Quantum computation is an extension of randomized computation that captures the ability of quantum mechanical devices to occupy a "superposition" of states. Quantum computers have not yet been built but there is reason to hope that they will be in the future. In contrast to the practical difficulties of quantum computing, a well-developed theory of their abilities and limitations has been around since the 1980s and remains an active area of research.

Besides applications like simulating quantum systems in physics and chemistry, quantum computers are interesting because they can be used to solve some problems much faster than classical computers.

1 Reversible computation

One property of quantum mechanical systems is that they must be *reversible*: No loss of information can happen while the system is evolving unobserved. Classical computation is not reversible. I can erase my memory and forget what was written there before.

Deterministic classical computation can be made reversible at the cost of more memory. 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).

A circuit can be made reversible by replacing all gates (instructions) with "reversible gates". It is enough to do this for NOT and AND gates as they form a complete basis. NOT is already reversible, while AND can be simulated by the reversible gate:[1]

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

When the third input register z is initialized to zero the corresponding output is precisely x AND y. The gate is now reversible: Applying T to its own output recovers the original state $|x, y, z\rangle$.

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 applying the same instructions in reverse order).

From a global 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

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

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.[2]

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 query gate can also be implemented reversibly as $Q^x qi, z = |i, z \text{ XOR } x_i\rangle$. 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 $|s^0\rangle = |s_1, \ldots, s_n\rangle$ and a sequence of local permutations $L_1, \ldots, L_t$ 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 $L_1, \ldots, L_t$ 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 $P_0, \ldots, P_q$ are permutations that do not depend on x and $Q_1^x, \ldots, Q_q^x$ 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 $|0s_2 \ldots s_n\rangle$ and $|1s_2 \ldots s_n\rangle$ into the state $|0s_2 \ldots s_n\rangle$ with probability half and $|1s_2 \ldots s_n\rangle$ 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

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

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

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

The transformations from p to p' are *linear* and *stochastic*: Each $p'(s')$ is a nonnegative linear combination of $p(s)$, and $p(s)$ 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.

2 Quantum Computation

In a quantum system with N states, each of the N possible "standard basis" states s corresponds to a vector $|s\rangle$ in (N-dimensional) Hilbert space. These vectors form an orthonormal basis: They are orthogonal

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

vectors of unit length. The state of the quantum system can be any unit vector in 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 $|s\rangle : s \in \{0,1\}^n$. At any given time its state is some superposition 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 unobserved 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 they can. 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 $\sum \sqrt{p(s)} \cdot |s\rangle$: The amplitudes are square roots of the probabilities. Since quantum transformation preserve length, i.e. sums of squares of amplitudes, the probabilities always add up to one. 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 a randomized computation, state $|s\rangle$ is "observed" with probability $p(s)$. At the end of a quantum computation, the quantum computer is in some superposition state $\sum a(s) \cdot |s\rangle$. A non-reversible operation called a *measurement* is then performed in which the value s is sampled with probability $|a(s)|^2$. This measurement destroys the superposition state. If a fresh sample is needed the computation must be restarted.

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 $|a(s)|^2$. (This is a bare-bones model of a quantum computer that can be extended with more features like partial measurements.)

3 The Power of Quantum Queries

To illustrate a quantum ability that is impossible to obtain classically, we give an algorithm for computing the XOR of two bits x_0 XOR x_1 with a single quantum query to x. In this example $N = 4$ and the standard basis states are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. We can think of the state of the quantum computer as consisting of two Boolean registers, or *qubits*.

Prepare the superposition state $|s^0\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$.
Apply a query gate Q^x.
Apply a Hadamard gate H to the first register (denoted by H_1).
Measure and output the first register's value.

The state $|s^0\rangle$ is an example of a *product state*: It can be factored as

$$|s^0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

To analyze this algorithm we can explicitly calculate the final state $H_1 Q^x |s^0\rangle$ for all possible inputs $x \in \{00, 01, 10, 11\}$. Recall that Q^x is the linear transformation $Q^x|iz\rangle = |i(x_i \text{ XOR } z)\rangle$. For example, when $x = 00$, Q^x is the identity and using linearity we get

$$H_1 Q^{00}|s^0\rangle = H_1|s^0\rangle = \left(H\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |0\rangle \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

The first register's value is zero with amplitude one, so the measurement will output zero with probability one. When $x = 01$, Q^x does not change $|0z\rangle$ but sends $|1z\rangle$ to $|1\overline{z}\rangle$. Therefore

$$Q^{01}|s^0\rangle = \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle) = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \cdot \frac{|0\rangle - |1\rangle}{\sqrt{2}},$$

from where $H_1 Q^{01}|s^0\rangle = |1\rangle \cdot (|0\rangle - |1\rangle)/\sqrt{2}$, and the measurement outputs one as desired. You can (and should) verify that $H_1 Q^{10} = -|1\rangle \cdot (|0\rangle - |1\rangle)/\sqrt{2}$ and $H_1 Q^{11} = -|0\rangle \cdot (|0\rangle + |1\rangle)/\sqrt{2}$. In all cases the measurement produces the correct answer.

