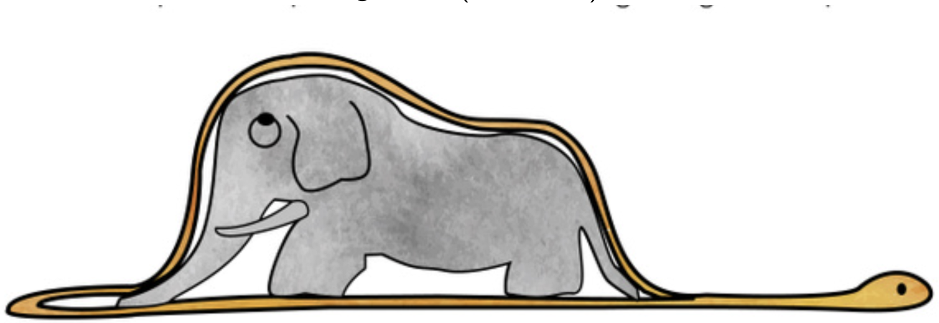
Lecture Topic: Quantum Algorithms I

Input, Compute, Output

Antoine de Saint-Exupéry has provided an excellent illustration of a quantum algorithm (cca. 1943):



Input, Compute, Output

Quantum algorithms ideally have only a modest amount of input. Typically, one uses an initial state with only a single complex probability amplitude, whose magnitude squares is 1. We can assume without loss of regularity that this corresponds to the basis state of all zeros.

Often, people assume that the input is “magically made available” via oracles – but that is often “magical thinking”.

Often, one loads the input via controlled rotations, one scalar at a time.

Input, Compute, Output

Quantum algorithms ideally have only a modest amount of input. Typically, one uses an initial state with only a single complex probability amplitude, whose magnitude squares is 1. We can assume without loss of regularity that this corresponds to the basis state of all zeros.

Often, people assume that the input is “magically made available” via oracles – but that is often “magical thinking”.

Often, one loads the input via controlled rotations, one scalar at a time.

Input, Compute, Output

Quantum algorithms ideally have only a modest amount of input. Typically, one uses an initial state with only a single complex probability amplitude, whose magnitude squares is 1. We can assume without loss of regularity that this corresponds to the basis state of all zeros.

Often, people assume that the input is “magically made available” via oracles – but that is often “magical thinking”.

Often, one loads the input via controlled rotations, one scalar at a time.

Input, Compute, Output

Quantum algorithms then construct a maximally-entangled state on q qubits, whose representation requires a 2^q complex probability amplitudes. (In order for the state to be maximally entangled, the complex probability amplitudes have to be equal.) One can see the large entangled state as a root of a large tree, where in the leaves, one applies Hadamard gates to individual qubits, and in the non-leaf nodes, one applies CNOT. Then, applying a single-qubit gate may change all exponentially-many complex probability amplitudes in parallel, at a unit cost in terms of the depth of the quantum circuit.

Input, Compute, Output

Finally, the output needs to be very simple.

Clearly, the measurement ends up with a basis state, depending on the corresponding complex probability amplitude. In some sense, one may wish the output were only a single complex probability amplitude whose magnitude squares is 1.

This is to be seen from the sample complexity of parameter estimation in multivariate distributions: if we expect a non-trivial superposition on the output, we will need very many copies of the circuit and very many collapsing measurements to estimate the output state.

Often, one employs some classical post-processing.

Input, Compute, Output

Finally, the output needs to be very simple.

Clearly, the measurement ends up with a basis state, depending on the corresponding complex probability amplitude. In some sense, one may wish the output were only a single complex probability amplitude whose magnitude squares is 1.

This is to be seen from the sample complexity of parameter estimation in multivariate distributions: if we expect a non-trivial superposition on the output, we will need very many copies of the circuit and very many collapsing measurements to estimate the output state.

Often, one employs some classical post-processing.

Input, Compute, Output

Finally, the output needs to be very simple.

Clearly, the measurement ends up with a basis state, depending on the corresponding complex probability amplitude. In some sense, one may wish the output were only a single complex probability amplitude whose magnitude squares is 1.

This is to be seen from the sample complexity of parameter estimation in multivariate distributions: if we expect a non-trivial superposition on the output, we will need very many copies of the circuit and very many collapsing measurements to estimate the output state.

Often, one employs some classical post-processing.

Input, Compute, Output

Finally, the output needs to be very simple.

