Lecture 2: Quantum Algorithms

1 Tensor Products

A single quantum bit is a unit vector in the Hilbert space $\mathscr{C}^2$. Now suppose we have two quantum bits. How do we write them together? We need a new Hilbert space which captures the interaction of the two bits.

If V, W are vector spaces with bases $\{v_1 \dots v_n\}$, $\{w_1 \dots w_m\}$, the *tensor product* $V \otimes W$ of V and W is a nm-dimensional vector space which is spanned by elements of the form $v \otimes w$ - called *elementary tensors*. These elementary tensors behave bilinearly, that is, we have the relations

$$\alpha(v \otimes w) = \alpha v \otimes w = v \otimes \alpha w$$

$$u \otimes v + w \otimes v = (u+w) \otimes v \qquad u \otimes v + u \otimes w = u \otimes (v+w).$$

A basis for the tensor product space consists of the vectors: $\{v_i \otimes w_j : 1 \le i \le n, 1 \le j \le m\}$, and thus a general element of $V \otimes W$ is of the form

$$\sum_{i,j} \alpha_{ij} v_i \otimes w_j$$

This definition extends analogously to tensor products with more than two terms.

The tensor product space is also a Hilbert space with the inherited inner product:

$$(v \otimes w, v' \otimes w') = (v, v')(w, w')$$

As it turns out, a two bit system is conveniently represented by a unit vector in the Hilbert space $\mathscr{C}^2 \otimes \mathscr{C}^2$. $\mathscr{C}^2 \otimes \mathscr{C}^2$ is necessarily isomorphic to $\mathscr{C}^4$ since there is only one complex four dimensional Hilbert space, but as we will see, in the world of quantum mechanics it is convenient to be able to "construct" the larger space from the smaller ones.

Using Dirac "ket" notation, we write the basis of $\mathscr{C}^2 \otimes \mathscr{C}^2$ as

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$$

We will often write $|0\rangle \otimes |0\rangle$ as $|0\rangle|0\rangle$ or $|00\rangle$.

In general, we represent an n-particle system by n copies of $\mathscr{C}^2$ tensored together. We will often write $(\mathscr{C}^2)^{\otimes n} = \mathscr{C}^{2^n}$. So the state of an n-qubit system can be written as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle.$$

This means that the state of an n-particle system is represented by a 2^n dimensional space! The idea behind quantum computation is to harness the ability of nature to manipulate the exponential number of α_xs.

1.1 The Significance of Tensor Products

Classically, if we put together a subsystem that stores k bits of information with one that stores l bits of information, the total capacity of the composite system is $k+l$ bits.

From this viewpoint, the situation with quantum systems is extremely paradoxical. We need k complex numbers to describe the state of a k-level quantum system. Now consider a system that consists of a k-level

subsystem and an l-level subsystem. To describe the composite system we need kl complex numbers. One might wonder where nature finds the extra storage space when we put these two subsystems together.

An extreme case of this phenomenon occurs when we consider an n qubit quantum system. The Hilbert space associated with this system is the n-fold tensor product of $\mathscr{C}^2 \equiv \mathscr{C}^{2^n}$. Thus nature must "remember" of 2^n complex numbers to keep track of the state of an n qubit system. For modest values of n of a few hundred, 2^n is larger than estimates on the number of elementary particles in the Universe.

This is the fundamental property of quantum systems that is used in quantum information processing.

Finally, note that when we actually a measure an n-qubit quantum state, we see only an n-bit string - so we can recover from the system only n, rather than 2^n, bits of information.

1.2 Tensor product of operators

Suppose $|v\rangle$ and $|w\rangle$ are unentangled states on $\mathscr{C}^m$ and $\mathscr{C}^n$, respectively. The state of the combined system is $|v\rangle \otimes |w\rangle$ on $\mathscr{C}^{mn}$. If the unitary operator A is applied to the first subsystem, and B to the second subsystem, the combined state becomes $A|v\rangle \otimes B|w\rangle$.

