Quantum Computing Notes
Prof. Hamzeh Roumani
EECS, York University, Toronto, Ont., Canada

THE SIMON ALGORITHM

1. The Problem

Consider a function *f:* $\{0,1\}^n$ --> $\{0,1\}^n$. Its domain and co-domain consist of 2^n elements each. In programming context, *f* takes n boolean parameters and returns an array of n booleans. And if you treat the n 0/1 values as bits in the binary representation of integers, then *f* can be thought of as a function that maps an integer in [0,N-1] to an integer in [0,N-1], where $N=2^n$. We assume that *f* is provided as a black-box U_f (an *oracle*) that implements it in hardware.

Given that *f* satisfies the property (the *promise*):

$$\exists s \in \{0,1\}^n : \forall x, y \in \{0,1\}^n, f(x) = f(y) \Leftrightarrow x = y \oplus s$$

find the bit string *s*. In other words, *f* is either 2-to-1 (maps pairs connected via the mask s to the same image) or 1-to-1 (maps distinct elements to distinct images). The 1-to-1 case, which corresponds to s being a string of 0s, is trivial, and we will sidestep by adding $s \neq 0^n$ to the promise. Hence, we will assume that *f* is 2-to-1. As before, we assume that *f* is given through a black box U_f (an *oracle*) that implements it.

2. Examples

n=1: f(0)=0, f(1)=0 → s = 1

n=2: f(0,0)=00, f(0,1)=01, f(1,0)=01, f(1,1)=00 → s = 11

n=3: f(0,0,0)=010, f(0,0,1)=111, f(0,1,0)=100, f(0,1,1)=011, f(1,0,0)=100, f(1,0,1)=011, f(1,1,0)=010, f(1,1,1)=111 → s = 110

3. A Classical Algorithm

All we need to do is find two distinct elements in the domain, say *a* and *b*, that map to the same value, i.e. *f(a)=f(b)*. Once we find such a pair, we can readily compute s using:

$$a = b \oplus s \Rightarrow s = a \oplus b$$

Hence solving the problem amounts to finding any one pair of elements with the same image. There should be many such pairs in the domain thanks to the 2-to-1 precondition. Specifically, there are 2^n elements in the domain so if we compute *f* at half of them and found all their images distinct, which is the worst case, then computing *f* at one more element will surely lead to a *collision* (a value equal to one of the previously computed values). Hence, the number of

times we need to evaluate *f* (the oracle query count) grows exponentially with n, the number of variables in *f*. This problem is thus intractable, i.e. no efficient, deterministic (classical) algorithm exists for solving it.

- For n=3, give an example of a 2-to-1 function (different from the above) and determine the mask s.
- Show that the cardinality of the range of *f* is 2^{n-1}.
  The range of a function is the subset of its co-domain that contains all the image. It is equal to the co-domain if f is surjective.
- Show that $2^n / 2 + 1$ queries are needed in the worst case in the above classical algorithm.
- Argue that the query complexity of any deterministic classical algorithm is $\Omega(2^n)$.

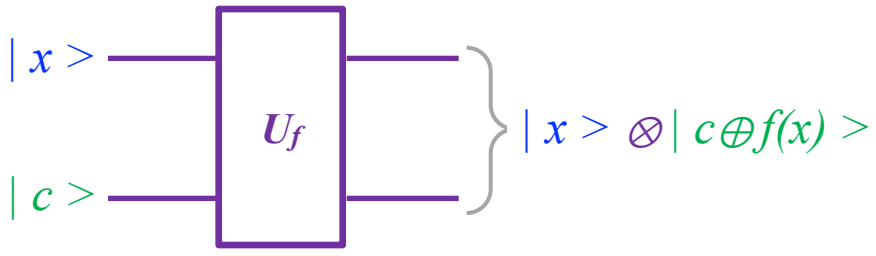
4. The Quantum Advantage

The explosive growth of the classical algorithm stems from the enormous size the domain of *f* (2^n elements) and the fact that we need to evaluate *f* at over half of them. Superposition, in the quantum realm, allows us to feed the oracle a superposition of all those 2^n possibilities and get a return that captures all the corresponding *f* values in just one oracle query! Hence, if we can somehow manipulate the return to give us a collision, we can compute s in O(1) query complexity instead of the classical $O(2^n)$. We will see shortly that the actual complexity is O(n), rather than O(1), but this is still exponentially better than the classical algorithm.

5. The Idea

Computing *f* at a superposition of states does indeed evaluate it at all those states in one shot, but the individual function values are inaccessible to us because they appear in a superposition, and once we measure, we only "see" one of them. Hence, we need to manipulate the quantum state, before measuring, so that the sought collision can be found.

