0.1 Extended Church-Turing Thesis

The extended Church-Turing thesis is a foundational principle in computer science. It asserts that any "reasonable" model of computation can be efficiently simulated on a standard model such as a Turing Machine or a Random Access Machine or a cellular automaton. This thesis forms the foundation of complexity theory — for example ensuring that that the class P (polynomial time) is well defined. But what do we mean by "reasonable"? In this context, reasonable means "physically realizable in principle". One constraint that this places is that the model of computation must be digital. Thus analog computers are not reasonable models of computation, since they assume infinite precision arithmetic. In fact, it can be shown that with suitable infinite precision operations, an analog computer can solve NP-Complete problems in polynomial time. And an infinite precision calculator with operations +, x, =0?, can factor numbers in polynomial time.

We have already seen that quantum computers are digital computers, and therefore a reasonable model of computing. But we also established that $P \subseteq BPP \subseteq BQP \subseteq PSPACE$. Since we do not know how to separate P from $PSPACE$, it follows that we cannot unconditionally prove that quantum computers are more powerful than classical computers. Instead we will show that there is an oracle relative to which a particular problem *recursive fourier sampling* is in BQP but is not in BPP. This will complete a demonstration that quantum computers violate the Extended Church-Turing thesis.

1 Fourier Sampling

Consider a quantum circuit acting on n qubits, which applies a Hadamard gate to each qubit. i.e. the circuit applies the unitary transformation $H^{\otimes n}$, or H tensored with itself n times.

Another way to define this unitary transformation H_{2^n} is as the $2^n \times 2^n$ matrix in which the (x,y) entry is $2^{-n/2}(-1)^{x\cdot y}$.

Applying the Hadamard transform (or the Fourier transform over Z_2^n) to the state of all zeros gives an equal superposition over all 2^n states

$$\mathscr{H}_{2^n}|0\cdots 0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle .$$

In general, applying the Hadamard transform to the computational basis state $|u\rangle$ yields:

$$\mathscr{H}_{2^n}|u\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{u\cdot x}|x\rangle$$

We define the Fourier sampling problem as follows: Input an n qubit state $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$. Compute $H^{\otimes n}|\phi\rangle$ and measure the resulting state $\sum_y \hat{\alpha}_y |y\rangle$ to output y with probability $|\hat{\alpha}_y|^2$.

This problem is easy to solve on a quantum computer, but appears to be hard to solve classically. The recursive fourier sampling problem gives a way of showing that in general, fourier sampling is indeed difficult.

2 Phase State

We will now see how to set up an interesting state for fourier sampling. Given a classical circuit for computing a boolean function $f : \{0,1\}^n \to \{0,1\}$, this procedure due to Deutsch and Jozsa, shows how to transform it into a quantum circuit that produces the quantum state $\left|\phi\right\rangle = 1/2^{n/2} \sum_x (-1)^{f(x)} \left|x\right\rangle$.

The quantum algorithm to carry out this task uses two quantum registers, the first consisting of n qubits, and the second consisting of a single qubit.

- Start with the registers in the state $|0^n\rangle\,|0\rangle$
- Compute the Fourier transform on the first register to get $\sum_{x\in\{0,1\}^n} |x\rangle \otimes |0\rangle$.
- Compute f to get $\sum_x |x\rangle\,|f(x)\rangle$.
- Apply a conditional phase based on $f(x)$ to get $\sum_x (-1)^{f(x)} |x\rangle\,|f(x)\rangle$.
- Uncompute f to get $\sum_x (-1)^{f(x)} |x\rangle \otimes |0\rangle$.

3 Extracting n bits with 2 evaluations of Boolean Function

Suppose we are given a black box (or an obfuscated classical circuit) that computes the function function $f_s : \{0,1\}^n \to \{1,-1\}$, where $f(x) = s\cdot x$. $s\cdot x$ denotes the dot product $s_1x_1 + \cdots + s_nx_n \bmod 2$. The challenge is to use this black box to efficiently determine s.

It is easy to see how to perform this task with n queries to the black box: simply input in turn the n inputs x of Hamming weight 1. The outputs of the black box are the bits of s. Since each query reveals only one bit of information, it is clear that n queries are necessary.