In general, the two subsystems will be entangled with each other, so the combined state is not a tensor-product state. We can still apply A to the first subsystem and B to the second subsystem. This gives the operator $A \otimes B$ on the combined system, defined on entangled states by linearly extending its action on unentangled states.

(For example, $(A\otimes B)(|0\rangle \otimes |0\rangle) = A|0\rangle \otimes B|0\rangle$. $(A\otimes B)(|1\rangle \otimes |1\rangle) = A|1\rangle \otimes B|1\rangle$. Therefore, we define $(A\otimes B)(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle)$ to be $\frac{1}{\sqrt{2}}(A\otimes B)|00\rangle + \frac{1}{\sqrt{2}}(A\otimes B)|11\rangle = \frac{1}{\sqrt{2}}\left(A|0\rangle \otimes B|0\rangle + A|1\rangle \otimes B|1\rangle\right)$.)

Let $|e_1\rangle, \ldots, |e_m\rangle$ be a basis for the first subsystem, and write $A = \sum_{i,j=1}^m a_{ij}|e_i\rangle\langle e_j|$ (the i,jth element of A is a_{ij}). Let $|f_1\rangle, \ldots, |f_n\rangle$ be a basis for the second subsystem, and write $B = \sum_{k,l=1}^n b_{kl}|f_k\rangle\langle f_l|$. Then a basis for the combined system is $|e_i\rangle \otimes |f_j\rangle$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The operator $A \otimes B$ is

$$\begin{aligned} A\otimes B &= \left(\sum_{ij} a_{ij}|e_i\rangle\langle e_j|\right) \otimes \left(\sum_{kl} b_{kl}|f_k\rangle\langle f_l|\right) \\ &= \sum_{ijkl} a_{ij}b_{kl}|e_i\rangle\langle e_j| \otimes |f_k\rangle\langle f_l| \\ &= \sum_{ijkl} a_{ij}b_{kl}(|e_i\rangle \otimes |f_k\rangle)(\langle e_j| \otimes \langle f_l|) \ . \end{aligned}$$

Therefore the $(i,k),(j,l)$th element of $A \otimes B$ is $a_{ij}b_{kl}$. If we order the basis $|e_i\rangle \otimes |f_j\rangle$ lexicographically, then the matrix for $A \otimes B$ is

$$\begin{pmatrix} a_{11}B & a_{12}B & \cdots \\ a_{21}B & a_{22}B & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \ ;$$

in the i,jth subblock, we multiply a_{ij} by the matrix for B.

2 The Principle of Safe Storage

In the last lecture we learned that performing a measurement changes the state of a quantum system. For example, consider this circuit:

The second Hadamard gate cancels out the first, since $H^2 = I$. If the circuit is given the pure state $|0\rangle$ as input, the bit will again be in the state $|0\rangle$ before the measurement, and so the bit "0" will always be observed:

State before measuring: $\Pr\left[|0\rangle\right] = 1$

$\Pr[0] = 1 \quad \Pr[1] = 0$

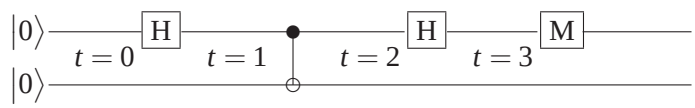
Now, consider the same circuit with an additional measurement inserted between the two gates:

The first measurement collapses the state of the qubit to $|0\rangle$ or $|1\rangle$, so the input to the final measurement is $|+\rangle$ or $|-\rangle$, and are no longer certain to observe the bit 0 in the final measurement. (Recall our notations $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$.)

State before measuring: $\Pr\left[|+\rangle\right] = \dfrac{1}{2} \quad \Pr\left[|-\rangle\right] = \dfrac{1}{2}$

$\Pr[0] = \dfrac{1}{2} \quad \Pr[1] = \dfrac{1}{2}$

What if we replaced the first measurement with CNOT gate, which uses our qubit as the control bit with another qubit as the target?



Let's compute the state of our qubits at each step.