Clearly, the measurement ends up with a basis state, depending on the corresponding complex probability amplitude. In some sense, one may wish the output were only a single complex probability amplitude whose magnitude squares is 1.

This is to be seen from the sample complexity of parameter estimation in multivariate distributions: if we expect a non-trivial superposition on the output, we will need very many copies of the circuit and very many collapsing measurements to estimate the output state.

Often, one employs some classical post-processing.

Functional Problems

In the definition of Aaronson et al.:

FBPP, resp. FQPP is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time randomized, resp. quantum algorithm A such that for all x for which there exists a y with $(x, y) \in R$ and all $\varepsilon > 0$,

$$\mathbb{P}[(x, A(x, 0^{1/\varepsilon})) \in R] > 1 - \varepsilon,$$

where the probability is over A's outputs.

Functional Problems

In the definition of Aaronson et al.:

FBPP, resp. FQPP is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time randomized, resp. quantum algorithm A such that for all x for which there exists a y with $(x,y) \in R$ and all $\varepsilon > 0$,

$$\mathbb{P}[(x, A(x, 0^{1/\varepsilon})) \in R] > 1 - \varepsilon,$$

where the probability is over A's outputs.

Functional Problems with Advice

In the definition of Aaronson et al.:

FP/rpoly, resp. FBQP/qpoly is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time deterministic classical algorithm A, resp. a polynomial-time quantum algorithm Q, a polynomial $p(n,m)$, and an infinite list of advice distributions $\{\mathcal{D}_{n,m}\}_{n,m\geq 1}$, where $\mathcal{D}_{n,m}$ is supported on $\binom{}{p(n,m)}$, resp. advice states $\{|\psi_{n,m}\rangle\}_{n,m\geq 1}$, where $|\psi_{n,m}\rangle$ is on $p(n,m)$ qubits, such that for all x for which there exists a y such that $(x,y) \in R$ and all m,

$$\mathbb{P}_{r\sim\mathcal{D}_{n,m}}[(x, A(x, 0^m, r)) \in R] > 1 - \frac{1}{m},$$

resp.

$$\mathbb{P}[(x, Q(x, 0^m, |\psi_{n,m}\rangle)) \in R] > 1 - \frac{1}{m}.$$

Functional Problems with Advice

In the definition of Aaronson et al.:

FP/rpoly, resp. FBQP/qpoly is the class of polynomially-bounded relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a polynomial-time deterministic classical algorithm A, resp. a polynomial-time quantum algorithm Q, a polynomial $p(n,m)$, and an infinite list of advice distributions $\{\mathcal{D}_{n,m}\}_{n,m\geq 1}$, where $\mathcal{D}_{n,m}$ is supported on $\binom{}{p(n,m)}$, resp. advice states $\{|\psi_{n,m}\rangle\}_{n,m\geq 1}$, where $|\psi_{n,m}\rangle$ is on $p(n,m)$ qubits, such that for all x for which there exists a y such that $(x,y) \in R$ and all m,

$$\mathbb{P}_{r\sim\mathcal{D}_{n,m}}[(x, A(x, 0^m, r)) \in R] > 1 - \frac{1}{m},$$

resp.

$$\mathbb{P}[(x, Q(x, 0^m, |\psi_{n,m}\rangle)) \in R] > 1 - \frac{1}{m}.$$

Quantum Random Oracle Model

If you rely on your circuit calling an oracle, you often typically cannot prove any "usual" separation between complexity classes, e.g., BPP $\neq$ BQP. You can prove only weaker "relativized" results, known as "oracle separations".

The strongest results consider unstructured, random oracles. In the classical/quantum random oracle model of Boneh et al., a random function H is chosen at the beginning, anyone can classically/quantumly access H, i.e., apply a unitary $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus H(x)\rangle$.

Quantum Random Oracle Model

If you rely on your circuit calling an oracle, you often typically cannot prove any "usual" separation between complexity classes, e.g., BPP $\neq$ BQP. You can prove only weaker "relativized" results, known as "oracle separations".

The strongest results consider unstructured, random oracles. In the classical/quantum random oracle model of Boneh et al., a random function H is chosen at the beginning, anyone can classically/quantumly access H, i.e., apply a unitary $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus H(x)\rangle$.

Quantum Random Oracle Model

Bennett et al. show that relative to an oracle chosen uniformly at random with probability 1 an NP-Complete problem cannot be solved on a quantum Turing machine (QTM) in time $o(2^{n/2})$.

Yamakawa and Zhandry show that relative to a random oracle with probability 1, there are NP search problems solvable by BQP machines but not BPP machines.

