Quantum Algorithms, Spring 2022: Lecture 7 Scribe

Hrishi Narayanan and Sabyasachi Mukhopadhyay

February 13, 2022

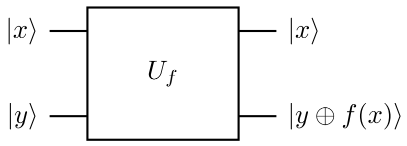
1 Recap

1.1 Deustch-Josza Algorithm

Problem Statement: Given a black-box u_f for $f : \{0,1\}^n \to \{0,1\}$, with the promise that f is either constant or balanced. How many queries we need to make to u_f to determine which is the case?

Classical Deterministic: $2^n/2 + 1$ is the worst case

Classical Probabilistic: $O(log(\frac{1}{\varepsilon}))$, for the success probability $\geqslant 1 - \varepsilon$



Quantum: 1 query

Summary: A function is given to a black-box, Quantum Algorithm needs 1 query, randomized Classical Algorithm needs a reasonable amount of time. Deustch-Josza is in the class BPP, wrt the deterministic classical algorithm we achieve exponential speedups but not wrt the randomized Classical Algorithm.

1.2 Bernstein-Vazirani Algorithm

Problem Statement: Given a black-box u_f for $f : \{0,1\}^n \to \{0,1\}$, with $f = s \cdot x \mod 2$ and x is the inputs, such that s is unknown, s being a n-bit string. How many queries do we need to make to u_f to determine s?

Classical: C_f is queried n times with inputs $(1\dots 0, 01\dots 0, \dots, 0\dots 1)$ to obtain $s_1, s_2, \dots, s_n$ respectively. So classically it taken n queries to determine a n-bit string s.

Quantum: In contrast to the classical solution which needs at least n-queries of the function to find s, only one query is needed using quantum computing. The quantum algorithm is as follows.

$$|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \xrightarrow{U_f^{\pm}} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)+x\cdot y} |y\rangle = |s\rangle \tag{1}$$

2 Basic differences between Classical and Quantum Algorithms

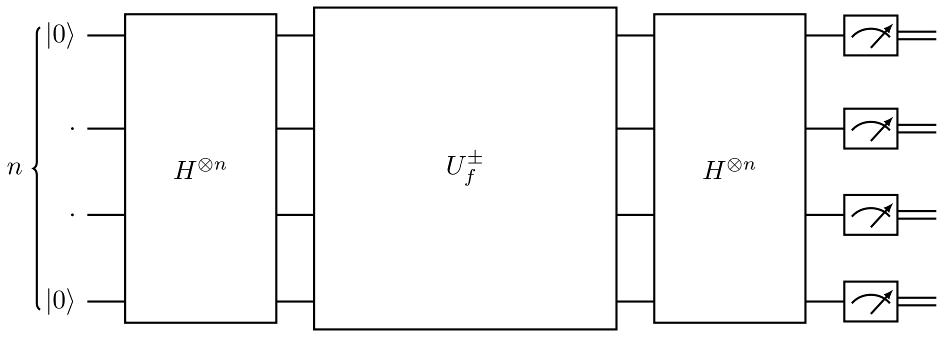
While we are not yet able to show that BPP $\neq$ BQP there are three approaches that we can pursue to study the differences between the capabilities of classical and quantum algorithms.

- **Non-exponential speedup**: We can find quantum algorithms that are demonstrably faster than the best classical algorithm but not exponentially faster. These algorithms shed no light on the conventional classification of complexity but they do demonstrate a type of separation between tasks that classical and quantum computers can perform Example: Grover's quantum speedup of the search of an unsorted data base

- **"Relativized" exponential speedup**: We can consider the problem of analysing the contents of a “quantum black box”. The box performs a priory unknown) unitary transformation. We can prepare an input for the box and measure its output; our task is to find out what the box does. It is possible to prove these Quantum Black boxes aka Oracles exist with this property: By feeding quantum superpositions to the box, we can learn what is inside with an exponential speedup, compared to how long it will take if we were allowed only classical inputs. A computer scientist could say BPP $\neq$ BQP "relative to the oracle", example: simon’s exponential quantum speedup for finding the period of a 2 to 1 function.
- **Exponential speedup for apparently hard problems**: We can exhibit a quantum algorithm that solves a problem in polynomial time , where the problem appears to hard classically, so it is strongly suspected(though not proved) that the problem is not in BPP. Example Shor’s factoring algorithm.