$t = 0 \qquad |0\rangle \otimes |0\rangle$

Recall the Hadamard matrix $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

$t = 1 \qquad |+\rangle \otimes |0\rangle = \dfrac{1}{\sqrt{2}}|00\rangle + \dfrac{1}{\sqrt{2}}|10\rangle$

The CNOT gate acts on the base states by flipping the second bit iff the first bit is one.

$t = 2 \qquad \dfrac{1}{\sqrt{2}}|00\rangle + \dfrac{1}{\sqrt{2}}|11\rangle = |\Phi^+\rangle$

The second Hadamard gate maps $|00\rangle \mapsto |+\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle$ and $|11\rangle \mapsto |-\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|11\rangle$. Adding these together, we have:

$t = 3 \qquad \dfrac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)$

We now see that our final measurement behaves exactly the same way as when we had a measurement in the place of the CNOT gate:

$\Pr[0] = \dfrac{1}{2} \quad \Pr[1] = \dfrac{1}{2}$

3 Quantum Teleportation

3.1 The No Cloning Theorem

The *No Cloning Theorem* states that no quantum system can copy a qubit; that is, there is no unitary operator U sending $\left|\psi\right\rangle \otimes \left|0\right\rangle \mapsto \left|\psi\right\rangle \otimes \left|\psi\right\rangle$.

Proof: Suppose our operator U exists. Then for any states $\left|\psi_1\right\rangle$ and $\left|\psi_2\right\rangle$,

$$\begin{aligned}\left\langle\psi_1\middle|\psi_2\right\rangle =&(\left|\psi_1\right\rangle \otimes \left|0\right\rangle, \left|\psi_2\right\rangle \otimes \left|0\right\rangle)\\ =&(U(\left|\psi_1\right\rangle \otimes \left|0\right\rangle), U(\left|\psi_2\right\rangle \otimes \left|0\right\rangle))\\ =&(\left|\psi_1\right\rangle \otimes \left|\psi_1\right\rangle, \left|\psi_2\right\rangle \otimes \left|\psi_2\right\rangle)\\ =&\left\langle\psi_1\middle|\psi_2\right\rangle^2,\end{aligned}$$

which is impossible. □

3.2 Quantum Teleportation

Despite the No Cloning Theorem, it is possible to transmit a qubit, even to a remote location, if we are willing to destroy the original.

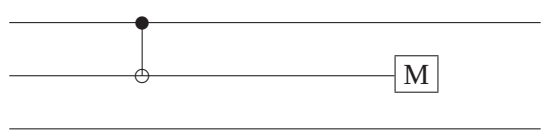
Suppose A has access to a quantum state $|\psi\rangle = a_0|0\rangle + a_1|1\rangle$, which she wants to transmit to a remote party B. She can accomplish this by transmitting only classical bits of information, provided A and B share the entangled two-qubit state

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

The technique is known as *quantum teleportation.*

The basic idea is this. A controls $|\psi\rangle$ and the first qubit of $|\phi\rangle$. A's strategy, roughly speaking, is to forcibly entangle $|\psi\rangle$ with the first qubit $|\phi\rangle$. A then measures the first qubit of $|\phi\rangle$, resolving it completely, and hopes this will cause $|\psi\rangle$ to become entangled with the *second* qubit of $|\phi\rangle$. Presumably, B could then transfer $|\psi\rangle$ to the second qubit of $|\phi\rangle$.

As a first try, consider the following diagram. The top line represents $|\psi\rangle$; the bottom two represent the two qubits of $|\phi\rangle$.



