Quantum Algorithms
Homework 3 Solutions

Patrick Canny

Due: 2019-02-19

1 Book Problems

1. Exercise 3.3
   (Paraphrased) Suppose we have some NP-Oracle. Prove that there is a poly-time algorithm that finds a satisfying assignment for a given formula by making a polynomial number of queries to the oracle.

   SOLUTION

   To solve this problem, I initially thought that it would be a good idea to start with an assignment of the input variables for a predicate, making them all False or something, and then going through the predicate clause-by-clause in order to set and eliminate possibilities for each variable one at a time. After some thought, I discovered that this solution was actually an exponential time solution, as some backtracking would need to occur if an assignment was incorrect, and there was no limit on the number of times that this backtracking would need to occur.

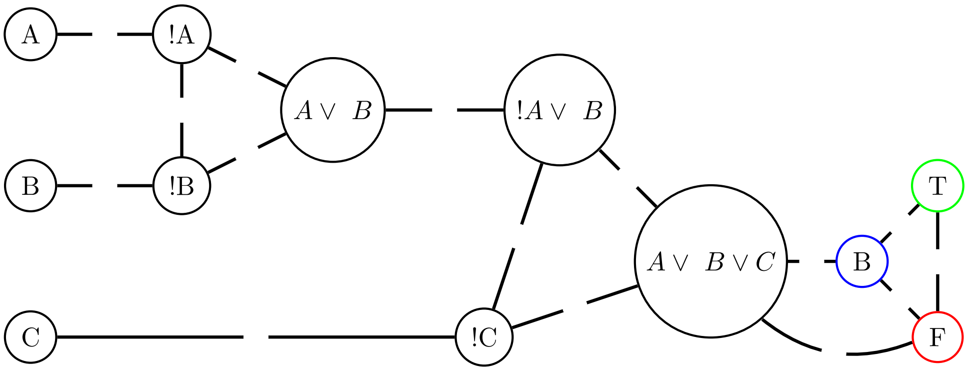
   The actual solution is the opposite procedure really: start with a given predicate and ask the Oracle if it is satisfiable. If it is, then assign one of the variables as False and ask the Oracle if the original predicate can be satisfied with this given assignment. If it can't, then flip the variable's value. If it can, then this assignment of the variable is valid in the solution. Repeat this procedure until all variables in the original predicate have a valid assignment.

   This procedure is polynomial in the number of calls to the Oracle since the Oracle will be invoked at most $1 + k$ times, where k is the number of input variables since the Oracle will be invoked once per input variable and an additional time at the beginning of the procedure.

2. Exercise 3.6(a)
   Construct a polynomial reduction of 3-SAT to 3-Color

   SOLUTION

   In order to create this polynomial reduction, we have to show in some way that a 3-SAT problem can be represented a 3-color problem. This can be done fairly easily by using the nodes of the graph as variables and the edges as their relationships to one another. We will then invoke new nodes to represent the True/False values of the results of these clauses. The following graph can be constructed as follows:



This graph will need to be constructed for each and every clause in the initial instance of 3-SAT.

Now to show that this instance of 3-SAT is satisfiable, we have to show that the graph above is 3-Colorable.

First suppose that the instance is satisfiable. For every variable v_i that is False, color that node on the graph the same color as node F, and go through a similar process for the True variables. In order for this graph to be 3-Colorable, one of the input variables must be the True color. This is because the node $A \vee B \vee C$ must be a different color from the base color (The arbitrary 3rd color of X) and the False color. If this is not possible, the graph is not 3-colorable for the clause, and thus is not satisfiable for the whole 3-SAT instance.

These graphs will need to be made for each clause in the given 3-SAT predicate, and the $A \vee B \vee C$ nodes for each graph will need to be connected to the base and false colors in the same way as the graph above.

This is enough to show that 3-SAT can be reduced to 3-Color.

2 Additional Problems

1. Compute $2^{3^{4^5}}$ mod 79. If you use a computer to do this, submit your code. There is a way to do this by hand that will almost certainly be faster than a computer, however. [*Hint:* $78 = 2 \cdot 3 \cdot 13$.]