Quantum Random Oracle Model

Bennett et al. show that relative to an oracle chosen uniformly at random with probability 1 an NP-Complete problem cannot be solved on a quantum Turing machine (QTM) in time $o(2^{n/2})$.

Yamakawa and Zhandry show that relative to a random oracle with probability 1, there are NP search problems solvable by BQP machines but not BPP machines.

Decision Problems

From the point of view of Theoretical Computer Science, the situation in decision problems is somewhat dire:

We do not know any non-relativized separation between P, BPP, and BQP.

(Notice that with non-linear quantum mechanics, we could actually solve NP-Complete and #P-Complete problems.)

Decision Problems

From the point of view of Theoretical Computer Science, the situation in decision problems is somewhat dire:

We do not know any non-relativized separation between P, BPP, and BQP.

(Notice that with non-linear quantum mechanics, we could actually solve NP-Complete and #P-Complete problems.)

Decision Problems

From the point of view of Theoretical Computer Science, the situation in decision problems is somewhat dire:

We do not know any non-relativized separation between P, BPP, and BQP.

(Notice that with non-linear quantum mechanics, we could actually solve NP-Complete and #P-Complete problems.)

Functional Problems

In functional problems, the situation is somewhat better.

In February 2023 Aaronson et al. have shown FP $\neq$ FBPP, unconditionally, and FBQP/qpoly $\neq$ FBQP/poly.

Notice that the FBQP/qpoly refers to a quantum advice, not to an oracle.

Functional Problems

In functional problems, the situation is somewhat better.

In February 2023 Aaronson et al. have shown FP $\neq$ FBPP, unconditionally, and FBQP/qpoly $\neq$ FBQP/poly.

Notice that the FBQP/qpoly refers to a quantum advice, not to an oracle.

Functional Problems

In functional problems, the situation is somewhat better.

In February 2023 Aaronson et al. have shown FP $\neq$ FBPP, unconditionally, and FBQP/qpoly $\neq$ FBQP/poly.

Notice that the FBQP/qpoly refers to a quantum advice, not to an oracle.

Deutsch–Jozsa Problem

In the so-called Deutsch–Jozsa problem, we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}$ that has a rather unusual property: either it is a constant function (there is $y \in \{0,1\}$ such that for all $x \in \{0,1\}^n$, the output is y) or balanced (for precisely 2^{n-1} inputs, the output is 0, and for precisely 2^{n-1} inputs, the output is 1)

The decision version of the problem asks whether the unknown function f is constant. It is clear that classically, one may need to perform $2^{n-1} + 1$ oracle calls in the worst-case, but that there would be excellent randomized algorithms. Notice that for $n = 1$, one asks whether $f(0) + f(1)$ mod 2 is zero.

Deutsch–Jozsa Algorithm

Let us illustrate the algorithm of David Deutsch for $n = 1$:

1. creates an initial, two-register state $|0\rangle\,|1\rangle$
2. apply Hadamard transform to both registers: $\frac{1}{2}\sum_{x=0}^{1}|x\rangle\,(|0\rangle - |1\rangle)$
3. apply the function via the oracle to obtain $\frac{1}{2}\sum_{x=0}^{1}|x\rangle\,(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)$
4. apply the Hadamard transform on the first register again:
$$\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\left[\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x\oplus y}\,|y\rangle\right] = \sum_{y=0}^{1}\left[\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}(-1)^{x\oplus y}\right]|y\rangle$$
5. obtain y by measuring the first register. The probability of measuring $|0\rangle$ is $\left|\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\right|^2$, which evaluates to which evaluates to 1 for constant functions (constructive interference) and to 0 for balanced functions (destructive interference).

We have used 1 query to the oracle. This can be generalized to any n, without increasing the number of queries!

Deutsch–Jozsa Algorithm Explained

If the algorithm seems hard to parse, do not despair. There are a few insights that will help us elucidate its workings:

- the Boolean group
- the oracle
- the Hadamard transform
- amplitude amplification.

We will also introduce the phase kickback, which we will need later.

Artihmetics modulo 2

First, let us consider the artihmetics modulo 2 and its relationship to the XOR operation ($\oplus$, “must have one or the other but not both”).