3 Bernstein-Vazirani Quantum Algorithm (elaboration)

The circuit for Bernstein-Vazirani Quantum algorithm is exactly same as that of the Deustch-Josza Algorithm.



3.1 The Algorithm

1. Perform a Hadamard gate operation on the n-qubit input state.
2. Pass the output of the Hadamard to a phase-kickback oracle u_f^{+-}.
3. Perform a Hadamard gate operation on the output.
4. Do a measurement.

3.2 Analysis

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} |x\rangle \xrightarrow{u_f^{+-}} \frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} (-1)^{f(x)} |x\rangle \tag{2}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} (-1)^{s\cdot x \mod 2} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} (-1)^{s\cdot x} |x\rangle = H^{\otimes n}|s\rangle \tag{3}$$

$$\frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} (-1)^{s\cdot x \mod 2} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} (-1)^{s\cdot x} |x\rangle \tag{4}$$

as keeping or removing $\mod 2$ won’t make a difference because $(-1)^{s\cdot x \mod 2} = (-1)^{s\cdot x}$ as, if $s\cdot x$ is odd $(-1)^{s\cdot x \mod 2}$ will give -1 and if $s\cdot x$ is even, $s\cdot x \mod 2 = 0$ so $(-1)^{s\cdot x \mod 2}$ will give 1, so effectively $(-1)^{s\cdot x \mod 2} \approx (-1)^{s\cdot x}$

$$\frac{1}{\sqrt{2^n}} \sum_{x\in(0,1)^n} (-1)^{s\cdot x} |x\rangle = H^{\otimes n}|s\rangle \tag{5}$$

$$\text{as } H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z\in(0,1)^n} (-1)^{x\cdot z}|z\rangle \tag{6}$$

As the Hadamard gate H is represented by $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$ we know that Hadamard gate is Hermetian i.e $H^\dagger = H$, we also know that for any unitary matrix $uu^\dagger = \mathbb{I}$, hence $u^\dagger = u^{-1}$, since Hadamard is an unitary operation $H = H^\dagger = H^{-1}$ or $H^2 = \mathbb{I}$ (identity matrix). So $H^{\otimes n}H^{\otimes n} = (H\otimes H\otimes H\otimes ...)(H\otimes H\otimes H\otimes ...) = H^2\otimes H^2\otimes ... = \mathbb{I}$, giving $H^{\otimes n}H^{\otimes n} = \mathbb{I}$.

As per the step 3, when we perform a Hadamard operation on the output we get $H^{\otimes n}(H^{\otimes n}|s\rangle) = (H^{\otimes n}H^{\otimes n})|s\rangle = |s\rangle$, we know $H^{\otimes n}H^{\otimes n} = \mathbb{I}$, so $H^{\otimes n}(H^{\otimes n}|s\rangle) = |s\rangle$.

So, when we measure at step 4, we get $|s\rangle$, with constant probability = 1. Number of queries to $u_f^{\pm} = 1$, we see a polynomial separation, i.e n separation in query complexity, since the classical algorithm taken n queries. This is unlike the Deusth-Jozsa case where the exponential separation is lost when compared to the classical randomized algorithm.

Bernstein and Vazirani also defined a recursive version of this problem, which can be solved exactly by a quantum algorithm in poly(n) steps, but for which every classical randomized algorithm needs $n^{\Omega(logn)}$ steps.

4 Simon's Algorithm

4.1 Introduction and Historical perspective

Simon's algorithm is the first involved Quantum Algorithm, given by Daniel Simon in 1994, it is a very influential Algorithm because Shor's famous algorithm followed Simon's algorithm and uses many of the things which Simon developed in his algorithm.

Historically, when Simon published his paper to STOC, this was rejected. Shor, who was in the advisory committee of the conference contacted Simon and understood the result of Simon's Algorithm and started developing the Shor's algorithm. He contact Simon and submitted his paper on Shor's algorithm to FOCS with the promise that if his paper was accepted they will merge their papers. But both their papers were accepted in FOCS conference 1994.