That is, A passes $|\psi\rangle$ and the first qubit of $|\phi\rangle$ through a CNOT gate, and then measures the first qubit of $|\phi\rangle$. Now the input into the system as a whole is

$$|\phi\rangle \otimes |\psi\rangle = \sum_{i=0,1} a_i|i\rangle \otimes \sum_{j=0,1} \frac{1}{\sqrt{2}}|j,j\rangle.$$

After passing through the CNOT gate this becomes

$$\sum_{i,j} a_i\left|i, i\oplus j, j\right\rangle.$$

Now A measures the middle qubit. Suppose it is measured as l; then $l = i \oplus j$. The state is now

$$\sum_j a_{j\oplus l} \big| j \oplus l, j \big\rangle.$$

Next, A transmits l to B. If $l = 0$, B takes no action, while if $l = 1$, then B performs a bit flip on his qubit (the bottom qubit in the diagram.) A bit flip is just the transformation $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Thus we have

$$\sum_j a_{j\oplus l} \big| j, j \big\rangle.$$

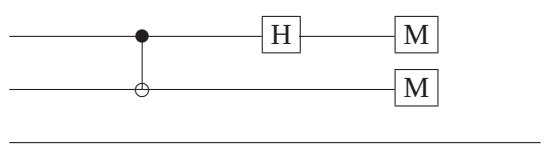
Finally, B does a phase flip on his qubit, yielding

$$\sum_j a_j \big| j, j \big\rangle.$$

This is almost exactly what we want. The only problem is that now, the qubit corresponding to $|\psi\rangle$ is entangled with B's qubit. The entanglement that was necessary to get the whole process started is now a liability. One way to disentangle them would be for A to measure her remaining qubit. But this would destroy B's qubit as well.

The ideal solution would be to send the entangle qubits through a CNOT gate—but A controls the first qubit and B controls the second. This would require quantum communication between A and B, which is prohibited.

The correct solution is to go back and modify the original diagram, inserting a Hadamard gate and an additional measurement:



Now the algorithm proceeds exactly as before. However A's application of the Hadamard gate now induces the transformation

$$\sum_j a_j \big| j, j \big\rangle \longrightarrow \sum_{ij} a_j (-1)^{ij} \big| i, j \big\rangle.$$

Finally A measures i and sends the measurement to B. The state is now:

$$\sum_j a_j (-1)^{ij} |j\rangle.$$

If $i = 0$ then we are done; if $i = 1$ then B applies a phase flip. In either case the state is now $a_0|0\rangle + a_1|1\rangle$.

So A has transported the quantum state to B simply by sending two classical bits.

4 Quantum Circuits and the class BQP

In the previous lecture, we saw some examples of quantum gates. A quantum circuit is a sequence of gates composed together. Each gate acts on a subset of the bits, and leaves the rest unchanged. For example, a

CNOT gate acting on the first two bits of an n-bit system corresponds to the unitary transformation CNOT $\otimes$ I_{n-2}. If the ith gate performs the transformation U_i, then the whole circuit performs the transformation $U = U_T U_{T-1} \cdots U_1$. We measure the efficiency of the circuit by the number of gates T.

At this point it is natural to ask what gates U_i we allow when constructing our circuit. We will show later that there are small "universal" families of gates, such that any $k \times k$ unitary matrix can be approximated by a circuit of size (k) composed of gates from the family. (Note that in an n-qubit system, $k = 2^n$.) CNOT and arbitrary single-qubit operations together form one such family.

To perform a computation using a quantum circuit, we prepare an input state $|\psi\rangle$, apply the circuit, and perform a measurement on the result $U(|\psi\rangle)$. We usually allow the circuit a small probability of giving the wrong answer. The class of decision problems that can be solved with polynomial-size quantum circuits, where the final measurement gives the wrong answer with probability at most $1/3$, is called BQP. Since inputs have different lengths, we allow a family of circuits $\{U_n\}$. For uniformity, we assume there exists an algorithm that on input n constructs the circuit U_n, in time polynomial in n.

BQP is analagous to the classical complexity class BPP, of classical algorithms that can flip coins during their computation, and must give the right answer in the end with probability at least $2/3$. In the classical case, the probability of error can be reduced exponentially by running the algorithm many times and taking the answer that appears a majority of the time. A natural question to ask is whether quantum algorithms can likewise reduce their probability of error – we'll come back to this later on.