The analysis checks out but does not explain how one might have discovered this algorithm. One complication in this algorithm is that we have to use two qubits of memory just to implement the query gate Q^x. There is an alternative query gate Φ^x that returns the answer not in a separate register but in the phase of the query itself:

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

This phase-query gate Φ^x can be implemented by the following sequence: Apply H to the answer register, then Q^x, then 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 phase-query gates the action of this algorithm on the first qubit has a simpler description: 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.[3] 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.

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

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

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

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

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

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:
Prepare the initial state $|+\rangle = (|1\rangle + \cdots + |n\rangle)/\sqrt{n}$.
Repeat the following t times:
    Apply a phased query Φ^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 x_{i^*} 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.

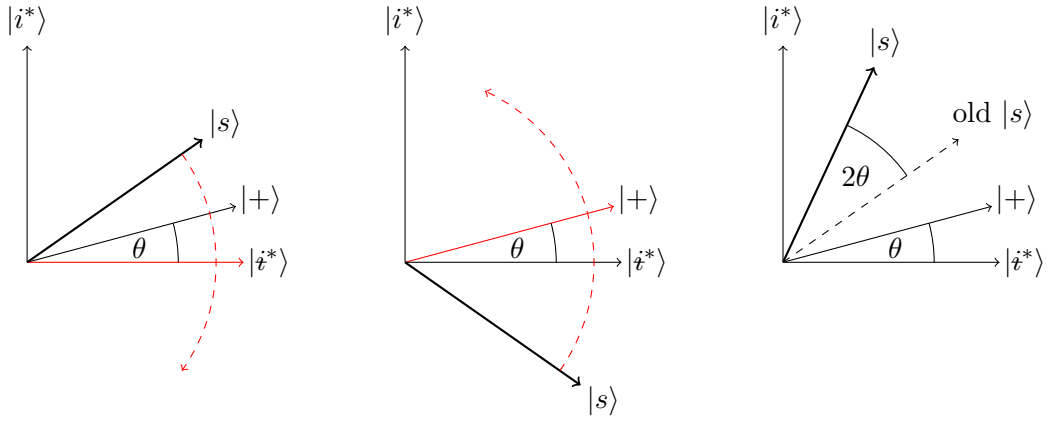
Thus starting with the $|+\rangle$ vector, the state is reflected about the plane perpendicular to $|i^*\rangle$, then about $|+\rangle$, then about the plane perpendicular to $|i^*\rangle$ again, and so on. This vector will always remain in the plane spanned by $|i^*\rangle$ and $|+\rangle$. A reflection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

$$|i^*\rangle = \sqrt{\frac{n}{n-1}}\left(|+\rangle - \frac{1}{\sqrt{n}}|i^*\rangle\right) = \frac{1}{\sqrt{n-1}}\sum_{i\neq i^*}|i\rangle.$$

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

1 Set the initial state $|s\rangle$ to $|+\rangle = (|1\rangle + \cdots + |n\rangle)/\sqrt{n}$.
2 Repeat the following t times:
3 Reflect $|s\rangle$ about $|i^*\rangle$ and then about $|+\rangle$.
4 Measure and output the measured string.

If θ be the angle between $|+\rangle$ and $|i^*\rangle$, 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 $|i^*\rangle$, so $\sin\theta = \langle +|i^*\rangle = 1/\sqrt{n}$. Using the Taylor approximation $\sin\theta = \theta - O(\theta^3)$ we get that $\theta = 1/\sqrt{n} - O(1/n^{3/2})$. After t steps the angle between $|s\rangle$ and $|i^*\rangle$ is $(2t+1)\theta$. If we choose $2t+1$ to be the odd 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 improved? 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. *For every q-query quantum algorithm A there exists a polynomial p of degree at most $2q$ so that $p(x)$ equals the probability that A accepts x for all $x \in \{0,1\}^n$.*

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 $2q$. □

In particular, $\widetilde{\deg}_\varepsilon(f) \leq 2Q_\varepsilon(f)$. Therefore quantum query complexity is also polynomially related to all other complexity measures from the last lecture:

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

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

Lemma 2. *Let $APXOR$ be the n-input partial function that evaluates to 0 at 0^n and to 1 at strings with exactly one 1. If p is a polynomial that $1/3$-approximates $APXOR$ on all such strings and takes values between 0 and 1 on all of $\{0,1\}^n$ then p has degree $\Omega(\sqrt{n})$.*

Therefore $\widetilde{\deg}_{1/3}(APXOR) = \Omega(\sqrt{n})$. 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 $APXOR$. Therefore the $\mathrm{bsens}(f) \le \Omega\big(\widetilde{\deg}(f)^2\big)$ for every function f.