Recall that the U_f oracle has the following general structure:



We feed an oracle in |x> an equal superposition of all 2^n possible values. As to the control input |c>, we note that unlike the Deutsch problem in which *f* is single-qubit-valued, the function in Simon's problem has multi-qubit values--as many qubits as the number of variables. Hence, the lower input must consist of n qubits, and we will initialize them to zeros.

The oracle's output would be a superposition of 2^n states each of which has a possible value for x entangled with a corresponding value for f(x):

$$\sum_{x=0}^{2^n-1} |x> \otimes |f(x)>$$

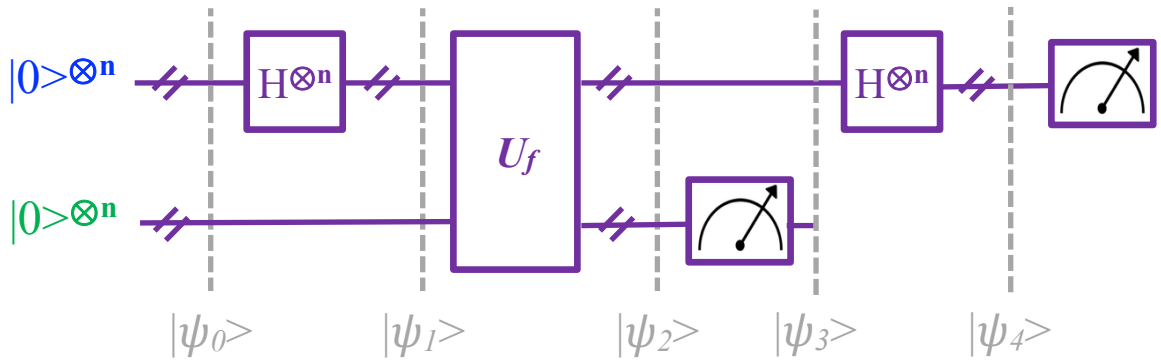
If we measure the lower output $|f(x)>$, it will collapse to one of the values of *f*, say *F*, and the entangled upper output $|x>$ will collapse to values of x that are consistent with *F*. Since *f* is 2-to-1, only two such values exist, and if we denote them by a and b, the upper output collapses to:

$$|a> + |b>, \text{ where } f(a) = f(b) = F \text{ and } b = a \oplus s$$

Note that the problem is done in the "quantum world" because s is simply $a \oplus b$. Unfortunately it is not done in "our world" because, even though we know the state is $|a>+|b>$, we have no way of knowing a and b individually. All we can do is measure the upper output, but if we do, we either get a or b, not both. Cloning the state is not allowed, and repeating the entire process may lead to other colliding pairs (depending on which value $|f(x)>$ collapsed to) so that also is a dead end. We therefore need to manipulate the state before we measure, and to that end, we follow the general pattern that we saw in the Apple Gedanken and add a "merge" on the upper output (i.e. H on each of its 2 qubits) to allow the components to interfere. Since a and b are connected via s, their interference pattern will reveal information about s. If this information is sufficient to determine s, we are done. Otherwise, we repeat the entire process to gain more information about s. Note that s is independent of which value $|f(x)>$ collapses to, so repetition is meaningful here. See the next section for details.

6. The Quantum Algorithm

The input consists of 2n qubits, n in the upper register and n in the lower, all initialized to $|0>$. Superposition is created in the upper input thru Hadamard gates and is then fed to the *Uf* gate that implements *f*. The gate's lower output is then measured. The upper output undergoes a second set of Hadamard gates before we arrive at the final upper output, which gets measured.



$$|\psi_0> = |000\ldots000> \otimes\ |000\ldots000>$$

$$|\psi_1> = \frac{|0>+|1>}{\sqrt{2}} \otimes \frac{|0>+|1>}{\sqrt{2}} \otimes \ldots \otimes \frac{|0>+|1>}{\sqrt{2}} \ldots \otimes |000\ldots000>$$

$$|\psi_1> = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k> \otimes |000\ldots000>$$

$$|\psi_2> = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k> \otimes |f(k)>$$

If we now measure the lower register of $|\psi_2>$, $|f(k)>$ will collapse to some value *F* and the entangled upper register |k> will collapse to values compatible with *F*, i.e. to values of x that map to *F*. Since *f* is 2-to-1 by the promise, only two such values, say a and b, can exist, and also by the promise, they must be connected by s. Hence:

$$|\psi_3> = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k>\ \otimes |F> \rightarrow \frac{1}{\sqrt{2}}(|a>\ +|b>)$$

where $f(a) = f(b) = F$ and $b = a \oplus s$

$$|\psi_4> = \frac{1}{\sqrt{2} \times \sqrt{2^n}} \sum_{l=0}^{2^n-1} [\ (-1)^{a.l} + (-1)^{b.l}\ ]\ |l>$$

$$|\psi_4> = \frac{1}{\sqrt{2} \times \sqrt{2^n}} \sum_{l=0}^{2^n-1} [\,(-1)^{a.l} + (-1)^{a \oplus s.l}\,]\,|l>$$

$$|\psi_4> = \frac{1}{\sqrt{2} \times \sqrt{2^n}} \sum_{l=0}^{2^n-1} (-1)^{a.l}\,[\,1 + (-1)^{s.l}\,]\,|l>$$

