CSCI 5170: Computational Complexity **Lecture 6**
The Chinese University of Hong Kong, Fall 2019

Quantum computation is an extension of randomized computation that captures the ability of quantum mechanical devices to occupy a "superposition" of states. Quantum computers are still nonexistent but there is reasonable belief that it may be possible to build them in the future.[1] In contrast a rigorous mathematical model of quantum computing has been around since the 1980s.

Besides various applications like simulating quantum systems in physics and chemistry, quantum computers are interesting because if the model is accurate then they can be used to solve certain problems substantially faster than classical computers.

1 Reversible computation

One property of quantum mechanical systems is that they must be *reversible*: No loss of information can occur during the evolution of the system. Classical computation is certainly not reversible. I can format my memory and forget what was written there before.

Deterministic classical computation, however, can easily be made reversible. One interesting example of reversible computation that we already saw is the simulation of circuits by branching programs. To implement this simulation we designed a 3-register machine that starts in the memory state (A, B, C) and ends in $(A, B, C + f \cdot B)$, where f is the function to be computed. If we computed f again starting in this state we would recover the original state (A, B, C).

If we are not too worried about memory usage, any deterministic sequential computation can be made reversible. We can do this by replacing all the gates (instructions) by "reversible gates". It is enough to illustrate this for NOT and AND gates. NOT is already reversible, while AND can be simulated by the reversible gate:[2]

$$T|x, y, z\rangle = |x, y, z \text{ XOR } (x \text{ AND } y)\rangle.$$

When the input z is initialized to zero then the third component computes exactly x AND y, but the gate is now reversible: Applying T to its own output recovers the original state.

Any sequential computation implemented with these gates can be "undone" by applying the same gates in reverse order. To be concrete think of the formula

$$f = (\text{NOT } (x_1 \text{ AND } x_2)) \text{ AND } x_3.$$

To implement this formula reversibly, we introduce extra registers y_1 and y_2 meant to store the intermediate and final value of the formula evaluation. We then run the sequence of instructions $T|x_1, x_2, y_1\rangle$, NOT $|y_1\rangle$, and $T|y_1, x_3, y_2\rangle$ in order. These have the following effect on the state:

$$\begin{aligned} |x_1, x_2, x_3, y_1, y_2\rangle &\xrightarrow{T|x_1,x_2,y_1\rangle} |x_1, x_2, x_3, y_1 \text{ AND } (x_1 \text{ XOR } x_2), y_2\rangle \\ &\xrightarrow{\text{NOT } |y_1\rangle} |x_1, x_2, x_3, y_1 \text{ AND NOT } (x_1 \text{ AND } x_2), y_2\rangle \\ &\xrightarrow{T|y_1,x_3,y_2\rangle} |x_1, x_2, x_3, y_1 \text{ XOR NOT } (x_1 \text{ AND } x_2), y_1 \text{ XOR } y_2 \text{ XOR } f\rangle. \end{aligned}$$

If y_1 and y_2 were initialized to zero, the value of f can be read off from the third registers. The computation is reversible because no matter what the five registers were initially set to their values can be recovered from the final state (by reapplying the same instructions in reverse order).

[1] This is being debated as we speak: See here and here for the latest quantum computing gossip.

[2] For now $|x_1, \ldots, x_n\rangle$ is just fancy notation for the sequence/string $x_1 \ldots x_n$.

From an algebraic perspective, a reversible computation is simply a *permutation* of the states. This permutation is obtained by composing *local* permutations corresponding to the gates that only act on 2 or 3 bits of the state at a time. In the example we just did, there are 32 possible states described by the 5-bit state string $|x_1, x_2, x_3, y_1, y_2\rangle$. The computation has the effect of permuting all these strings: It swaps $|00000\rangle$ and $|00011\rangle$, it swaps $|00001\rangle$ and $|00010\rangle$, and so on.[3]

In the analysis of sublinear-time computations, we were interested in query complexity, that is in counting how many times the algorithm queries a bit of its input x. In the non-reversible setting, we can think of input queries as being implemented by a "query gate" that asks for the i-th bit of the queried string x. This download gate can also be implemented reversibly as $Q^x(i, z) = (i, z \text{ XOR } x_i)$. If z is initialized to zero, it contains the value x_i after evaluation.

