Workshop seminar: Quantum Computing II

Jonathan Kley, 17/01/2023

References:
- de Wolf's lecture notes (Chapt. 2,3)
- Preskill's lecture notes (Chapt. 6)

Outline:
- Recap and techniqualities
- Deutsch-Jozsa algorithm
- Bernstein-Vazirani algorithm
- Simon's algorithm

Today, we want to find some specific scenarios where quantum computing is superior to classical computing which mostly means that less operations have to be applied to get to the final result.

We will have a look at several early (more or less useful) algorithms eventually building up to Shor's algorithm (next week) which has serious applications in cryptography. In fact, Shor came up with his factoring algorithm by generalising Simon's algorithm which we will study today.

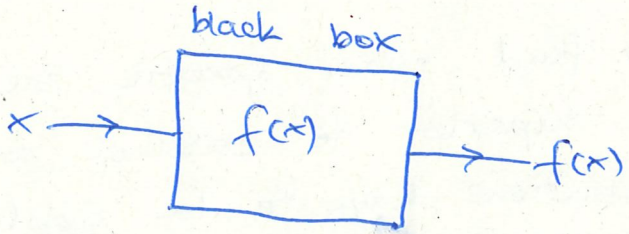
We will start with a recap from last time to remind everyone of the notation previously introduced and to introduce new notation which will simplify our analysis.

Last time: Deutsch problem

Given a function $f: \{0,1\} \longrightarrow \{0,1\}$, $x \longmapsto f(x)$, can we determine if $f(0) = 0$ or $f(0) = 1$ with only 1 evaluation of the function?

Answer: Yes, with a quantum computer as we saw last time.

We will study similar problems today, generalising the Deutsch problem, where we have a function of which we only know some generic properties and we have to find out what it does as efficiently as possible. This kind of problem is often reffered to as a "black box" or "oracle".



Reminder: Hadamard gate

Last time we were introduced to the Hadamard gate H, which we will continuously use today. It is defined as follows

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

in matrix rep.: $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

We can generalize this to an n-bit state. First, for an n-bit null state

$$H^{n\otimes}|0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} |j\rangle$$

and for a generic state: $i \cdot j = \sum_{k=1}^{n} i_k j_k$

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

One can verify that this is compatible with the 1-bit example.

Let us now see how we can implement the "black box" function into our quantum algorithms. Consider an N-bit piece of information $x = (x_0, \dots, x_{N-1}) \in \{0,1\}^N$ with $N = 2^n$ such that each bit can be accessed by an n-bit address i. This could for example be a piece of information saved in an N-bit memory.
We can access the memory via a "black box" which returns the bit x_i given the address i. In this way we can also implement our black box functions.
As a quantum operation:

$$O_x : |i, 0\rangle \longrightarrow |i, x_i\rangle$$

"query"

address bit, target bit

A query returns bit x_i for given address bit $|i\rangle$ at target bit. We want O_x to be unitary, so we have to specify $O_x |i, 1\rangle$. Therefore, we define in general

$$O_x : |i, b\rangle \longrightarrow |i, b \oplus x_i\rangle$$

which is unitary. Here, $a \oplus b = a + b \bmod 2$

For the analysis of the different oracles, it turns out to be useful to define another type of query, the so-called "phase-query"

$$|i\rangle \longrightarrow (-1)^{x_i} |i\rangle$$

which can be done by using the Hadamard state $|-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ as a target bit. Then,

$$O_x |i\rangle|-\rangle = |i\rangle \frac{1}{\sqrt{2}} \left(|x_i\rangle - |1-x_i\rangle \right) =$$

$$= \begin{cases} |i\rangle \frac{1}{\sqrt{2}} \left(|0\rangle - |1-0\rangle \right) & x_i = 0 \\ |i\rangle \frac{1}{\sqrt{2}} \left(|1\rangle - |1-1\rangle \right) & x_i = 1 \end{cases}$$

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

$\Longrightarrow$ $x_i = 0$, nothing happens

$x_i = 1$, sign flipped

This can be useful to gain information about black box. We will denote the phase-query as $O_{x,\pm}$. One can check that: $O_x |i\rangle|+\rangle = |i\rangle|+\rangle$ irrespective of x.

We are now ready to tackle the first problem, the Deutsch-Jozsa problem.

Deutsch-Jozsa algorithm

Assume we have the following black-box function, where the black box contains $N = 2^n$ bits of information $x \in \{0,1\}^N$, such that either:

1) all x_i have same value (0 or 1) "constant"

2) $\frac{N}{2}$ of x_i are 0 and $\frac{N}{2}$ of x_i are 1 "balanced"

Goal: Find out if function is constant or balanced with least amount of steps.