SOLUTION

$2^{3^{4^5}}$ mod 79 can be solved using the following process:
Examine

$$3^{4^5} \bmod 79 = l + k(78)$$

which implies

$$2^l * (2^{78})^k = 2^l * (1)^k = 2^l \bmod 79$$

from here, we have to consider $l = 3^{4^5} \bmod 78$. From the problem statement, it is known that $78 = 2 \cdot 3 \cdot 13$, so the problem can be simplified slightly. We can also simplify the problem by noticing that $4^5 = (2^2)^5 = 2^10 = 1024$.

$$l = 3^{1024} \bmod 78 = 3 \bmod 78$$

finally leading to

$$2^3 \bmod 79 = 8 \bmod 79$$

2. Suppose that the Turing machine $\mathcal{M}$ computes the predicate $L(x)$ probabilistically:

$$\begin{aligned} L(x) = 1 \quad &\Rightarrow \quad \mathcal{M} \text{ outputs “yes” with probability } \geq 1 - \varepsilon; \\ L(x) = 0 \quad &\Rightarrow \quad \mathcal{M} \text{ outputs “no” with probability } \geq 1 - \varepsilon. \end{aligned}$$

We wish to decrease the probability of making an error by running the machine $\mathcal{M}$ several times (say k times) and selecting the “yes”/“no” answer that occurs most frequently.

(i) Let P_E be the probability of producing a wrong answer using the above procedure. Assuming that $\varepsilon < 1/2$, prove that

$$P_E \leq \left(2\sqrt{(1-\varepsilon)\varepsilon}\right)^k \qquad \text{and} \qquad 2\sqrt{(1-\varepsilon)\varepsilon} < 1.$$

Be sure to fully justify your proof.

SOLUTION

To begin thinking of this problem, we must first enumerate the runs of the machine. Say that $S \in \{1...k\}$ are the runs outputting “yes”. By the definition of the machine, $|S| \leq k/2$. so the probability of all “yes” happening in one go is given by

$$(1-\varepsilon)^{|S|}) * \varepsilon^{k-|S|}$$

thus, the overall probability of error P_E

$$P_E \leq \sum_{S \in \{1...k\}}^{\varepsilon^{k-|S|}} (1-\varepsilon)^{|S|}) * \varepsilon^{k-|S|} \qquad (|S| \leq k/2)$$

take $|S|$ and call it l. We can then rewrite the above sum in the following way:

$$P_E \leq \sum_{l=0}^{k/2} (1-\varepsilon)^{l}) * \varepsilon^{k-l}$$

allowing us to reason about the sum in a more simple way. From here, it is possible to simplify the sum until we can reduce it to a closed form:

$$P_E \leq (1-\varepsilon)^{k/2} * \varepsilon^{k/2} * \sum_{l=0}^{k/2} \varepsilon^{(l-k/2)} * \varepsilon^{k/2} * \binom{k}{l} = \left(\sqrt{(1-\varepsilon)\varepsilon}\right)^k \sum_{l=0}^{k/2} (\frac{\varepsilon}{1-\varepsilon})^{k/2} - l * \binom{k}{l}$$

The inner portion of the sum, $(\frac{\varepsilon}{1-\varepsilon})^{k/2} - l$ will always be less than 1, and since we are concerned with showing that $P_E \leq 1$, we can just coerce this quantity to 1, simplifying the following calculations:

$$\left(\sqrt{(1-\varepsilon)\varepsilon}\right)^k \sum_{l=0}^{k/2} \binom{k}{l} \leq \left(\sqrt{(1-\varepsilon)\varepsilon}\right)^k * 2^k = \left(2\sqrt{(1-\varepsilon)\varepsilon}\right)^k$$

This is enough to show $P_E \leq \left(2\sqrt{(1-\varepsilon)\varepsilon}\right)^k$, but now we have to show $2\sqrt{(1-\varepsilon)\varepsilon} < 1$. This ensures that the probability attained through this calculation does not exceed 1. Call this quantity λ.

