there exists no more efficient algorithm, even when low time complexity solutions already exist.

For example, the most efficient known algorithm for classical integer factorization is the general number field sieve (GNFS), with asymptotic running time $O(\exp\left(\left(\sqrt[3]{\frac{64}{8}}+o(1)\right)(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}\right)$ for integer n. However, Shor's Algorithm for a general quantum computer performs factorization in $O(b^3)$ time and $O(b)$ space; there may be an algorithm with greater time complexity but lower time complexity than that of GNFS.

In contrast, the Deutsch-Jozsa algorithm demonstrates a proven speedup over the classical case; the fastest deterministic classical solution has exponentially greater time complexity. There may, however, be some non-deterministic algorithm with a lower time complexity.

References

Babai, László, and Eugene M. Luks. "Canonical labeling of graphs." Proceedings of the fifteenth annual ACM symposium on Theory of computing. 1983.
Bach, Eric. "Discrete logarithms and factoring." (1984).
Miller, Gary L. "Riemann's hypothesis and tests for primality." Journal of computer and system sciences 13.3 (1976): 300-317.
Schöning, Uwe. "Graph isomorphism is in the low hierarchy." Journal of Computer and System Sciences 37.3 (1988): 312-323.