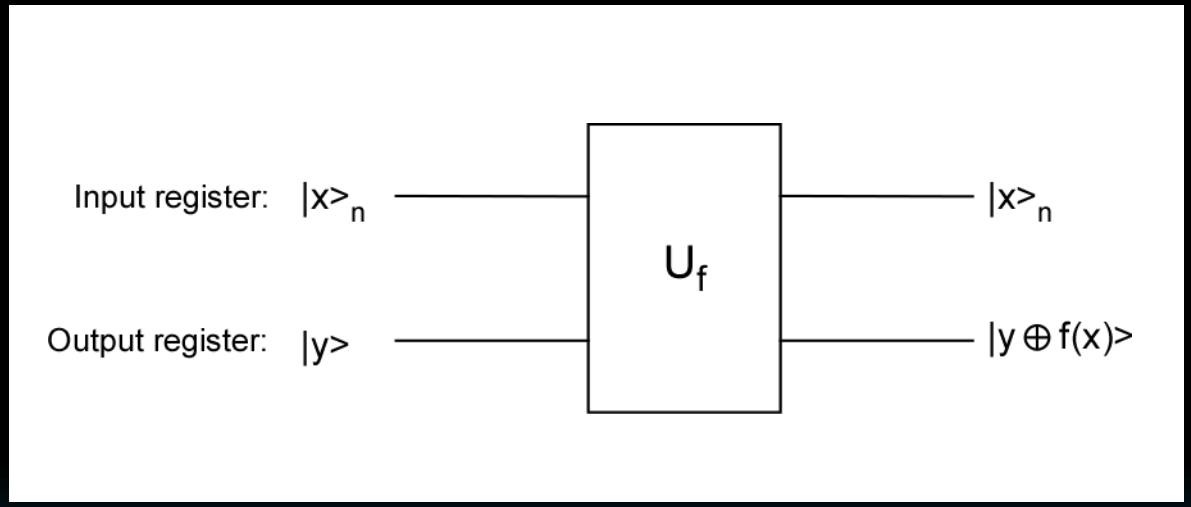
Simon's Problem

Background: The Deutsch-Jozsa problem and the Bernstein-Vazirani uncover the behavior of an "oracle" but the speedup over random classical algorithms isn't really that impressive.

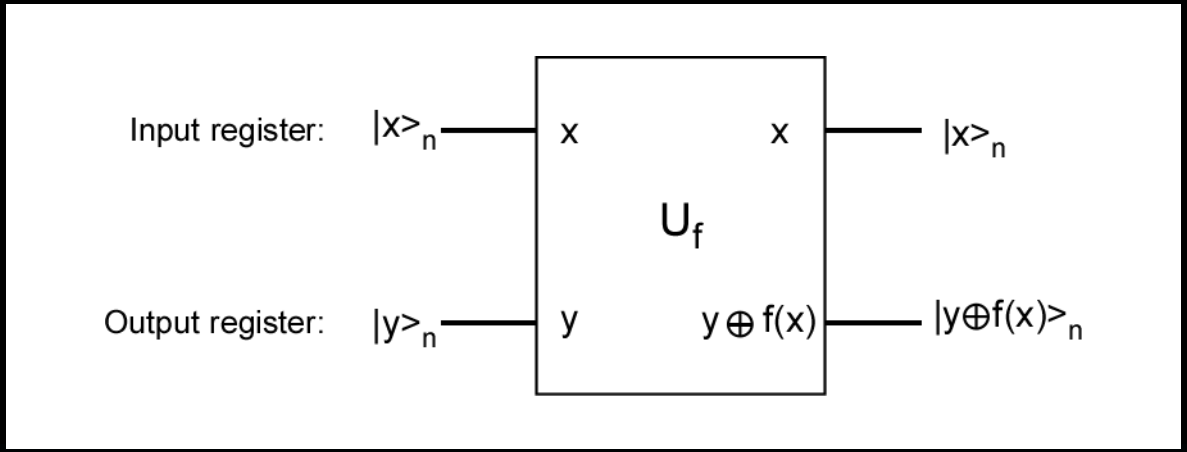


Goal: To show a *non-trivial* problem where a random classical algorithm takes a number of queries exponential in the number of bits, but which is solvable in a number of queries linear in the number of bits (plus some classical post-processing).