The final state exposes the spectacular interference effect: if *s.l* is odd for certain components |l> then those components interfere destructively, and we get zero probability of obtaining these components when we measure. But if it is even then the components interfere constructively, and we get 100% probability of obtaining these components. Hence, if we measure the upper output and find it in state |L> then we know that s.L = 0 (mod 2). This gives us some information about the sought s--in fact, it reduces in half the number of possible values of s. To learn more about it, we repeat the entire process enough times to determine s completely. Since s contains n bits, repeating the computation O(n) times should be sufficient to determine s completely.

Note: After n-1 repetitions, we get a homogeneous system of n-1 linear equations in n unknowns (the bits of s). It can be shown that this system will have a unique, non-zero solution with probability > 1/4. Hence, if we repeat m times (for a total of m(n-1) repetitions), the probability of failure $(1-1/4)^m < e^{-m/4}$ *drops exponentially with m.*

Hence, O(n) oracle queries are sufficient to determine s. This is an exponential speedup over the classical computation.

- Show that the measurement outcome of $|\psi_4>$ is equally likely to be any possible value of *l* that satisfies s.*l* = 0. Specifically, show that the probability of obtaining any such *l* is: $1/2^{n-1}$.
- Given two integers in $[0,2^{n-1}]$, r.s is defined as the number of common 1s in their binary representations. Show that it can also be defined as the dot product of r and s if thought of as ordinary vectors in an n-dimensional space in which each component is either 0 or 1.
- In reaching the final expression, we used the fact that ⊕ is distributive over . (the dot). Verify this through an example.
- Does the algorithm benefit from the value of *F* ? If yes, how; and if not, why measure it? *Hint: Think about this before reading the Appendix.*

7. A Non-Deterministic Classical Algorithm

If we settle for a *randomized* classical algorithm, one whose conclusion is mostly--but not always-- correct, can we do better than $O(2^n)$ (as in the Deutsch problem)? Rather than evaluating *f* at exponentially many arguments, let us randomly pick a few pairs from the domain in the hope of finding a collision amongst them (a pair that map to the same value under *f*). For example, let us pick a sample of k elements and compute the probability of finding at least one

collision in them? It is easier to compute the negation: what is the probability that the images of these elements are all distinct? Here is an estimate:

Each image is an integer in $[0,2^{n-1}]$ so it can have any one of $N=2^n$ different values. For an image to be distinct, it must be different from all the other images, and the probability of this is about $(N-k)/k = 1 - k/N$. Hence, the probability of all k images being distinct is:

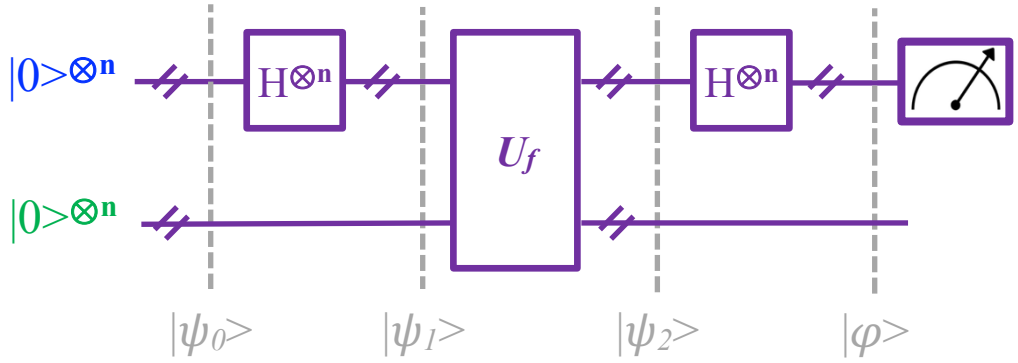
$$(1-\frac{k}{N})^k \approx 1-\frac{k^2}{N}$$

So even if we settle for 50% success rate, our sample size needs to be about $\sqrt{N}$ or 2^{n-1} which is still exponential. Hence, even non-deterministic classical algorithms are exponential in terms of the number of oracle queries.