We can do this by using clever combinations of the Hadamard gate and function queries. We start with the initial state $|0^n\rangle$ and apply a Hadamard transform (we ignore the target bit here as it doesn't change, c.f. phase query)

$$H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle$$

The $O_{x,\pm}$-query turns this into

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}} (-1)^{x_i} |i\rangle$$

After the query we apply another Hadamard gate giving the final superposition

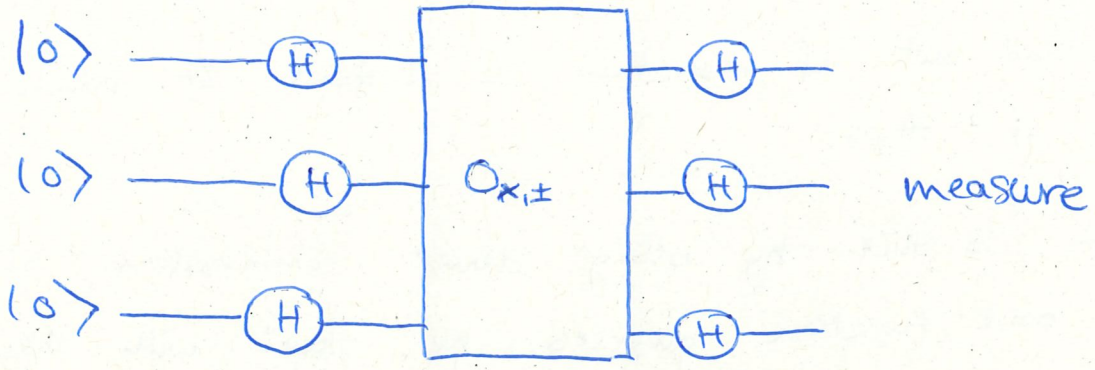
$$\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{x_i} \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle$$

to which we apply a measurement in the initial state. We know that $i \cdot 0^n = 0 \quad \forall\, i \in \{0,1\}^n$

$\Rightarrow$ only $\frac{1}{2^n} \sum_{i \in \{0,1\}^n} (-1)^{x_i}$ remains which can easily be evaluated for all cases.

$$\frac{1}{2^n}\sum_{i\in\{0,1\}^n}(-1)^{x_i} = \begin{cases} 1 & \text{if } x_i = 0 \ \forall i \\ -1 & \text{if } x_i = 1 \ \forall i \\ 0 & \text{if } x \text{ is balanced} \end{cases}$$

$\Rightarrow$ Final state has probability of 1 to be $|0^n\rangle$ if x is constant and probability of 0 if x is balanced.



Therefore we can solve the Deutsch-Jozsa problem with 1 query and $O(n)$ other operations with certainty.

Classically, we solve the problem as follows:
We query one bit after another. Then, if x is balanced we can get the correct answer as early as with the 2nd query. However to know for sure in general that the function is constant we need $\frac{N}{2}+1$ queries. Therefore, counting queries the quantum algorithm is superior.

Caveats: • We only compare the number of function queries. Since the problem is so simple it's not obvious if the quantum algorithm is still superior considering a real experiment with the full quantum circuit including the $O(n)$ other operations.

- We only considered classical approaches without allowing for errors. If we query the output at 2 random positions instead, we get the correct answer with probability 1 if x is constant and with probability $\frac{1}{2}$ if x is balanced. Therefore, allowing for small errors we can do much better and the quantum-classical separation only holds for deterministic classical algorithms.

Bernstein-Vazirani algorithm

Problem: $N = 2^n$, given $x \in \{0,1\}^N$ with the property that there is some unknown $a \in \{0,1\}^n$, s.t. $x_i = i \cdot a \mod 2 = i \oplus a$.

Goal: Find a.

We repeat all steps of the Deutsch-Jozsa algorithm until after the query

$$O_{x,\pm} H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{x_i} |i\rangle \quad (=)$$

We know: $(-1)^{x_i} = (-1)^{i \cdot a \mod 2} = (-1)^{i \cdot a}$

$$(=) \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{i \cdot a} |i\rangle$$

Furthermore $H^2 = \mathbb{1} \Rightarrow H$ is its own inverse

$$\Rightarrow H^{\otimes n} O_{x,\pm} H^{\otimes n} |0^n\rangle = |a\rangle$$

and we have miraculously found a!

As for the Deutsch-Jozsa algorithm we can find the solution with only 1 query and $\mathcal{O}(n)$ other operations. However, for a classical computer this problem is much harder. We can see this easily from a information-theoretic point of view: $a \in \{0,1\}^n$ is an n-bit string. Since a classical query gives at most 1 bit of information, we need at least n queries to get the final answer. This doesn't change if we use randomization with small error probabilities.

Therefore, here the quantum-classical separation is stronger. Again, we only compared the number of oracle queries. Implementing the full realistic quantum algorithm in an experiment could make the quantum-classical separation less severe.

Simon's problem

The previous examples could be solved with less queries by a quantum computer but were possible to solve reasonably well classically. For Simon's problem we will see that there will be a polynomial vs exponential separation between quantum and classical.

Problem: $N = 2^n$, given $x = (x_0, \ldots, x_{N-1})$ where $x_i \in \{0,1\}^n$ with the property that there is some unknown non-zero $s \in \{0,1\}^n$, s.t. $x_i = x_j$ iff $i = j \oplus s \equiv j \cdot s \bmod 2$

Goal: Find s.