From this we derive

$$\lambda^2 = 4(1-\varepsilon)\varepsilon = 4(\varepsilon - \varepsilon^2) = 4(-(\varepsilon - 1/2)^2 + 1/4) = -4(\varepsilon - 1/2)^2 + 1 < 1$$

the above is true since ε will always be less than 1.

(ii) Does taking $\varepsilon < 1/2$ in the above proof actually matter?

SOLUTION

No, since it is possible to increase k to a point where we achieve whatever probability is required, so epsilon can really be anything smaller than 1.

(iii) If $\varepsilon = 0.49$, how many times do we have to run $\mathcal{M}$ for our answer to be accurate to within "5σ" (i.e. the probability of a correct answer is 0.9999994)?

SOLUTION

Set up equation $(1 - P_E) = \left(2\sqrt{(1-\varepsilon)\varepsilon}\right)^k$ where $\varepsilon = 0.49$ and $P_E = 0.9999994$ and solve for k

The above equation can be simplified to

$$k = \frac{\log(1 - P_E)}{\log 2\sqrt{(1-\varepsilon)\varepsilon}}$$

And solving this equation with the given values for P_E and ε gives

$$k = 71617.4$$

so the machine must be run 71618 times

(iv) If $\varepsilon = 0.25$, how many times do we have to run $\mathcal{M}$ for our answer to be accurate to within "5σ"?

SOLUTION

Repeat the procedure above with a different value for ε, yielding

$$k = 99.5984$$

so the machine must be run 100 times.

3. Let $n \in \mathbb{N}$ and define $\varphi(n) = \left|(\mathbb{Z}/n\mathbb{Z})^\times\right|$ (i.e. the number of numbers coprime to n between 1 and n).

(i) Prove that if $\gcd(m, n) = 1$ then $\varphi(m \cdot n) = \varphi(m)\varphi(n)$.

SOLUTION

Claim 2.1. *The above is true.*

Proof of claim. Take A, B, C to be the sets of coprimes from 1 to m, n, and mn respectively. These sets are all **rings** since they are closed under multiplication, addition and have associative and distributive properties. They have a "zero-element" (m, n or mn) as well. These rings must also be finite, since $m, n \in \mathbb{N}$.

The multiplicative property of the Euler Totient Function can be proven using some facts about rings, and the fact that A, B, and C are all rings.

In general, if A and B are rings then $A \times B$ is also a ring with elements (a, b) where $a \in A$ and $b \in B$. Since the rings are finite, the number of units in $A \times B$ is the number of units in A times the number of units in B.

Now, we must find the **units** in each of our rings. A unit is a number within one of our rings is a number that is coprime to n from $\mathbb{Z}/n\mathbb{Z}^\times$, so finding the number of units in one of these rings is equivalent to finding $\varphi(a)$ where a can be n, m, or mn.

Now take $\gcd(m, n) = 1$. This implies the following, which is eerily close to the Chinese Remainder Theorem:

$$\mathbb{Z}\langle mn\rangle \cong \mathbb{Z}\langle m\rangle \times \mathbb{Z}\langle m\rangle$$

From the logic above, the number of units in $\mathbb{Z}\langle mn\rangle = \varphi(mn)$ and the number of units in $\mathbb{Z}\langle m\rangle \times \mathbb{Z}\langle n\rangle = \varphi(m)\varphi(n)$. Because of the above fact, this implies $\varphi(mn) = \varphi(m)\varphi(n)$. •

(ii) Prove that if p is a prime then

$$\varphi(p^k) = p^{k-1}(p-1) = p^k\left(1 - \frac{1}{p}\right).$$

SOLUTION

Claim 2.2. *Part 2 holds*

Proof of claim. Since p is prime, then the values for

$$\gcd(p^k, m) = \{1, p^2, p^3, ..., p^k\}$$

For $\gcd(p, m)$ to equal anything other than 1, m must be a direct multiple of p. The multiples of p smaller than k can be given by

$$\{p, 2p, 3p, ..., p^{k-1} * p = p^k\}$$

and the size of this set is given by p^{k-1}.