In conclusion, a reversible computation can be described by an initial state $|s^0\rangle = |s_1, \dots, s_n\rangle$ and a sequence of local permutations $L_1, \dots, L_t$ applied to this state, resulting in the final state

$$|s^t\rangle = L_t \cdots L_1 |s^0\rangle. \tag{1}$$

The notation $L_i|s\rangle$ stands for "permutation L_i applied to string s". If the computation makes (at most) q queries, then (at most) q of the permutations $L_1, \dots, L_t$ implement query gates. The other ones are completely independent on the input x. If we lump those permutations together in chunks of (possibly non-local) permutations, we obtain that a reversible q-query algorithm implements a state transformation of the form

$$|s^t\rangle = P_q Q_q^x P_{q-1} Q_{q-1}^x \cdots P_1 Q_1^x P_0 |s^0\rangle, \tag{2}$$

where $P_0, \dots, P_q$ are permutations that do not depend on x and $Q_1^x, \dots, Q_q^x$ are (reversible) queries to x.

Randomized register machines As soon as we try to enhance reversible computation with the ability to produce random bits we run into trouble. The (classical) process of producing randomness is not reversible: A "random register" takes value $|0\rangle$ zero with probability half and $|1\rangle$ with probability half regardless of what was in there before so the initial information is lost.

We can still model randomized computation as a register state machine, but some of the gates no longer represent permutations of the registers. In particular the operation $R(s_1)$ meaning "generate a random number in the first register" sends both states of the form $|0s_2 \dots s_n\rangle$ and $|1s_2 \dots s_n\rangle$ into the state $|0s_2 \dots s_n\rangle$ with probability half and $|1s_2 \dots s_n\rangle$ with probability half.

Once the start state and input are fixed, the state of an n-register machine at any point in time is a probability distribution p over all 2^n possible register values $x \in \{0,1\}^n$. Every instruction induces a transformation of this distribution p into a new distribution p': For example the instruction $R(s_1)$ randomizes the value of register 1 while leaving all others intact, in which case

$$p'(0s_2 \dots s_n) = p'(1s_2 \dots s_n) = \frac{1}{2}p(0s_2 \dots s_n) + \frac{1}{2}p(1s_2 \dots s_n) \quad \text{for all } y \in \{0,1\}^{n-1}.$$

On the other hand, an instruction like $T(s_1, s_2, s_3)$ preserves the values of the probabilities but permutes the states they correspond to, namely

$$p'(s_1 s_2 s_3' s_4 \dots s_n) = p(s_1 \dots s_n), \quad \text{where } s_3' = s_3 \text{ XOR } (s_1 \text{ AND } s_2).$$

The transformations from p to p' are *linear* and *stochastic*: Each $p'(x')$ is a nonnegative linear combination of $p(x)$s, and the contributions of each $p(x)$ add up to one. The composition of stochastic linear transformations is also a stochastic linear transformation, so the final state p is also a probability distribution. Thus a randomized computation can again be described by

[3] In fact it is an involution: all permutation cycles have length 1 or 2.

2 Quantum Computation

In a quantum system with N states, each of the N possible base states s corresponds to a vector $|s\rangle$ in N-dimensional Hilbert space. These vectors form an orthonormal basis: They are orthogonal vectors of unit length. The state of the quantum system can be any unit vector in 2^n dimensions, namely any vector of the form

$$|v\rangle = \sum a(s) \cdot |s\rangle, \quad \text{where} \sum_s a(s)^2 = 1.$$

The *amplitudes* $a(s)$ may be positive, zero, negative, or even complex numbers. The state of the quantum system can only be manipulated via unitary transformations, namely linear transformations that preserve the length of vectors.

Quantum systems can occupy a *superposition of states*: For example, a 1-qubit quantum system can be in either of the two "basis states" $|0\rangle$ or $|1\rangle$, but also in any length-preserving superposition of them like $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and also $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. An "$n$-qubit" quantum computer has 2^n possible base states $|s\rangle : s \in \{0,1\}^n$ and can in principle occupy various superpositions of them.