In $n = 1$ we have seen

$$(0+1) \bmod 2 = 1 \bmod 2 = 1 = (0 \oplus 1) \text{ and}$$

$$(1+1) \bmod 2 = 2 \bmod 2 = 0 = (1 \oplus 1).$$

Beyond $n = 1$ the binary inner product $\odot$ of bitvectors $x, y \in \{0, 1\}^n$ is $x_1 y_1 + \cdots + x_n y_n \bmod 2 = x_1 y_1 \oplus \cdots \oplus x_n y_n$, i.e., essentially counting ones that appear at the corresponding positions in two bitstrings, modulo 2, and thus suggesting whether the count is odd or even.

Artihmetics modulo 2

First, let us consider the artihmetics modulo 2 and its relationship to the XOR operation ($\oplus$, "must have one or the other but not both").

In $n = 1$ we have seen

$$(0 + 1) \bmod 2 = 1 \bmod 2 = 1 = (0 \oplus 1) \text{ and}$$

$$(1 + 1) \bmod 2 = 2 \bmod 2 = 0 = (1 \oplus 1).$$

Beyond $n = 1$ the binary inner product $\odot$ of bitvectors $x, y \in \{0, 1\}^n$ is $x_1 y_1 + \cdots + x_n y_n \bmod 2 = x_1 y_1 \oplus \cdots \oplus x_n y_n$, i.e., essentially counting ones that appear at the corresponding positions in two bitstrings, modulo 2, and thus suggesting whether the count is odd or even.

Artihmetics modulo 2

First, let us consider the artihmetics modulo 2 and its relationship to the XOR operation ($\oplus$, "must have one or the other but not both").

In $n = 1$ we have seen

$$(0+1) \bmod 2 = 1 \bmod 2 = 1 = (0 \oplus 1) \text{ and}$$

$$(1+1) \bmod 2 = 2 \bmod 2 = 0 = (1 \oplus 1).$$

Beyond $n = 1$ the binary inner product $\odot$ of bitvectors $x, y \in \{0,1\}^n$ is $x_1y_1 + \cdots + x_ny_n \bmod 2 = x_1y_1 \oplus \cdots \oplus x_ny_n$, i.e., essentially counting ones that appear at the corresponding positions in two bitstrings, modulo 2, and thus suggesting whether the count is odd or even.

The Oracle

Next, let us consider the oracle. One assumes that for a function f, there is an oracle U_f that maps $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, where $\oplus$ denotes the XOR operation (or addition modulo 2). This can be simplified to $U_f |x\rangle = (-1)^{f(x)} |x\rangle$.

Clearly, $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$.

Either way, this is a reversible operation and can be implemented in a unitary.

The Oracle

Next, let us consider the oracle. One assumes that for a function f, there is an oracle U_f that maps $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, where $\oplus$ denotes the XOR operation (or addition modulo 2). This can be simplified to $U_f |x\rangle = (-1)^{f(x)} |x\rangle$.

Clearly, $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$.

Either way, this is a reversible operation and can be implemented in a unitary.

The Oracle

Next, let us consider the oracle. One assumes that for a function f, there is an oracle U_f that maps $|x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$, where $\oplus$ denotes the XOR operation (or addition modulo 2). This can be simplified to $U_f |x\rangle = (-1)^{f(x)} |x\rangle$.

Clearly, $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$.

Either way, this is a reversible operation and can be implemented in a unitary.

The Oracle

In the case of Boolean function f on $n = 1$ bits, one can think of this as a CNOT gate controlled by the value of $f(x)$. In the top-left corner, you have I (the identity) for $f(0) = 0$ and σ_x (flip) for $f(0) = 1$. Similarly in the bottom-right corner, you have I (the identity) for $f(1) = 0$ and σ_x (flip) for $f(1) = 1$.

$$U_f = \begin{pmatrix} 1 - f(0) & f(0) & 0 & 0 \\ f(0) & 1 - f(0) & 0 & 0 \\ 0 & 0 & 1 - f(1) & f(1) \\ 0 & 0 & f(1) & 1 - f(1) \end{pmatrix}$$

Amplitude Amplification

In a way, the Deutsch–Jozsa algorithm also demonstrates the significance of allowing quantum amplitudes to take both positive and negative values.

In the Qiskit Textbook and many other sources, this is illustrated starting with an interference experiment (cf. Young's double-slit interferometer, 1803): a particle can travel from the source to an array of detectors through two slits. Each detector has a probability of observing a particle that depends on the phases of the incoming waves. Same phases increase the probability (constructive interference); very different phases reduce the probability (destructive interference).