- The above estimate, albeit correct in its conclusion, is rather rough. Derive a more rigorous one.
  Hint: the first element can have any image; the second can have any but that of the first; the third must avoid the first two images; and so on.
  Hint: $(1-r) \approx e^{-r}$ if $r < 1$. This $O(r^2)$ approximation allows us to turn a product of factors to a sum of exponents.
- Look up the definition of the "*birthday paradox*" which arises when discussing cryptographic hash functions. How does it tie in with the computation here?
- Look up the definition of the complexity classes BPP and BQP and show that the Simon problem defines a relation between them.

Appendix: Did we need to measure the lower register?

Since we did not benefit from the outcome of the lower input measurement, one wonders if we could have bypassed it, i.e. if we used this circuit diagram instead:



As before, we have:

$$|\psi_2> = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} |k> \otimes |f(k)>$$

Since we are not measuring the lower register, no collapse will take place and all the *f* values (along with the corresponding k values) will be involved. After the upper Walsh-Hadamard gate, the final state will be:

$$|\varphi> = \frac{1}{2^n} \sum_{k=0}^{2^n-1} \sum_{l=0}^{2^n-1} (-1)^{k.l} |l> \otimes |f(k)>$$

And since we will be measuring the upper register, let us rewrite the state as:

$$|\varphi> = \frac{1}{2^n} \sum_{l=0}^{2^n-1} |l> \otimes \sum_{k=0}^{2^n-1} (-1)^{k.l} |f(k)>$$

If we now measure the upper register $|l>$, what is the probability of getting a value L in $[0,2^n]$? The probability is the modulus squared of the amplitude, and we can clearly see from the above that the amplitude is:

$$\frac{1}{2^n} \sum_{k=0}^{2^n-1} (-1)^{k.L} |f(k)>$$

The sum involves all the 2^n possible values of $|f(k)>$, but since *f* is 2-to-1, these values pair up, i.e. there are only half as many distinct values. Specifically, for every k, there exists a k' = k⊕s such that *f*(k') = *f*(k). Hence, the above sum can be written as:

$$\frac{1}{2^n} \sum_k [\, (-1)^{k.L} + (-1)^{k \oplus s.L} \,] \, |f(k)>$$

This sum involves half as many elements and k takes on all the values in the *range* of *f* (rather than its co-domain). Pulling out the common factor yields:

$$\frac{1}{2^n} \sum_k (-1)^{k.L} [\, 1 + (-1)^{s.L} \,] \, |f(k)>$$

This amplitude is zero if s.L = 1. Hence, we will only see outcomes that satisfy s.L = 0 and the probability of measuring any of them is:

$$\Pr(L|s.L=0) = \frac{1}{2^{2n}} \left\| \sum_k 2(-1)^{k.L} \, |f(k)> \right\|^2 = \frac{4 \times 2^{n-1}}{2^{2n}} = \frac{1}{2^{n-1}}$$

Hence, the outcome of the measurement L gives us information about s, exactly the same as in the other circuit where we measured both registers. It is therefore not necessary to measure the lower register. Interestingly, Daniel Simon, the inventor of this algorithm, did incorporate the lower-register measurement because, as he put it: "it would have been a lot harder to think through the algorithm without this".

Remarks

- This algorithm was proposed by Daniel Simon c. 1994.
- There is some similarity between Simon's and Deutsch-Jozsa problems, but the problem here is not to distinguish between two cases (constant vs balanced) but between exponentially many cases, one per value of s.
- As is the case in query complexity analysis, we focus only on query count and ignore everything else. For example, gate delays, memory, and other resources involved in computing f are not counted, nor is the $O(n^3)$ Gaussian elimination of the post processing.
- This algorithm employs the so-called *quantum parallelism* by computing all values of *f(x)* in one shot. The result appears as a superposition of states each of which involves a value of x together with a value of *f* at that x. The algorithm manipulates the phases of these terms in such a way that exposes the sought property. Specifically, the terms interfere destructively (i.e. cancel out) if *f* has a particular global property (1-to-1) and interfere constructively if *f* is 2-to-1.
- This success of this algorithm can be traced back directly to the first pillar of quantum computing: *superposition*. Its circuit has the same characteristics as our Apple Gedanken (or its implementation in a Mech-Zehnder interferometer); namely: a fork to induce superposition (a splitter), and a merge (a 2nd splitter) to allow the superposition components to interfere.
- Note that *entanglement*, the second pillar of quantum computing, plays a key role in (the original, two-measurement version) of this algorithm. The two outputs of *Uf* are entangled so measuring the lower one collapses the upper to a state compatible with the outcome of that measurement.
- The *Deferred Measurement Principle* states that delaying measurements until the end of a quantum computation doesn't affect the probability distribution of the outcomes.