1. Simon's Problem

Simon's algorithm discovers the values of a "hidden string" a that defines the behavior of an oracle. The input and output registers are both n-Qbit quantum states.

$$f : \{0,1\}^n \rightarrow \{0,1\}^n$$



The function f(x) is a 2-1 function with the behavior

$$[f(x) = f(y)] \iff [y = x \oplus a]$$

$$f(x \oplus a) = f(x)$$

2. A Simon example

$$f(x \oplus a) = f(x)$$

x		f(x)
0	0000	0110
1	0001	1000
2	0010	1001
3	0011	1101
4	0100	1001
5	0101	1101
6	0110	0110
7	0111	1000
8	1000	1110
9	1001	0000
10	1010	0101
11	1011	1001
12	1100	0101
13	1101	1001
14	1110	1110
15	1111	0000

While the function f(x) is “periodic,” that period is NOT terribly apparent when you look at the function output.

To classically find the hidden string *a* you need to find two values with the same output:

$$x_0 \text{ and } y_0 = x_0 \oplus a$$

Note that:

$$x_0 \oplus y_0 = x_0 \oplus x_0 \oplus a = a$$

Example:

$$f(x_0 = 0001) = f(y_0 = 0111) = 1000$$

$$x_0 \oplus y_0 = 0110$$

You check: Is this true for any “collision pair?”

How many queries do you have to make to find a collision?

x		f(x)
0	0000	0110
1	0001	1000
2	0010	1001
3	0011	1101
4	0100	1001
5	0101	1101
6	0110	0110
7	0111	1000
8	1000	1110
9	1001	0000
10	1010	0101
11	1011	1001
12	1100	0101
13	1101	1001
14	1110	1110
15	1111	0000

This is something like the “birthday paradox” problem. If you take a collection of *N* uniformly distributed numbers (days of the year/birthdays) the probability of having any two be equal when you randomly sample a set containing *m* values is

$$P_{\text{collision}} \approx 1 - e^{m^2/2N}$$

Conversely, you can set the likelihood of a collision and ask how many samples you need to take to get that

$$m \approx \sqrt{2N \ln\left(\frac{1}{1 - P_{\text{collision}}}\right)}$$

Example: N = 365, $P_{\text{collision}} = 0.5$

$$m \approx 23$$

Example: N = 365, $P_{\text{collision}} = 0.99$

$$m \approx 58$$

How many queries do you have to make to find a collision?

x		f(x)
0	0000	0110
1	0001	1000
2	0010	1001
3	0011	1101
4	0100	1001
5	0101	1101
6	0110	0110
7	0111	1000
8	1000	1110
9	1001	0000
10	1010	0101
11	1011	1001
12	1100	0101
13	1101	1001
14	1110	1110
15	1111	0000

If the number $N = 2^n$

$$m \approx \sqrt{2^{n+1} \ln\left(\frac{1}{1 - P_{\text{collision}}}\right)}$$

$$= 2^{n/2} \sqrt{2 \ln\left(\frac{1}{1 - P_{\text{collision}}}\right)}$$

The number of queries needed scales as the square root of the number of values in the set (possible birthdays) – which seems impressive.

However, the number of queries is *exponential* in the number of bits *n*.

Note: This is *not* a lower bound, but it turns out that the lower bound also scales the same way with *n*.

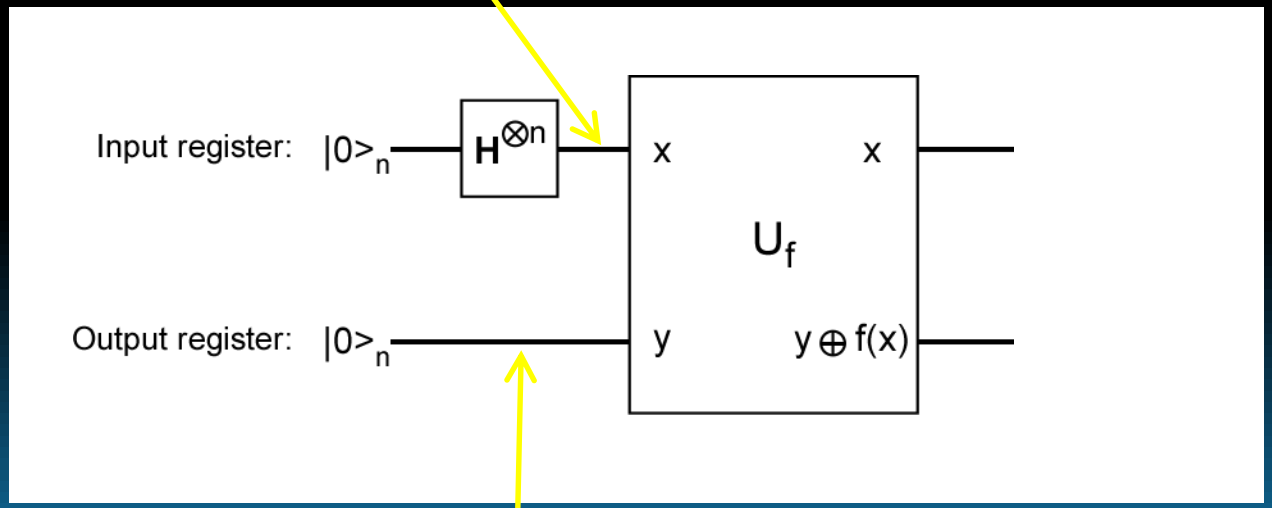
3. Solving Simon's Problem Quantum Mechanically