The algorithm is a beautiful exposition of Quantum computing and provided an inspiration for Shor's factoring Algorithm.

4.2 Comparison with Deutsch-Jozsa and Bernstein-Vazirani algorithms

The Deutsch-Jozsa problem showed an exponential quantum improvement over the best deterministic classical algorithms; the Bernstein-Vazirani problem showed a polynomial improvement over the best randomized classical algorithms that have error probability $\leqslant \frac{1}{3}$.

Simon's algorithm combines these two features it shows quantum computers are provably exponentially more efficient (in terms of number of queries) than bounded-error randomized algorithms.

4.3 Problem Statement

Given a blackbox u_f, for a function f from n bit-strings to n bit-strings i.e $f : \{0,1\}^n \to \{0,1\}^n$, with the promise that f is $2-1$ function defined as follows:$\forall$ input $(x, y) \in \{0,1\}^n, f(x) = f(y), \iff x = y \oplus s.$

Here, $s \in (0,1)^n$ is secret string. How many queries we need to make to u_f to determine s?

- $f(x) = f(y), \iff x = y \oplus s$
- $x \neq y \oplus s$, then $f(x) \neq f(y)$
- For any s $\neq 0^n$, f(x) is two-one, other wise if s was 0, function would have been one-one, i.e $(x = y)$
- $x = y \oplus s \iff x \oplus y = s(x \oplus y = (y \oplus s) \oplus y = s)$

- The points in the domain of the function where f(x) is equal, if we XOR the two positions x and y, we get s
- Indeed, if for some $x, y : f(x) = f(y)$, we not only have $x = y \oplus s$, we also have $s = x \oplus y$
- Finding s then reduces to finding a collision pair, i.e for some (x_i, x_j) such that $f(x_i) = f(x_j)$ where $[x_i, x_j \in \{0,1\}^n]$

4.3.1 Example

Consider $f : \{0,1\}^3 \to \{0,1\}^3$, and $s = 110$

x	f(x)
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

1. We take two collisions: $f(000) : f(x) = f(110) : f(y)$

$$x = y \oplus s = 110 \oplus 110 = 000$$

$$s = x \oplus y = 000 \oplus 110 = 110$$

2. We take another two collisions: $f(011) : f(x) = f(101) : f(y)$

$$x = y \oplus s = 101 \oplus 110 = 011$$

$$s = x \oplus y = 011 \oplus 101 = 110$$

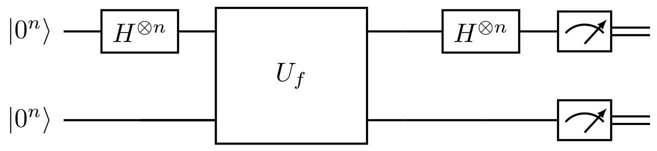
4.4 Classical Query Complexity

What is the classical complexity of collision finding?

- Suppose we pick T values of $x \in \{0,1\}^n$, uniformly at random $\{x_1, x_2, ...x_T\}$ and we make T queries to C_f to obtain $\{f(x_1), f(x_2), ...f(x_T)\}$.
- If we obtain a collision with this T queries, i.e we get some (x_i, x_j), such that $f(x_i) = f(x_j)$, we obtain s as $x_i \oplus x_j = s$
- How many pairs of (x_i, x_j) we obtain if we make T queries to C_f, we obtain $\binom{T}{2}$ pairs $= \frac{(T)(T-1)}{2}$ pairs
- What is the probability that any fixed pair (x_i, x_j) has a collision?
    1. The number of possible bit strings possible out of $[x_i, x_j \in \{0,1\}^n] = 2^n$
    2. We know that $s \neq 0^n$, so number of options available is $2^n - 1$
    3. $x_j \oplus y_j$ can have any non-zero bit strings out of which 1 bit-string is s, so probability of finding a collision is $P(collision) = \frac{1}{2^n - 1}$
- Expected number of collisions after making T queries is $= \frac{1}{2^n-1}\binom{T}{2} = \frac{T(T-1)}{2(2^n-1)} \approx \frac{T^2}{2^{n+1}}$
- We want at least one collision, so $\frac{T^2}{2^{n+1}} \geqslant 1 \implies T \geqslant \sqrt{2^{n+1}}$
- So classically we will need $\mathcal{O}(2^n)$ queries
- There also exists a $\Omega(2^n)$ lower bound. Thus the analysis is tight and classically we need exponentially many queries.