The distinction between being a superposition and a probability distribution is important: Unitary transformations are invertible, but stochastic transformations are not (unless they are permutations). For this reason, any quantum process, including a computation, can always be reversed: The input can be "uncomputed" from the output. Since permutations are in particular orthogonal (a permuted orthogonal basis remains orthogonal), quantum computations generalize deterministic computations. Randomized computations, on the other hand, are not reversible, so it is less clear that they can be simulated quantumly.

It turns out that this can be done. The invariant maintained by the simulation is that if the classical machine is in state s with probability $p(s)$ then the quantum machine will be in quantum state $\sum \sqrt{p(s)} \cdot |s\rangle$ with $a(s)$. Since quantum transformation preserve length, i.e. sums of squares of amplitudes, they are probability preserving. The instruction "generate a random bit" can be implemented by the *Hadamard gate* H:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

If we represent $|0\rangle$ and $|1\rangle$ as basis vectors in the plane, this has the effect of rotating the state by -45 degrees and then flipping it about the $|0\rangle$ axis. Repeating the transformation recovers the original state: H^2 is the identity. The squared amplitudes of the state $H|0\rangle$ are precisely the probabilities of a random bit.

In this simulation all of the amplitudes of the quantum machines are positive. One property of quantum computation that makes it potentially more powerful than randomized classical computation are the negative amplitudes: While probabilities can only add up, amplitudes can also cancel out.

At the end of the computation, the machine is in some superposition state $\sum a(s) \cdot |s\rangle$. To observe the register a *measurement* is performed: The resulting value is s with probability $|a(s)|^2$. Thus the outcome of a quantum computation is always probabilistic.

To summarize, quantum computations have the same forms (1) and (2) as reversible classical computations, but the relevant "state change" operators L_i, P_i are now unitary of dimension $N \times N$ for an N-state machine. The final state is a superposition of the N states $\sum a(s) \cdot |s\rangle$. A measurement is then performed which outputs s with probability $|a(s)|^2$.

3 The Power of Quantum Queries

To illustrate a quantum ability that is impossible to perform classically, we give an algorithm for computing the XOR of two bits x_0 XOR x_1 with a single quantum query to x:

Set the initial state $|s^0\rangle$ to $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$.
Apply a query gate Q^x.
Apply a Hadamard gate H to the first register.
Measure and output the first register's value.

Recall that the query gate looks at the index i to be queried from the first register and XORs the value x_i to the contents of the second register. If $x_0 = x_1$ then $Q|s^0\rangle$ is either $|s^0\rangle$ or $-|s^0\rangle$. In particular, for any value of the second register, the first register is of the form $\pm(|0\rangle + |1\rangle)$. After applying the Hadamard gate, the first register becomes $|0\rangle$, so measurement will always produce 0.

If $x_0 \neq x_1$ then $Q|s^0\rangle$ is one of the states

$$\tfrac{1}{2}(-|00\rangle + |01\rangle + |10\rangle - |11\rangle) \quad \text{or} \quad \tfrac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$$

and for any value of the second register, the first one is of the form $\pm(|0\rangle - |1\rangle)$. After applying a Hadamard gate this register maps to $|1\rangle$ and measurement will always produce 1.

There is a more geometric way to interpret this algorithm. To do this, we modify the query gate so that instead of writing down the answer in a separate register, it encodes it in the phase of the query register:

$$\Phi^x|i\rangle = (-1)^{x_i}|i\rangle.$$

This *phased query* gate Φ can be implemented by the following sequence: Apply H to the answer register, apply query Q, then apply another H to the answer register. These operations preserve $|i0\rangle$ regardless of the value of x_i. On the other hand, $|i1\rangle$ is preserved when $x_i = 0$ but flipped when $x_i = 1$.

Using phased query gates the algorithm has a simpler implementation: Start with the state $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and apply a phase query Φ^x. If $x_0 = x_1$ then the state after the query is $\pm H|0\rangle$; if not it is $\pm H|1\rangle$. The two types of states are orthogonal to one another so they can be distinguished perfectly after a suitable orthogonal transformation (in this case, another Hadamard gate) and measuring the outcome.

4 Grover's Algorithm

We now consider the search problem of finding a marked item in the database, provided that exactly one such item exists.[4] Given a string $x \in \{0,1\}^n$, the task is to find a marked index i such that $x_i = 1$ under the promise that exactly one such index exists. Solving this problem in particular gives the ability to compute the $APXOR$ function, whose randomized query complexity is $\Omega(n)$. Grover's algorithm solves it using $O(\sqrt{n})$ quantum queries.