One can consider 2^n possible paths x and 2^n possible detectors y, both labelled by bitstrings. The phase accumulated at detector x along a path y equals $C(-1)^{f(x)+x\cdot y}$, where $x \cdot y$ is the binary inner product and C is a normalizing constant.

The probability of a particle at detector y is $\mathbb{P}(y) = |C \sum_x (-1)^{f(x)+x\cdot y}|^2$ with $C = 2^{-n}$.

Amplitude Amplification

In a way, the Deutsch–Jozsa algorithm also demonstrates the significance of allowing quantum amplitudes to take both positive and negative values.

In the Qiskit Textbook and many other sources, this is illustrated starting with an interference experiment (cf. Young's double-slit interferometer, 1803): a particle can travel from the source to an array of detectors through two slits. Each detector has a probability of observing a particle that depends on the phases of the incoming waves. Same phases increase the probability (constructive interference); very different phases reduce the probability (destructive interference).

One can consider 2^n possible paths x and 2^n possible detectors y, both labelled by bitstrings. The phase accumulated at detector x along a path y equals $C(-1)^{f(x)+x\cdot y}$, where $x \cdot y$ is the binary inner product and C is a normalizing constant.

The probability of a particle at detector y is $\mathbb{P}(y) = |C \sum_x (-1)^{f(x)+x\cdot y}|^2$ with $C = 2^{-n}$.

Amplitude Amplification

In a way, the Deutsch–Jozsa algorithm also demonstrates the significance of allowing quantum amplitudes to take both positive and negative values.

In the Qiskit Textbook and many other sources, this is illustrated starting with an interference experiment (cf. Young's double-slit interferometer, 1803): a particle can travel from the source to an array of detectors through two slits. Each detector has a probability of observing a particle that depends on the phases of the incoming waves. Same phases increase the probability (constructive interference); very different phases reduce the probability (destructive interference).

One can consider 2^n possible paths x and 2^n possible detectors y, both labelled by bitstrings. The phase accumulated at detector x along a path y equals $C(-1)^{f(x)+x\cdot y}$, where $x \cdot y$ is the binary inner product and C is a normalizing constant.

The probability of a particle at detector y is $\mathbb{P}(y) = |C \sum_x (-1)^{f(x)+x\cdot y}|^2$ with $C = 2^{-n}$.

Amplitude Amplification

In a way, the Deutsch–Jozsa algorithm also demonstrates the significance of allowing quantum amplitudes to take both positive and negative values.

In the Qiskit Textbook and many other sources, this is illustrated starting with an interference experiment (cf. Young's double-slit interferometer, 1803): a particle can travel from the source to an array of detectors through two slits. Each detector has a probability of observing a particle that depends on the phases of the incoming waves. Same phases increase the probability (constructive interference); very different phases reduce the probability (destructive interference).

One can consider 2^n possible paths x and 2^n possible detectors y, both labelled by bitstrings. The phase accumulated at detector x along a path y equals $C(-1)^{f(x)+x\cdot y}$, where $x \cdot y$ is the binary inner product and C is a normalizing constant.

The probability of a particle at detector y is $\mathbb{P}(y) = |C \sum_x (-1)^{f(x)+x\cdot y}|^2$ with $C = 2^{-n}$.

Amplitude Amplification

Now let us consider the probability of observing an all-zero string y, which is

$$|2^{-n}\sum_x (-1)^{f(x)+x\cdot y}|^2 = |2^{-n}\sum_x (-1)^{f(x)+0}|^2,$$

in the two cases of the promise problem:

- if the $f(x) = c$, then the probability is $|2^{-n}\sum_x (-1)^c|^2 = 1$
- if the $f(x)$ is balanced, then the probability is zero $|2^{-n}\sum_x (-1)^{f(x)}|^2 = 0$, because the alternating sign will lead to a cancellation of the terms.

Exercise 1

Exercise

Plot a diagram of the double-slit experiment and the 2^n detectors and the inference pattern for some $n \geq 8$.

The Hadamard Transform

We have seen the Hadamard gate:

$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle =: |+\rangle \tag{5.1}$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle =: |-\rangle \tag{5.2}$$

$$H(|+\rangle) = H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) + \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |0\rangle \tag{5.3}$$

$$H(|-\rangle) = H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\Big(|0\rangle + |1\rangle\Big) - \frac{1}{2}\Big(|0\rangle - |1\rangle\Big) = |1\rangle \tag{5.4}$$

without really understanding it.