a. Quantum Parallelism

The input to the oracle, $|\psi_1\rangle$ is a state that is a superposition of all possible input register-states in a tensor product with the state $|0\rangle_n$ for the output register.

$$|\psi_i\rangle_n = \mathbf{H}^{\otimes n}|0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{0 \le x < 2^n} |x\rangle_n$$

$$|\psi_1\rangle = |\psi_i\rangle|\psi_o\rangle$$



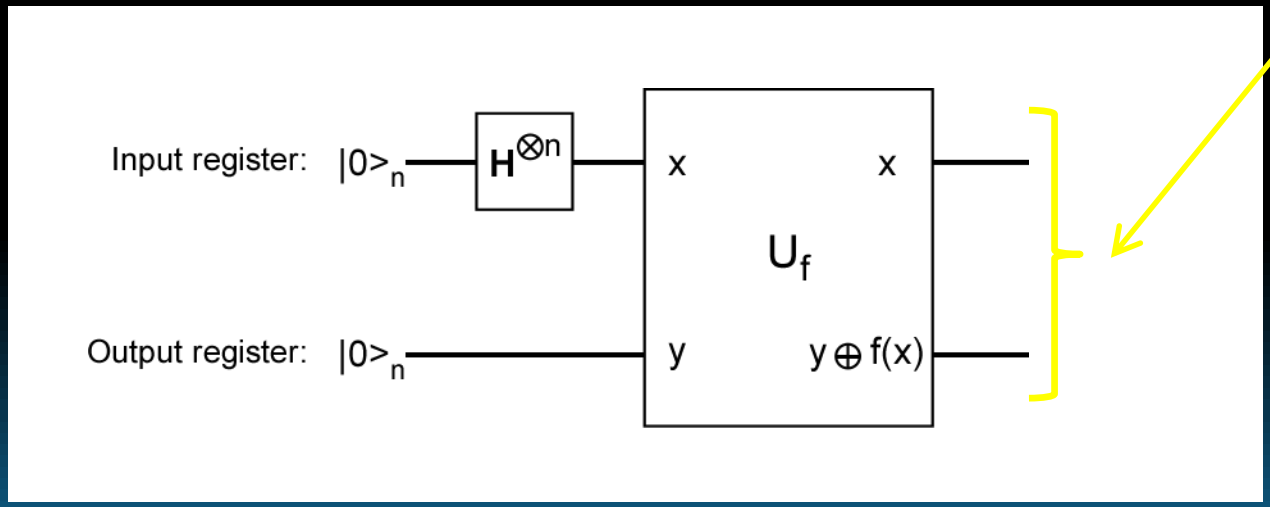
$$|\psi_o\rangle_n = |0\rangle_n$$

b. Output of the oracle

The output of the (linear) oracle is a state that is entangled between the input and output registers.

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{0 \le x < 2^n} \mathbf{U}_f |x\rangle_n |0\rangle_n$$