Grover's algorithm is based on the following transformation which acts on the space spanned by the basis vectors $|1\rangle, \dots, |n\rangle$:

$$S|i\rangle = \frac{2}{\sqrt{n}}|+\rangle - |i\rangle,$$

[4] The algorithm works even if there are multiple marked items, provided we know the number of marked items say within a factor of two.

where $|+\rangle = (|1\rangle + \cdots + |n\rangle)/\sqrt{n}$. This is a unitary transformation. One way to check this is to show that $S|1\rangle, \dots, S|n\rangle$ is an orthogonal basis. The "braket notation" is very convenient for such calculations:

$$\langle i|S^T S|j\rangle = \left(\frac{2}{\sqrt{n}}\langle +| - \langle i|\right)\left(\frac{2}{\sqrt{n}}|+\rangle - |j\rangle\right) = \frac{4}{n}\langle +|+\rangle - \frac{2}{\sqrt{n}}\langle i|+\rangle - \frac{2}{\sqrt{n}}\langle +|j\rangle + \langle i|j\rangle.$$

The sum of the first three terms vanishes because $|+\rangle$ is a unit vector which has value $1/\sqrt{n}$ in the i-th and j-th coordinate. Thus $S|i\rangle$ and $S|j\rangle$ have the same inner product as $|i\rangle$ and $|j\rangle$ so the transformation preserves orthogonality. Applied to a superposition state, S averages out the amplitudes in the $|+\rangle$ direction then subtracts the original state:

$$S\sum a(i)|i\rangle = 2\overline{a}\sqrt{n}\cdot|+\rangle - \sum a(i)|i\rangle = \sum(2\overline{a} - a(i))|i\rangle \quad \text{where} \quad \overline{a} = \frac{a(1) + \cdots + a(n)}{n}.$$

This transformation preserves the average value of the amplitudes, but it changes their individual values. To get a sense of what happens, suppose that initially all amplitudes $a(i)$ are $1/\sqrt{n}$, except a special marked one that has value $a(i^*) = -1/\sqrt{n}$. The average of these amplitudes is $(1-2/n)/\sqrt{n} \approx 1/\sqrt{n}$. After applying S the new amplitudes become $(1-4/n)/\sqrt{n} \approx 1/\sqrt{n}$, except the marked one which grows to $(3 - 2/n)/\sqrt{n} \approx 3/\sqrt{n}$. The amplitude of the marked item has grown by about $2/\sqrt{n}$. Thus S has the effect of "spreading out" the amplitudes apart.

What happens if we repeat? The transformation S is its own inverse, so repeating S annulls all the gains. However, if we flip back the phase of the marked item's amplitude to $-(3 - 2/n)/\sqrt{n}$, then applying S again increases the magnitude back to about $2/\sqrt{n} - 3/\sqrt{n} = 5/\sqrt{n}$. Each time we apply a phase flip followed by S the amplitude grows by about $2/\sqrt{n}$. After $O(\sqrt{n})$ steps we would expect it to reach a value close to 1.

```
Grover's algorithm:
Set the initial state to $|+\rangle = (|1\rangle + \cdots + |n\rangle)/\sqrt{n}$.
Repeat the following $t$ times:
    Apply a phased query $\Phi^X$.
    Apply $S$.
Measure and output the measured string.
```

The number of steps t is a number on the order of $\sqrt{n}$. It will be described precisely in the analysis.

To understand this algorithm we need to figure out the effect of applying $S\Phi^X$ repeatedly. One way to do this is to calculate the eigenvalues and eigenvectors of this transformation. There is also a more intuitive geometric interpretation. For a string X all but one of whose entries are zero, the transformation Φ^X is a reflection about the plane perpendicular to $|i^*\rangle$. The transformation S on the other hand is a reflection about the $|+\rangle$ vector.