The Hadamard Transform

First, notice that for an n-qubit state $|k\rangle$, the application of Hadamards qubit-wise yields:

$$H^{\otimes n}|k\rangle = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}(-1)^{k\odot j}|j\rangle\,,$$

where $j \odot k = j_1k_1 \oplus j_2k_2 \oplus \cdots \oplus j_nk_n$ and $\oplus$ is XOR as above.

Second, this is rooted in a non-trivial fact that the Hadamard transform is the Fourier transform on the Boolean group $(\mathbb{Z}/2\mathbb{Z})^n$.

The Hadamard Transform

First, notice that for an n-qubit state $|k\rangle$, the application of Hadamards qubit-wise yields:

$$H^{\otimes n}|k\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{k \odot j} |j\rangle ,$$

where $j \odot k = j_1 k_1 \oplus j_2 k_2 \oplus \cdots \oplus j_n k_n$ and $\oplus$ is XOR as above.

Second, this is rooted in a non-trivial fact that the Hadamard transform is the Fourier transform on the Boolean group $(\mathbb{Z}/2\mathbb{Z})^n$.

The Hadamard Transform in Deutsch–Jozsa

In the example of the second application of Hadamard in Deutsch–Jozsa for $n = 1$, we obtain:

$$\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}\left[\frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x\oplus y}\left|y\right\rangle\right] = \sum_{y=0}^{1}\left[\frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}(-1)^{x\oplus y}\right]\left|y\right\rangle.$$

More broadly, the Hadamard maps $|x\rangle$ to $2^{-n/2}\sum_y(-1)^{x\odot y}|y\rangle$. For our state $2^{-n/2}\sum_x(-1)^{f(x)}|x\rangle$, this will amount to $2^{-n}\sum_x(-1)^{f(x)+x\odot y}|y\rangle$, just as in the interference experiment.

Exercise 2

Exercise

Write down the Hadamard gate on $n = 3$.

Phase Kickback

In the previous chapter, we have seen the phase factor ("global phase", "global gauge"), whereby quantum states $|\psi\rangle$ and $e^{i\alpha}|\psi\rangle$ are indistinguishable by measurement with any linear operator ϕ in the sense of $|\langle\phi\rangle\,\psi|^2 = |e^{i\alpha}\langle\phi\rangle\,\psi|^2 = |\langle\phi\rangle\,\psi|^2$.

(A phase-shift gate $P(\alpha) = e^{i\alpha}I$ multiplies any state by a global phase α.)

If we apply a control-U gate to $|\psi\rangle$, where $|\psi\rangle$ is an eigenstate of U, then

- $|\psi\rangle$ is unchanged
- the phase $|0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1|$ is transferred to the input state $|\psi\rangle$ of the control qubit.

Phase Kickback

In the previous chapter, we have seen the phase factor ("global phase", "global gauge"), whereby quantum states $|\psi\rangle$ and $e^{i\alpha}|\psi\rangle$ are indistinguishable by measurement with any linear operator ϕ in the sense of $|\langle\phi\rangle\,\psi|^2 = |e^{i\alpha}\langle\phi\rangle\,\psi|^2 = |\langle\phi\rangle\,\psi|^2$.

(A phase-shift gate $P(\alpha) = e^{i\alpha}I$ multiplies any state by a global phase α.)

If we apply a control-U gate to $|\psi\rangle$, where $|\psi\rangle$ is an eigenstate of U, then

- $|\psi\rangle$ is unchanged
- the phase $|0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1|$ is transferred to the input state $|\psi\rangle$ of the control qubit.

Phase Kickback

In the previous chapter, we have seen the phase factor ("global phase", "global gauge"), whereby quantum states $|\psi\rangle$ and $e^{i\alpha}|\psi\rangle$ are indistinguishable by measurement with any linear operator ϕ in the sense of $|\langle\phi\rangle\,\psi|^2 = |e^{i\alpha}\langle\phi\rangle\,\psi|^2 = |\langle\phi\rangle\,\psi|^2$.

(A phase-shift gate $P(\alpha) = e^{i\alpha}I$ multiplies any state by a global phase α.)

If we apply a control-U gate to $|\psi\rangle$, where $|\psi\rangle$ is an eigenstate of U, then

- $|\psi\rangle$ is unchanged
- the phase $|0\rangle\langle 0| + e^{i\alpha}|1\rangle\langle 1|$ is transferred to the input state $|\psi\rangle$ of the control qubit.