5 Reversible Computation

Another question we might ask is whether P $\subseteq$ BQP. Is there a method for converting classical algorithms to quantum ones? One problem is that since quantum operations are always unitary, quantum computations cannot erase information: given a function $f : \{0,1\}^n \to \{0,1\}^n$, unless f is a bijection, we cannot construct a quantum circuit that given x as input outputs $f(x)$.

Our solution is to convert a classical circuit for computing f into a reversible circuit R_f. A reversible circuit is one in which each gate computes a bijection, and can therefore be reversed to compute its input from its output. Given our classical circuit for f, we construct a reversible circuit which takes as input x and a string of zeroes, and outputs x, $f(x)$ and some extra output junk(x). We do this by replacing each classical gate with a reversible equivalent: for example, NOT gates are already reversible, and an AND gate can be replaced by a gate which takes x, y and 0 as input, and outputs x, y and $x \wedge y$.

In a reversible circuit, we can even eliminate the extra output junk(x) by replacing it with a string of zeroes. Given a reversible circuit R_f mapping $x, \vec{0} \mapsto x, f(x), \text{junk}(x)$, we construct its inverse R_f^{-1}. Then on input $x, \vec{0}$, we first apply R_f to get $x, f(x), \text{junk}(x), \vec{0}$. Then we copy the value $f(x)$ onto some of the zeros; we could use CNOT gates for this. Now we have $x, f(x), \text{junk}(x), f(x)$. Applying R_f^{-1} gives us $x, \vec{0}, f(x)$, which is what we wanted.

Reversible gates correspond to unitary transformations that permute the basis states, so given a reversible circuit for computing f, we can construct a quantum circuit U_f for computing f. Whenever the input $|\phi\rangle$ is a basis state, the output $U_f\left(|\phi\rangle\right)$ is also a basis state.

In general, though, we don't need to feed U_f a classical state $|x\rangle$. If we feed U_f a superposition

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle |0\rangle$$

then, by linearity,

$$U_f\left(\sum_{x\in\{0,1\}^n}\alpha_x\left|x\right\rangle\left|0\right\rangle\right)=\sum_{x\in\{0,1\}^n}\alpha_x U_f\left(\left|x\right\rangle\left|0\right\rangle\right)=\sum_{x\in\{0,1\}^n}\alpha_x\left|x\right\rangle\left|f(x)\right\rangle$$

and we've computed $f(x)$ simultaneously for each basis state $|x\rangle$ in the superposition. Note that if we had not been able to eliminate the extra output junk(x) from the output of our circuit, we would have ended up with the quantum state $\sum_{x\in\{0,1\}^n}\alpha_x\left|x\right\rangle\left|f(x)\right\rangle\left|\text{junk}(x)\right\rangle$ instead, which is a very different.

The procedure for converting classical circuits into quantum circuits U_f is a useful primitive which we will use extensively in this course. A second primitive, introduced in the last lecture, is the Hadamard transform H, also called the Fourier transform. H is the unitary transformation on one qubit defined by the matrix

$$\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix}.$$

In other words, H maps $|0\rangle$ onto $\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$, and $|1\rangle$ onto $\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle$. Given an n-qubit system, we'll often want to apply H separately to each of the qubits in turn. We call the resulting transformation $H_{2^n}=H^{\otimes n}$; it equals H tensored with itself n times. For all $n>0$, $H^{\otimes n}$ can also be defined recursively by the matrix

$$\begin{pmatrix}\frac{1}{\sqrt{2}}H^{\otimes n-1} & \frac{1}{\sqrt{2}}H^{\otimes n-1}\\ \frac{1}{\sqrt{2}}H^{\otimes n-1} & -\frac{1}{\sqrt{2}}H^{\otimes n-1}\end{pmatrix}.$$

where $H^{\otimes 0}=\begin{pmatrix}1\end{pmatrix}$. (Incidentally, it is an exercise to show that for all $n\times n$ matrices U and V, if U and V are unitary then the tensor product $U\otimes V$ is also unitary.)

6 The Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm was published in 1992, and provided one of the first formal indications that quantum computers can solve some problems more efficiently than classical ones.