4.5 Quantum Algorithm

Simon's algorithm starts with the input $|0^n\rangle \otimes |0^n\rangle = |0^n\rangle|0^n\rangle$, where $|0^n\rangle$ is the quantum state with n zeros.



After first Hadamard operation

$$(H^{\otimes n}|0^n\rangle) \otimes |0^n\rangle = \Big(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle\Big) \otimes |0^n\rangle = \frac{1}{2^{n/2}} \sum_{x\in 0,1^n} (|x\rangle \otimes |0^n\rangle) \tag{7}$$

Oracle Computation

We call the oracle or black-box (U_f) to compute the function f on the transformed input (obtained after the Hadamard operation).

$$\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (|x\rangle \otimes |0^n\rangle) \xrightarrow{U_f} \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (|x\rangle \otimes |f(x)\rangle) \tag{8}$$

After second Hadamard operation

$$\begin{aligned}\frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} (|x\rangle \otimes |f(x)\rangle) \xrightarrow{H^{\otimes n}\otimes \mathbb{I}} & \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} \big((H^{\otimes n}|x\rangle) \otimes |f(x)\rangle\big) \\ &= \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} \left(\left(\frac{1}{2^{n/2}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle\right) \otimes |f(x)\rangle\right) \\ &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \left(\sum_{y\in\{0,1\}^n} ((-1)^{x\cdot y}|y\rangle \otimes |f(x)\rangle)\right)\end{aligned}$$

We have the following expression as output of the above presented circuit for Simon's algorithm.

$$\frac{1}{2^n} \sum_{x\in\{0,1\}^n} \left(\sum_{y\in\{0,1\}^n} ((-1)^{x\cdot y}|y\rangle \otimes |f(x)\rangle)\right) = \sum_{y\in\{0,1\}^n} \left(|y\rangle \otimes \left(\frac{1}{2^n} \sum_{x\in\{0,1\}^n} ((-1)^{x\cdot y}|f(x)\rangle)\right)\right)$$

Finally, at the end of the circuit, a measurement is performed.

4.5.1 Measurement and Post-processing

After running the above circuit operations we have:

- When $x \oplus y = 0^n$ (i.e. $s = 0^n$), the measurement results in each string $y \in \{0,1\}^n$ with probability $p_y = \frac{1}{2^n}$
- in the cas $x \oplus y = s$ (where $s \neq 0^n$), the probability to obtain each string $y \in \{0,1\}^n$ is given by

$$p_y = \begin{cases} 1/2^{n-1} & \text{if } y\cdot s \text{ is even} \\ 0 & \text{if } y \cdot s \text{ is odd}\end{cases}$$

Thus, in both cases, the measurement results is some string $y \in \{0,1\}^n$ that satisfies $s \cdot y = 0$, and the distribution is uniform over all of the strings that satisfy this constraint and this is enough information to determine s.

We repeat the above process $n-1$ times to get $n-1$ strings $y_1, y_2, \ldots, y_{n-1} \in \{0,1\}^n$, such that

$$\begin{cases} y_1 \cdot s & = 0 \\ y_2 \cdot s & = 0 \\ & \vdots \\ y_{n-1} \cdot s & = 0 \end{cases}$$

and we can use this set of linear equations to find s. We only get a unique non-zero solution s if $y_1, y_2, \ldots, y_{n-1} \in \{0,1\}^n$ are linearly independent (probability $\geq 1/4$).

4.6 Complexity

Simon's algorithm requires $O(n)$ queries to the black box, whereas a classical algorithm would need at least $\Omega(2^{n/2})$ queries. It is also known that Simon's algorithm is optimal in the sense that any quantum algorithm to solve this problem requires $\Omega(n)$ queries.