Phase Kickback

Let us see the *phase kickback* in action. Let us consider a single-qubit gate U and its eigenstate $|\psi\rangle$:

$$U|\psi\rangle = e^{i\phi}|\psi\rangle\,.$$

We wish to estimate ϕ up to the period (2π). This is possible with an extra ("ancilla") qubit, one Hadamard gate on the ancilla qubit, and a CNOT gate. Notably, after applying the CNOT controlled by the ancilla qubit, the phase of the ancilla will be ϕ:

$$XH \otimes I\,|0\rangle|\psi\rangle = \frac{|0\rangle + e^{i\phi}|1\rangle}{\sqrt{2}}|\psi\rangle\,,$$

where we have used the fact that CNOT $= [I0;0X]$.

Phase Kickback

In the two-qubit Deutsch algorithm above, the first qubit acts as an ancilla qubit, and the controlled qubit is in the eigenstate of the NOT gate with eigenvalue -1. $\phi = \pi$. Thus, we get $\frac{1}{\sqrt{2}} |0\rangle + (-1)^{f(0) \oplus f(1)} |1\rangle$. The phase kick-back is either 0 or π, which can be distinguished by measuring σ_x, or by applying Hadamard gate again and then measuring in the computational basis.

This will be important in the following discussion of Simon's algorithm.

Phase Kickback

In the two-qubit Deutsch algorithm above, the first qubit acts as an ancilla qubit, and the controlled qubit is in the eigenstate of the NOT gate with eigenvalue -1. $\phi = \pi$. Thus, we get $\frac{1}{\sqrt{2}} |0\rangle + (-1)^{f(0) \oplus f(1)} |1\rangle$. The phase kick-back is either 0 or π, which can be distinguished by measuring σ_x, or by applying Hadamard gate again and then measuring in the computational basis.

This will be important in the following discussion of Simon's algorithm.

First Oracle Sepration

Let us illustrate the first, historically, and most commonly taught "oracle separation" between BPP and BQP on Simon's problem. This is not a problem, which would be useful on its own, more akin a "guessing game". It uses a highly structured, "periodic" oracle. We will see, however, that the crucial concept of "period finding" underlies the famous Shor factoring algorithm. (Daniel Simon actually recalls[1] that Shor developed his factoring algorithm having seen a preprint of his.)

[1] https://aws.amazon.com/blogs/quantum-computing/simons-algorithm/

Simon's Problem

In the so-called Simon's problem, we have

- a dimension n
- a black-box function $f(x) : \{0,1\}^n \to \{0,1\}^n$ that has a rather unusual property: for all $x, y \in \{0,1\}^n$, $f(x) = f(y)$ if any only if $x \oplus y \in \{0^n, s\}$ for some unknown secret $s \in \{0,1\}^n$,

where $\oplus$ denotes the elementwise XOR operation. Notice that $x \oplus y = 0^n$, if and only if $a = b$. Thus,

- either the secret is $s = 0^n$ and the function is a bijection ("one-to-one", invertible)
- or the secret is $s \neq 0^n$ and the function is not a bijection, but rather "two-to-one".

The decision version of the problem asks whether the unknown function f is a bijection (and thus whether $s = 0^n$).

Exercise 3

Exercise

To get a feel for this, pick an $f(x) : \{0,1\}^3 \rightarrow \{0,1\}^3$. Ask your neighbour to guess whether it's a bijection. How many queries did he need?

Simon's Problem

Notice that there is *no input*. Hence, it is impossible to reason about the description complexity of the input.

Let us measure the complexity of (a classical or quantum algorithm) by the number of evaluations of f at distinct values (x) it requires. (This is also known as the number of oracle queries or oracle complexity.)

In a deterministic Turing machine, one may try 2^n inputs one by one until one obtains two inputs producing the same output, or decides that no two match.

In a probabilistic machine, one may try to sample the 2^n inputs randomly, and as long as no two match, suggest that the function is a bijection, with an ever higher probability.

Simon's Problem

Notice that there is *no input*. Hence, it is impossible to reason about the description complexity of the input.

Let us measure the complexity of (a classical or quantum algorithm) by the number of evaluations of f at distinct values (x) it requires. (This is also known as the number of oracle queries or oracle complexity.)

In a deterministic Turing machine, one may try 2^n inputs one by one until one obtains two inputs producing the same output, or decides that no two match.

In a probabilistic machine, one may try to sample the 2^n inputs randomly, and as long as no two match, suggest that the function is a bijection, with an ever higher probability.

Simon's Problem

Notice that there is *no input*. Hence, it is impossible to reason about the description complexity of the input.