Remarkably there is a quantum algorithm that requires only two (quantum) queries to the black box:

- Use the black box to set up the phase state $\left|\phi\right\rangle = 1/2^{n/2} \sum_x (-1)^{f(x)} \left|x\right\rangle$.
- Apply the Fourier transform $H^{\otimes n}$ and measure. The outcome of the measurement is s.

To see that the outcome of the measurement is s, recall that $H^{\otimes n}\left|s\right\rangle = 1/2^{n/2} \sum_x (-1)^{s\cdot x} \left|x\right\rangle = \left|\phi\right\rangle$. Since $H^{\otimes n}$ is its own inverse, it follows that $H^{\otimes n}\left|\phi\right\rangle = \left|s\right\rangle$.

More generally, the transformation $H^{\otimes n}$ maps the standard basis $\left|s\right\rangle$ to the fourier basis $\left|\phi_s\right\rangle = 1/2^{n/2} \sum_x (-1)^{s\cdot x} \left|x\right\rangle$ and vice-versa.

We have shown that a quantum algorithm can be more efficient than any probabilistic algorithm in terms of the number of queries. One way to use this difference in the number of queries in order to demonstrate a gap between quantum and probabilistic algorithms is to make the queries very expensive. Then the quantum algorithm would be $n/2$ times faster than any probabilistic algorithm for the given task. But this does not help us in our goal, which is to show that quantum computers violate the extended Church-Turing thesis. The idea behind proving a superpolynomial gap (which we will outline below) is to make each query itself be the answer to a Fourier sampling problem. Now each query itself is much easier for the quantum algorithm than for any probabilistic algorithm. Carrying this out recursively for $\log n$ levels leads to the superpolynomial speedup for quantum algorithms.

Before going through more details let us first show another consequence of our quantum fourier sampling: a simple proof that the inner product function has linear communication complexity.

4 Communication Complexity of Inner Product Function

5 Recursive Fourier Sampling

Our goal is to give a superpolynomial separation between quantum computation and classical probabilistic computation. The idea is to define a recursive version of the fourier sampling problem, where each query to the function (on an input of length n) is itself the answer to a recursive fourier sampling problem (on an input of length n/2). Intuitively a classical algorithm would need to solve n subproblems to solve a problem on an input of length n (since it must make n queries). Thus its running time satisfies the recurrence $T(n) \geq nT(n/2) + O(n)$ which has solution $T(n) = \Omega(n^{\log n})$. The quantum algorithm needs to make only two queries and thus its running time satisfies the recurrence $T(n) = 2T(n/2) + O(n)$, which solves to $T(n) = O(n \log n)$.

Here is how it works for two levels: we are given a black box computing a function $f : \{0,1\}^{3n/2} \to \{0,1\}$, with the promise that for every n bit string x, the function $f_x : \{0,1\}^{n/2} \to \{0,1\}$ defined by $f_x(y) = f(xy)$ (xy is denotes the concatenation of x and y) satisfies $f_x(y) = s_x \cdot y$ for some $s_x \in \{0,1\}^{n/2}$. We are also given a black box $g : \{0,1\}^{n/2} \to \{0,1\}$ which satisfies the condition that if we construct a boolean function h on n bits as $h(x) = g(s_x)$, then $h(x) = s \cdot x$ for some n-bit string s. The challenge is to figure out s.

A quantum algorithm use fourier sampling recursively as follows ...

The proof that no classical probabilistic algorithm can reconstruct s is somewhat technical, and establishes that for a random g satisfying the promise, any algorithm (deterministic or probabilistic) that makes $n^{o(\log n)}$ queries to g must give the wrong answer on at least $1/2 - o(1)$ fraction of g's. This lemma continues to hold even if the actual queries are chosen by a helpful(but untrusted) genie who knows the answer.

This establishes that relative to an oracle $BQP \not\subseteq MA$. MA is the probabilistic generalization of NP. It is conjectured that recursive fourier sampling does not lie in the polynomial hierarchy. In particular, it is an open question to show that, relative to an oracle, recusive fourier sampling does not lie AM or in BPP^{NP}. The latter class is particularly important since it includes approximate counting. open question whether