Moreover, it follows from these two lemmas that Grover's algorithm has optimal quantum query complexity:

Corollary 3 (Bennett-Bernstein-Brassard-Vazirani Theorem)**.** *Quantum search of an n-bit database requires $\sqrt{n}$ queries.*

Two distributions X and Y on n-bit strings are *k-wise indistinguishable* if $\mathrm{E}[J(X)] = \mathrm{E}[J(Y)]$ for every k-junta J.

Claim 4. *There is an $\varepsilon > 0$ so that for sufficiently large n, there exist $\varepsilon\sqrt{n}$-wise indistinguishable distributions X and Y such that $\Pr[X$ is the all-zero string$] \ge 0.99$ and $\Pr[Y$ has exactly one $1] \ge 0.62$.*

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

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

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. □

Optional reading: Proof of Claim 4

We represent bits by -1 and 1. Let $\phi\colon \{-1,1\}^n \to \mathbb{R}$ be the function

$$\phi(x) = \Big(\prod\nolimits_{i=1}^{n} x_i\Big) \cdot \mathrm{E}_S\Big[\prod\nolimits_{i\in S} x_i\Big]^2,$$

where S is a random subset of $\{1,\dots,n\}$ of size at most $(n-d)/2$ (chosen uniformly among all such subsets), $d = \varepsilon\sqrt{n}$, and E_S is expected value. The distributions over X and Y are specified by their probability mass functions

$$\mu(x) = \max\{2\phi(x)/Z, 0\} \qquad \text{and} \qquad \nu(x) = \max\{-2\phi(x)/Z, 0\}, \qquad \text{where } Z = \sum|\phi(x)|.$$

Here as in the rest of the proof summations range over all $x \in \{-1,1\}^n$.

μ and ν are probability mass functions: Both take nonnegative values so it suffices to show that $\sum\mu(x) = \sum\nu(x) = 1$. This is a consequence of the following fact:

$\sum \phi(x)p(x) = 0$ for all polynomials p of degree less than d.

The reason is that all monomials in (the multilinear expansion of) ϕ have degree at least d, so the multilinear expansion of $\phi \cdot p$ cannot have a degree-zero term, i.e., $\widehat{\phi p}(\varnothing) = 2^{-n}\sum \phi(x)p(x) = 0$. Plugging in $p = 1$ we obtain $\sum \phi(x) = 0$, so $\sum \mu(x) = \sum \nu(x)$. By the choice of Z, $\sum \mu(x) + \sum \nu(x) = 2$ so each of μ and ν must add up to one.

μ and ν are $(d-1)$-wise indistinguishable: Any junta J that depends on less than d inputs is a polynomial of degree less than d and $\sum \phi(x)J(x) = 0$. Since ϕ is proportional to $\mu - \nu$ we get $\sum \mu(x)J(x) = \sum \nu(x)J(x)$, or equivalently $\mathrm{E}[J(X)] = \mathrm{E}[J(Y)]$.

X is the all-ones string 1^n with probability at least 0.99: X equals 1^n with probability $\mu(1^n) = 2\phi(1^n)/Z = 2/Z$. We can calculate Z exactly:

$$\begin{aligned} Z &= \sum \mathrm{E}_x\Big[\prod\nolimits_{i\in S} x_i\Big]^2 \\ &= \sum \mathrm{E}_S\Big[\prod\nolimits_{i\in S} x_i\Big]\,\mathrm{E}_T\Big[\prod\nolimits_{i\in S} x_i\Big] \\ &= \sum \mathrm{E}_{S,T}\Big[\prod\nolimits_{i\in S\oplus T} x_i\Big] \\ &= \mathrm{E}_{S,T}\sum_{x\in\{-1,1\}^n}\prod\nolimits_{i\in S\oplus T} x_i, \end{aligned}$$

where $\oplus$ is symmetric set difference. The terms $S \neq T$ vanish because the product averages out to zero and take value 2^n when $S = T$. Therefore $Z = 2^n \Pr[S = T]$ and

$$\frac{Z}{2} = 2^{n-1}\Pr[S = T] = 2^{n-1}\Big/\Big(\tbinom{n}{0} + \cdots + \tbinom{n}{(n-d)/2}\Big).$$

In words, $2/Z$ the probability that a random 0/1 string has at most $(n-d)/2$ ones conditioned on it having at most $n/2$ ones. By the Central Limit Theorem when $d = \varepsilon\sqrt{n}$, $2/Z$ approaches the probability that a Normal$(0,1)$ random variable is greater than ε conditioned on it being positive. When ε tends to zero this probability tends to 1, so for ε sufficiently small $\mu(1^n) \geq 0.99$.