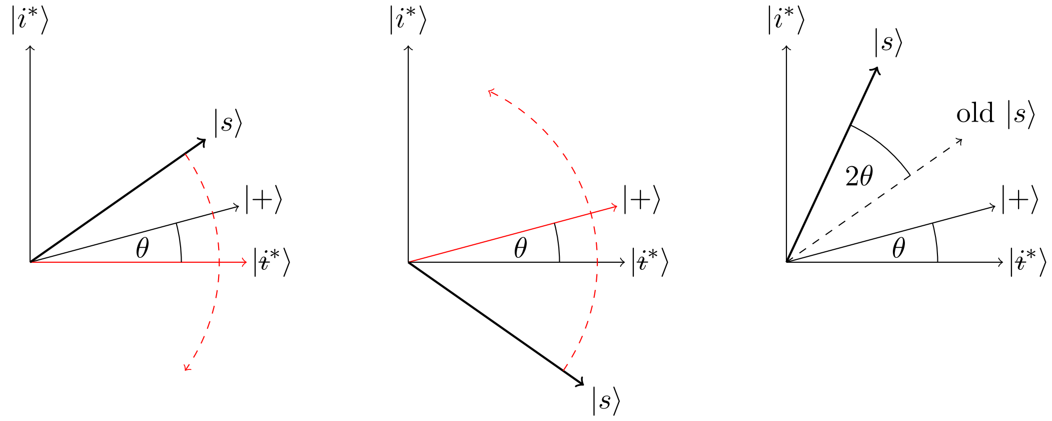
Thus starting with the $|+\rangle$ vector, the state is reflected about the plane perpendicular to $|i^*\rangle$, then the plane perpendicular to $|i^*\rangle$, and so on. This vector will always remain in the plane spanned by $|i^*\rangle$ and $|+\rangle$. Therefore a reflection in the plane perpendicular to $|i^*\rangle$ amounts to a reflection about the vector

$$|\not{i}^*\rangle = \sqrt{\frac{n}{n-1}}\left(|+\rangle - \frac{1}{\sqrt{n}}|i\rangle\right) = \frac{1}{\sqrt{n-1}}\sum_{i \neq i^*}|i\rangle.$$

In this notation, Grover's algorithm has a simple geometric description:

```
Set the initial state $|s\rangle$ to $|+\rangle = (|1\rangle + \cdots + |n\rangle)/\sqrt{n}$.
Repeat the following $t$ times:
    Reflect $|s\rangle$ about $|\not{i}^*\rangle$ and then about $|+\rangle$.
Measure and output the measured string.
```

If θ be the angle between $|+\rangle$ and $|\not{i}^*\rangle$, the transformation 3 has the effect of decreasing the angle between $|s\rangle$ and $|i^*\rangle$ by 2θ as shown in the following figure:



Initially, θ is the angle between $|+\rangle$ and $|\not{i}^*\rangle$, so $\sin\theta = \langle +|\not{i}^*\rangle = 1/\sqrt{n-1}$. Using the Taylor approximation $\sin\theta = \theta - O(\theta^3)$ we get that $\theta = 1/\sqrt{n-1} - O(1/n^{3/2})$. After t steps the angle between $|s\rangle$ and $|\not{i}^*\rangle$ is $(2t+1)\theta$. If we choose $2t+1$ to be the integer closest to $\pi/2\theta$, which is on the order of $\sqrt{n}$, the angle between $|s\rangle$ and $|i\rangle$ becomes at most θ. The probability that the measurement results in i^* is then at least $\cos\theta \geq 1 - \theta^2/2 = 1 - O(1/n)$.

5 Quantum Query Complexity and Approximate Degree

Can Grover's algorithm be imrpoved? How do we even bound the query complexity of a quantum algorithm? Recall that the randomized query complexity of an algorithm was lower bounded by its approximate degree. It turns out that this is almost true for quantum algorithms also. The *quantum query complexity* $Q_\varepsilon(f)$ is the smallest possible number of queries made by a quantum algorithm computing f with probability at least $1-\varepsilon$ on all inputs.

Lemma 1. *$\widetilde{\deg}_\varepsilon(f) \leq 2Q_\varepsilon(f)$.*

Proof. We show that the amplitudes of the state $|s^t\rangle$ in (2) of the computation after q queries are linear combinations of q-juntas (i.e. functions of the input that depend on at most q variables). This is true for the initial state $P_0|s^0\rangle$ which doesn't depend on x. Suppose it is true after $q-1$ queries. The effect of the i-th query Q_i^x is to map register $|i, z \text{ XOR } x_i, y\rangle$ to $|i, z, y\rangle$. After the query, the amplitude $a(i,z,y)$ is a sum of $(q-1)$-juntas $J_{i,z \text{ XOR } x_i, y}$ that may depend on x_i but not on the other inputs. So each amplitude is a sum of q-juntas. After this query the algorithm applies a unitary transformation P_q which is linear and does not change the property.