Suppose we're given a Boolean function $f:\{0,1\}^n\to\{0,1\}$. We're promised that f is either identically zero (meaning that $f(x)=0$ for all inputs $x\in\{0,1\}^n$), or else balanced (meaning that $f(x)=0$ for precisely half of all inputs x, and $f(x)=1$ for the other half). The challenge is to decide which is the case. To do this, we can make queries of the form "Is $f(x)$ equal to 1?" for particular values of x. Our goal is to minimize the number of queries that have to be made.

Clearly, a deterministic classical algorithm requires $2^{n-1}+1$ queries in the worst case: if 2^{n-1} bits have been queried and all turned out to be 0, we still need to query one more bit to decide whether the function is zero or balanced.

On the other hand, the problem admits an efficient randomized algorithm, as follows. Choose a value of x uniformly at random; if $f(x)=1$ then conclude that f is balanced, otherwise repeat. If, after k iterations (for some constant k), we still haven't found an x such that $f(x)=1$, then we halt and conclude that f is zero. This algorithm uses only $O(1)$ queries. However, has a nonzero probability of error, equal to

2^{-k}, or slightly less if we choose x values without replacement. This is because, even if f is balanced, the randomized algorithm has a nonzero probability of never seeing an x such that $f(x) = 1$.

What Deutsch and Jozsa showed is that a quantum computer can decide whether f is balanced, with certainty, using only two queries to f. What follows is an algorithm to accomplish this.

The algorithm uses two quantum registers, the first having n qubits and the second having only one qubit. We initialize the system to the basis state $|0\cdots 0\rangle |0\rangle$. Then we apply the Hadamard transform H_{2^n} to the first register (or equivalently, the Hadamard transform H to each qubit of the first register separately). This results in the superposition

$$\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle |0\rangle .$$

We then compute $f(x)$ and store the result in the second register. How is this done? If f is a black-box oracle, then (by assumption) we don't need to worry about how it's done. If, on the other hand, f is given explicitly (say, as a Boolean circuit), then we've seen that we can compute f reversibly using Fredkin gates. In either case, the resultant superposition is

$$\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle |f(x)\rangle .$$

Next we apply the one-qubit transformation

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

to the second register. This transformation, which is easily seen to be unitary, is a "phase flip": if $f(x) = 0$ then it leaves the amplitude of $|x\rangle |f(x)\rangle$ alone, whereas if $f(x) = 1$ then it inverts the amplitude. So we get

$$\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle |f(x)\rangle .$$

At this point we want to erase $f(x)$ from the second register, to allow for proper interference among the states. (What goes wrong if we don't erase $f(x)$ is left as an exercise.) We can't do this directly, since erasure is not a unitary operation. But since we have an oracle for f, we can simply compute f a second time and CNOT the result into the second register, so that the bit in that register is $f(x) \oplus f(x) = 0$. Doing so, we obtain

$$\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle |0\rangle .$$

The final step is to perform another Hadamard transform H_{2^n} on the first register. We could work out the result of this algebraically, but would rather reason about it to obtain more insight. We've seen that the Hadamard transform is its own inverse, and that

$$H_{2^n}(|y\rangle) = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} (-1)^{y \cdot x} |x\rangle .$$

From these it follows that if f is identically zero, then applying H_{2^n} brings us back to $|0 \cdots 0\rangle |0\rangle$. If, on the other hand, f is balanced, then the state

$$\frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$$

is a linear combination of $H_{2^n}(|y\rangle)$ for various values of y. However, none of these values can be $y = 0$, since the state is orthogonal to $H_{2^n}(|0\rangle)$ in the vector space Z_2^n (the two having inner product $2^{n-1} - 2^{n-1} = 0$).

Therefore, if f is zero, then $x = 0$ at the end of the computation, whereas if f is balanced, then $x \neq 0$. So by observing the $|x\rangle$ register, we can decide with certainty whether f is zero or balanced. We've done this using two queries to f and $\Theta(n)$ steps of computation.