Let us measure the complexity of (a classical or quantum algorithm) by the number of evaluations of f at distinct values (x) it requires. (This is also known as the number of oracle queries or oracle complexity.)

In a deterministic Turing machine, one may try 2^n inputs one by one until one obtains two inputs producing the same output, or decides that no two match.

In a probabilistic machine, one may try to sample the 2^n inputs randomly, and as long as no two match, suggest that the function is a bijection, with an ever higher probability.

Simon's Problem

Notice that there is *no input*. Hence, it is impossible to reason about the description complexity of the input.

Let us measure the complexity of (a classical or quantum algorithm) by the number of evaluations of f at distinct values (x) it requires. (This is also known as the number of oracle queries or oracle complexity.)

In a deterministic Turing machine, one may try 2^n inputs one by one until one obtains two inputs producing the same output, or decides that no two match.

In a probabilistic machine, one may try to sample the 2^n inputs randomly, and as long as no two match, suggest that the function is a bijection, with an ever higher probability.

Simon's Algorithm

The quantum algorithm for solving the problem is similar to the randomized algorithm. Repeatedly, for each sample, we perform the following steps:

1. creates an initial, two-register state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$
2. apply Hadamard transform on the first register: $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|0\rangle^{\otimes n}$
3. apply the function via the oracle to obtain $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|f(k)\rangle$
4. apply the Hadamard transform on the first register again:

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}\left[\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}(-1)^{j\odot k}|j\rangle\right]|f(k)\rangle = \sum_{j=0}^{2^n-1}|j\rangle\left[\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{j\odot k}|f(k)\rangle\right]$$

5. obtain y by measuring the first register. The probability of measuring $|j\rangle$ is $\left|\frac{1}{2^n}\sum_{k=0}^{2^n-1}(-1)^{j\odot k}|f(k)\rangle\right|^2$.

Then, we classically solve a system of equations given by the samples to obtain s. (Each sample satisfies $ys = 0$.) If $s = 0^n$, return YES.

Shor's Factoring

Shor's factoring algorihm uses a non-trivial initial preprocessing, but then we perform the following steps:

1. creates an initial, Q-qubit state $|0\rangle^{\otimes Q}$
2. apply Hadamard transform on it: $\frac{1}{\sqrt{Q}}\sum_{k=0}^{Q-1}|x\rangle$
3. apply the function $f(x) = a^x \bmod N$ using $U_f|x, 0^n\rangle = |x, f(x)\rangle$ to obtain
$$U_f \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x, 0^n\rangle = \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x, f(x)\rangle$$
such that the value we are looking for is in the phase of
4. apply the quantum Fourier transform: $\frac{1}{Q}\sum_{x=0}^{Q-1}\sum_{y=0}^{Q-1}\omega^{xy}|y, f(x)\rangle$
5. obtain y by measuring the first register. The probability of measuring $|y, z\rangle$ is
$$\frac{1}{Q^2}\frac{\sin^2(\frac{\pi m r y}{Q})}{\sin^2(\frac{\pi r y}{Q})}$$
.

Then, we apply classical post-processing.

Repetitive?

If you felt that there is common pattern across these algorithms, you are right. The quantum Fourier transform is, in some sense, just a more efficient way of measuring the phase.

There are, actually, very few “paradigms” in the design of quantum algorithms, and some of the steps (equal superposition, some operation thereupon) are necessary.

One may consider, for example , amplitude amplification (algorithms above and Grover) with or without quantum Fourier transform, Harrow-Hassidim-Lloyd (HHL), and quantum signal processing (QSP).

Repetitive?

If you felt that there is common pattern across these algorithms, you are right. The quantum Fourier transform is, in some sense, just a more efficient way of measuring the phase.

There are, actually, very few “paradigms” in the design of quantum algorithms, and some of the steps (equal superposition, some operation thereupon) are necessary.

One may consider, for example , amplitude amplification (algorithms above and Grover) with or without quantum Fourier transform, Harrow-Hassidim-Lloyd (HHL), and quantum signal processing (QSP).

Repetitive?

If you felt that there is common pattern across these algorithms, you are right. The quantum Fourier transform is, in some sense, just a more efficient way of measuring the phase.

There are, actually, very few “paradigms” in the design of quantum algorithms, and some of the steps (equal superposition, some operation thereupon) are necessary.

One may consider, for example , amplitude amplification (algorithms above and Grover) with or without quantum Fourier transform, Harrow-Hassidim-Lloyd (HHL), and quantum signal processing (QSP).