The above implies that the other $p^{k-1}(p-1)$ numbers are all relatively prime to p. •

(iii) Use the previous parts to prove that

$$\varphi(n) = n \prod_{\substack{p \text{ prime,} \\ p|n}} \left(1 - \frac{1}{p}\right)$$

(the product is over all prime divisors of n).

SOLUTION

Claim 2.3. *Part 3 holds true.*

Proof of claim. We know that a number can be given by a unique prime factorization, so proving this product is showing this to be true.

Explicitly, we can look at

$$n = {p_1}^{k_1} * ... * {p_z}^{k_z}$$

and apply the facts proven in parts 1 and 2. The above formula can then be expressed as:

$$\begin{aligned}\varphi(n) &= \varphi({p_1}^{k_1}) * \varphi({p_2}^{k_2}) * ... * \varphi({p_z}^{k_z}) \\ &= {p_1}^{k_1}\left(1 - \frac{1}{p_1}\right) * ... * {p_z}^{k_z}\left(1 - \frac{1}{p_z}\right) \\ &= {p_1}^{k_1}{p_2}^{k_2}...{p_z}^{k_z} * \left(1 - \frac{1}{p_1}\right) * ... * \left(1 - \frac{1}{p_z}\right) \\ &= n \prod_{\substack{p \text{ prime,} \\ p|n}} \left(1 - \frac{1}{p}\right)\end{aligned}$$

Proving the product formula. •

4. Using the φ function from the previous problem, prove that if x and n are coprime, then

$$x^{\varphi(n)} \equiv 1 \text{ mod } n.$$

Explain why this is a generalization of Fermat's Little Theorem.

SOLUTION

Claim 2.4. *The theorem above is true.*

Proof of claim. Take the set of numbers less than n and coprime to n. It looks like $\{a_1, a_2, ..., a_{\varphi(n)}\}$. Now, consider a number $c < n$ and coprime to n from the set above.

Observe that a_i, $ca_i \equiv a_j \pmod n$ for some j. If $ca_i \equiv ca_j \pmod n$ then $a_i = a_j$ since a_i and a_j must be equivalent mod n.We can use this fact to build a modified version of the original set, looking like: $\{ca_1, ca_2, ..., ca_{\varphi(n)}\}$

Thereby, we have

$$\prod_{k=1}^{\varphi(n)} ca_k \equiv \prod_{k=1}^{\varphi(n)} a_k \pmod n.$$

From the above, we see

$$c^{\varphi(n)} \prod_{k=1}^{\varphi(n)} a_k \equiv \prod_{k=1}^{\varphi(n)} a_k \pmod n$$

allowing us to cancel the product on each side since the product is coprime to n. This yields:

$$c^{\varphi(n)} \equiv 1 \pmod n$$

whenever $(c, n) = 1$. $\bullet$

This is a generalization of Fermat's theorem since it does not require the exponent to be $p - 1$ and could be used for any number coprime to n.

5. Determine the last 2 digits in the decimal expansion of $2^{3^{4^5}}$ (i.e. the digits in the 1s place and the 10s place).

SOLUTION

For this problem, we need to find the solution to $2^{3^{4^5}}$ mod 100

Taking a number modulo 100 will give us it's final two digits, so from here we can start to reason about the number itself.

$$\varphi(100) = \varphi(2^2) * \varphi(5^2) = (2^2 - 3)(5^2 - 5) = 40 \qquad 4^5 = 0 \text{ mod } 8 \qquad 4^5 = -1 \text{ mod } 5$$

by CRT, we can get

$$4^5 = 24 \text{ mod } 40 \qquad 3^{4^5} = 3^{24} \text{ mod } 40$$

At this point, I was really unsure of what to do, so I looked at the answer using wolfram. Seems like I will need to practice this kind of problem in the future. It turns out that the final 2 digits are 52.