Note that the blackbox function as a function mapping $\{0,1\} \to \{0,1\}$ is 2-to-1 (i.e. $\forall y \in \text{im}f : \exists x_1, x_2 \in \text{dom}f$ s.t.: $f(x_1) = y = f(x_2)$).

Furthermore, each x_i is now an n-bit string!

Similarly to previous problems we start in $|0^n\rangle|0^n\rangle$ state where target bit is now also element of $\{0,1\}^n$. Applying a Hadamard transform to the address bit gives

$$\left(H^{\otimes n}|0^n\rangle\right)|0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle|0^n\rangle$$

Now, we apply a query O_x to get the output of the fct. in the target bit

$$O_x\left(\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle|0^n\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle|x_i\rangle$$

The following step is not necessary but it will simplify the discussion significantly. Measuring the second state in the computational basis (i.e. using the projector $|x_j\rangle\langle x_j|$) gives

$$\frac{1}{\sqrt{2}}\left(|i\rangle + |i \oplus s\rangle\right)|x_i\rangle$$

which in the address state is a superposition of all addresses that give the output $|x_i\rangle$.

To differentiate the addresses in a measurement we apply another Hadamard gate to the address qubit

$$\frac{1}{\sqrt{2^{n+1}}}\left(\sum_{j \in \{0,1\}^n} (-1)^{i \cdot j}|j\rangle + \sum_{j \in \{0,1\}^n} (-1)^{(i \oplus s) \cdot j}|j\rangle\right) \quad (=)$$

We can now use the distributivity of $(\cdot, \oplus)$ to simplify: $(i \oplus s) \cdot j = (i \cdot j) \oplus (s \cdot j)$.

$$= \frac{1}{\sqrt{2^{n+1}}} \left(\sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle + \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} (-1)^{j \cdot s} |j\rangle \right) =$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} \underbrace{\left(1 + (-1)^{s \cdot j}\right)}_{\equiv (*)} |j\rangle$$

Now, $|j\rangle$ only has a non-zero amplitude if $(*) \neq 0$, i.e. if $s \cdot j = 0 \bmod 2$. Performing a measurement gives a random element from the set $\{j \mid s \cdot j = 0 \bmod 2\}$.

To find $s = (s_0, \dots, s_{n-1}) \in \{0,1\}^n$ we have to find n independent equations for all s_i:

$$j_1 \cdot s = 0$$
$$j_2 \cdot s = 0$$
$$\vdots$$

$\Rightarrow$ We repeat the algorithm until we have n independent equations. The solutions to these equations will be $|0^n\rangle$ and s.

They can be found by Gaussian elimination mod 2 by a classical computer of size $O(n^3)$.

One can show that the quantum algorithm needs $O(n)$ queries.

Classical solution to Simon's problem

Classically this problem is hard. The oracle has to be queried an exponentially large (as compared to $O(n)$ times for the quantum algorithm) amount of times to have a good probability of finding the shift s.

We will only give an upper bound on the number of queries.
Strategy: we randomly call the function until we find 2 queries with address i and j for $x_i = x_j$, s.t. $i = j \oplus s$.

The algorithm is based on the "birthday paradox" which is the phenomenon that in a group of only 23 people it is already very likely that 2 people share the same birthday even though there are 365 possible birthdays.

Intuitive explanation: The number of pairs of people goes quadratic with the number of people and each pair has a $\frac{1}{365}$ probability to have the same birthday.

Let's apply the algorithm to our problem:

We make T randomly chosen, distinct queries $\{i_1, \dots, i_T\}$. We will see now how large T has to be to get the same output for 2 different indices. We check all pairs (i_k, i_l) in the set $\{i_1, \dots, i_T\}$ until we find collision of queries:

$$x_{i_k} = x_{i_l} \qquad \text{for } k \neq l$$

Then $i_k = i_l \oplus s \implies s = i_k \oplus i_l$

For trivial solution $s = 0^n$: no collision because we demand $k \neq l$ for $x_{i_k} = x_{i_l}$.

There are $\binom{T}{2} = \frac{1}{2} T(T-1) \simeq \frac{T^2}{2}$ pairs that can be constructed from $\{i_1, \ldots, i_T\}$.

Because the indices are chosen randomly the probability for a collision is $\frac{1}{2^n - 1}$

$\Longrightarrow$ The expectation value for the number of collisions $\langle n_c \rangle$ is

$$\langle n_c \rangle = \frac{T^2}{2} \frac{1}{2^n - 1} \simeq \frac{T^2}{2^{n+1}}$$

$\Longrightarrow$ For $\langle n_c \rangle \overset{!}{=} 1$ we get

$$T \sim \sqrt{2^{n+1}}$$

(upper bound)

We expect an exponential scaling with n to find s.

Note: $\langle n_c \rangle = 1$ doesn't mean that it is likely to get one collision but more involved calculation shows that we have the same scaling also with higher probability.

Can also calculate lower bound which goes to $\sqrt{2^n}$ in large-n limit.

$\Longrightarrow$ Have shown that classical algorithm has exponential scaling & and there is a polynomial vs. exponential separation between classical and quantum.

For details on calculation of lower bound (involved calculation) see e.g. lecture notes by de Wolf.