Y has exactly one -1 with probability at least 0.62: Let N be the set of strings with exactly one -1 so that the desired probability is $\sum_{x\in N}\nu(x)$. Fix some $x \in N$. Conditioned on the size of S being s (and S being otherwise random), $\prod_{i\in S} x_i$ takes value -1 with probability s/n and 1 with the remaining probability, so $\mathrm{E}_S\big[\prod_{i\in S} x_i\big]^2 = \mathrm{E}[1 - 2|S|/n]^2 = \mathrm{E}[W_n]^2/n$, where W_n is the random variable $W_n = (n - 2|S|)/\sqrt{n}$. Since $\sum_{x\in N}\nu(x)$ is $2/Z$ times the sum of n such terms it equals $2\,\mathrm{E}[W_n]^2/Z$.

The random variable $|S|$ equals the number of ones B in a random n-bit string conditioned on there being at most $(n-d)/2$ ones. Therefore the CDF of w is

$$\begin{aligned} \Pr[W_n \leq w] &= \Pr[|S| \leq (n - w\sqrt{n})/2] \\ &= \Pr[B \leq (n - w\sqrt{n})/2 \mid B \leq (n - \varepsilon\sqrt{n})/2] \\ &= \frac{\Pr[B \leq (n - w\sqrt{n})/2]}{\Pr[B \leq (n - \varepsilon\sqrt{n})/2]}. \end{aligned}$$

By the Central Limit Theorem, the numerator and denominator converge to $\Pr[N \leq w]$ and $\Pr[N \leq \varepsilon]$ respectively, for a Normal$(0,1)$ random variable N. Therefore

$$\lim_{n\to\infty}\Pr[W_n \leq w] = \Pr[N \leq w \mid N \leq \varepsilon].$$

We would therefore expect that

$$\lim_{n\to\infty}\mathrm{E}[W_n] = \mathrm{E}[N \mid N \leq \varepsilon], \tag{3}$$

from where by continuity of the function $\varepsilon \to \mathrm{E}[N \mid N \leq \varepsilon]$ we can calculate

$$\lim_{\varepsilon \to 0} \lim_{n \to \infty} \mathrm{E}[W_n] = \mathrm{E}[N \mid N \leq 0] = \frac{2}{\sqrt{2\pi}} \int_{-\infty}^{0} x e^{-x^2/2} dx = -\sqrt{\frac{2}{\pi}},$$

which gives $\lim_{\varepsilon \to 0} \lim_{n \to \infty} \sum_{x \in N} \nu(x) = 2/\pi > 0.62$.

To be rigorous we should prove (3). It is a bit easier to work with $-W_n$ instead of W_n as $-W_n$ is non-negative. Let us also write N_ε for N conditioned on $N \geq \varepsilon$. By the Central Limit Theorem $\Pr[-W_n \geq w]$ and $\Pr[N_\varepsilon \geq w]$ are δ_n-close for some δ_n that goes to zero as n increases. Using the formula $\mathrm{E}[X] = \int_0^\infty \Pr[X \geq w] dw$ which holds for all non-negative X we can bound the difference in expectations by

$$\begin{aligned}
|\mathrm{E}[-W_n] - \mathrm{E}[N_\varepsilon]| &= \left| \int_0^\infty (\Pr[-W_n \geq w] - \Pr[N_\varepsilon \geq w]) dw \right| \\
&\leq \int_0^B |\Pr[-W_n \geq w] - \Pr[N_\varepsilon \geq w]| dw \\
&\quad + \int_B^\infty \Pr[-W_n \geq w] dw + \int_B^\infty \Pr[N_\varepsilon \geq w] dw.
\end{aligned}$$

Set $B = 1/\sqrt{\delta_n}$. By the Central Limit Theorem the integrand in the first term is at most δ_n so the value of the integral is at most $\sqrt{\delta_n}$. The other two terms can be handled using large deviation bounds. First, $\Pr[N_\varepsilon \geq w] = O(\Pr[N \geq w]) = O(e^{-w^2/2})$. For $\Pr[-W_n \geq w]$ we can apply for instance the Chernoff bound to conclude that it is also $O(e^{-w^2/2})$. Therefore, up to a constant, both integrals are at most $O(e^{-B^2/2}) = O(e^{-1/\delta_n})$. In the limit as δ_n goes to zero, the right-hand terms go to zero and so the expectations approach one another. □

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 Beals, Buhrman, Cleve, Mosca, and de Wolf. Lemma 2 was proved by Nisan and Szegedy. Corollary 3 was previously proved by Bennett, Brassard, Bernstein, and Vazirani using a different method.