Since the probability of an accepting measurement is a sum of squares of amplitudes, it is a sum of juntas that now depend on at most $2q$ inputs each. When x is a 0/1 input this is a polynomial of degree at most $2q$. By assumption this polymomial approximates f with error at most ε. □

Therefore quantum query complexity is also polynomially related to all other complexity measures from the last lecture:

$$\deg^{1/4} \leq \mathrm{sens}^{1/2} \preceq \tfrac{1}{2}\widetilde{\deg}_{1/3} \leq Q_{1/3} \preceq R_{1/3} \leq R_0 \leq D \leq \deg^2 \cdot \mathrm{bsens} \preceq \deg^2 \cdot \widetilde{\deg}^2_{1/3} \leq \deg^4.$$

To complete the chain of inequalities it remains to prove that $\mathrm{sens}(f) \leq \mathrm{bsens}(f) \leq \Omega\big(\widetilde{\deg}_{1/3}(f)^2\big)$, which we stated without proof in the last lecture. This is a consequence of the next lemma:

Lemma 2. *$\widetilde{\deg}_{1/3}(APXOR) = \Omega(\sqrt{n})$, where $APXOR$ is the n-input partial function that evaluates to 0 at 0^n and to 1 at strings with exactly one 1.*

Any function of block sensitivity n has $APXOR$ on n inputs "embedded" in it. By linearity we may assume the block sensitivity is achieved at zero. If we identify the inputs in every block with the same variable x_i and fix the other inputs to a constant then the degree cannot increase, and the resulting function is consistent with $APXOR$.

One proof of Lemma 2 relies on the following technical claim which you will prove in the homework. Two distributions X and Y on n-bit strings are called *k-wise indistinguishable* if $\mathrm{E}[J(X)] = \mathrm{E}[J(Y)]$ for every k-junta J.

Claim 3. *There is an $\varepsilon > 0$ so that for sufficiently large n, there exist $\varepsilon\sqrt{n}$-wise indistinguishable distributions X and Y such that $\Pr[X \text{ is the all-zero string}] \geq 0.99$ and $\Pr[Y \text{ has exactly one } 1] \geq 0.62$.*

Proof of Lemma 2. For technical reasons we prove the lemma for $\widetilde{\deg}_{0.3}(f)$ which is $O(\widetilde{\deg}_{1/3}(f))$. Assume for contradiction that some total function f extending $APXOR$ can be represented as $p(x) + e(x)$, where p has degree $0.01\sqrt{n}$ and $|e(x)| \leq 0.3$ for every x. By linearity of expectation,

$$\mathrm{E}[f(X)] - \mathrm{E}[f(Y)] = (\mathrm{E}[p(X)] - \mathrm{E}[p(Y)]) + (\mathrm{E}[e(X)] - \mathrm{E}[e(Y)]).$$

By linearity of expectation again, the first term is zero: p is a linear combination of low-degree monomials, each of which is a junta that cannot distinguish X from Y. Since $|e(x)|$ is bounded by 0.3, the second term can be at most 0.6, so

$$\mathrm{E}[f(X)] - \mathrm{E}[f(Y)] \leq 0.6.$$

On the other hand, $\mathrm{E}[f(X)]$ must be at least 0.99 because X places this much probability on the all-zero input. But $\mathrm{E}[f(Y)]$ can be at most $1 - 0.62$ because Y places at least 0.62 probability on the inputs that have exactly one 1. So the difference on the left is at least $0.99 - (1 - 0.62) = 0.61$ which is larger than 0.6, a contradiction. □

References

The presentation of quantum computation is partially based on Chapter 10 in the Arora-Barak textbook. The description of Grover's algorithm also borrows from these lecture notes of Ryan O'Donnell. Lemma 1 was proved by Nisan and Szegedy. The proof outlined here is based on some recent work I did with Mande, Thaler, and Williamson.