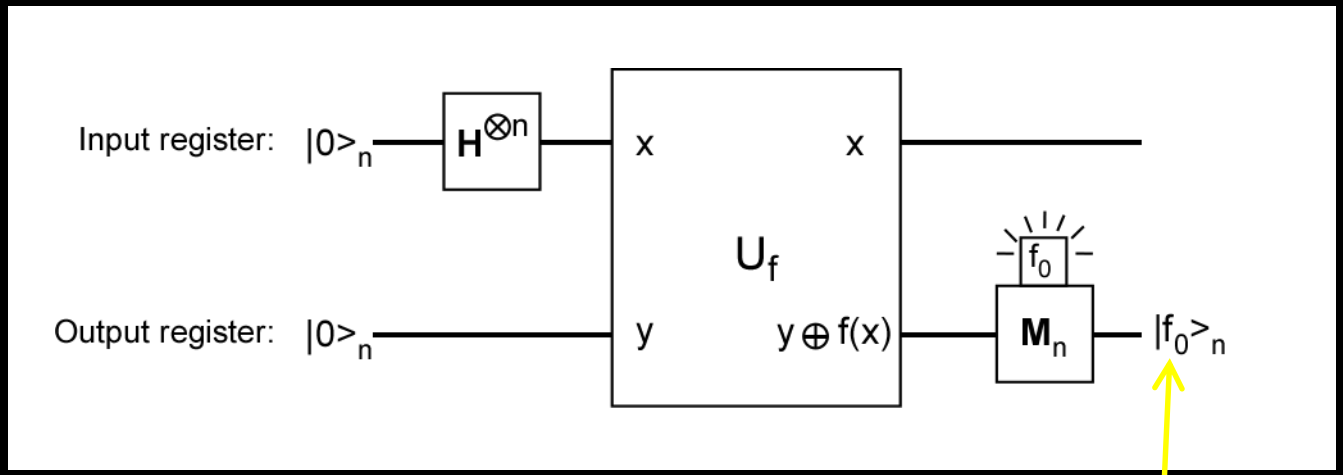
$$= \frac{1}{\sqrt{2^n}} \sum_{0 \le x < 2^n} |x\rangle_n |f(x)\rangle_n$$



All possible x values and their associated f(x) values are equally weighted. There are half as many f(x) values as x-values.

c. Manipulating the output to get an answer!

The output register is measured, and gives specific value of f(x), which is drawn with equal probability from all possible values of f(x).

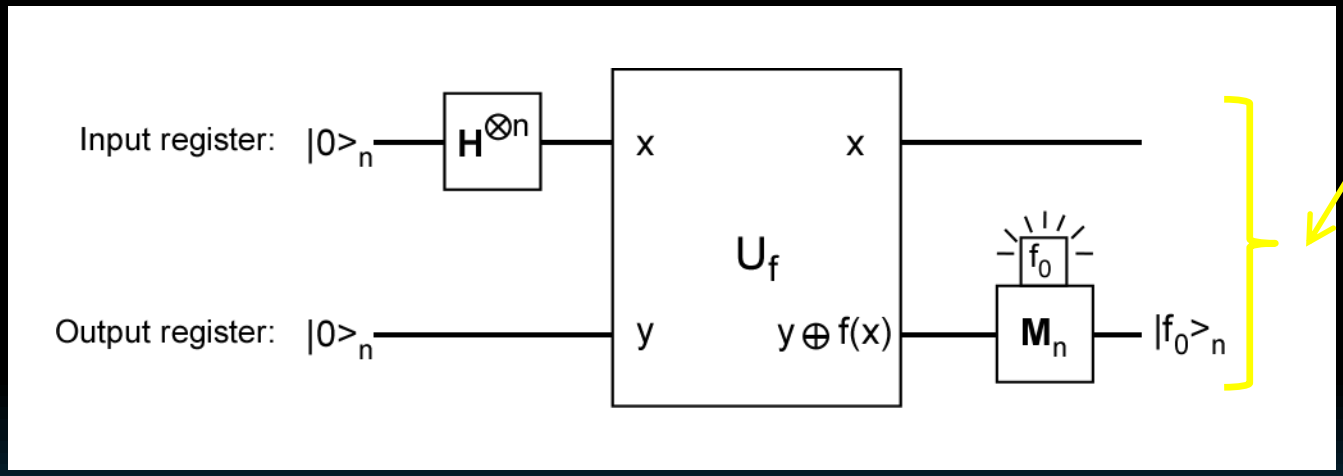


$$|\psi_o\rangle = |f_0\rangle = |f(x_0)\rangle = |f(y_0)\rangle$$

This is a partial measurement – which leaves the system in a (normalized) state which is conditioned on the output register being in the state $|f_0\rangle$.

There are two possible values of the input register that are consistent with a specific value of the output register.

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}\{|x_0\rangle_n + |y_0\rangle_n\}|f_0\rangle_n$$
$$= \frac{1}{\sqrt{2}}\{|x_0\rangle_n + |x_0 \oplus a\rangle_n\}|f_0\rangle_n$$



If we could just *clone* the output state, we could measure the state multiple times and determine both x_0 and y_0 (and therefore a), but the no-cloning theorem says we're out of luck with that approach!

d. Manipulating the output to get an answer, Part 2.

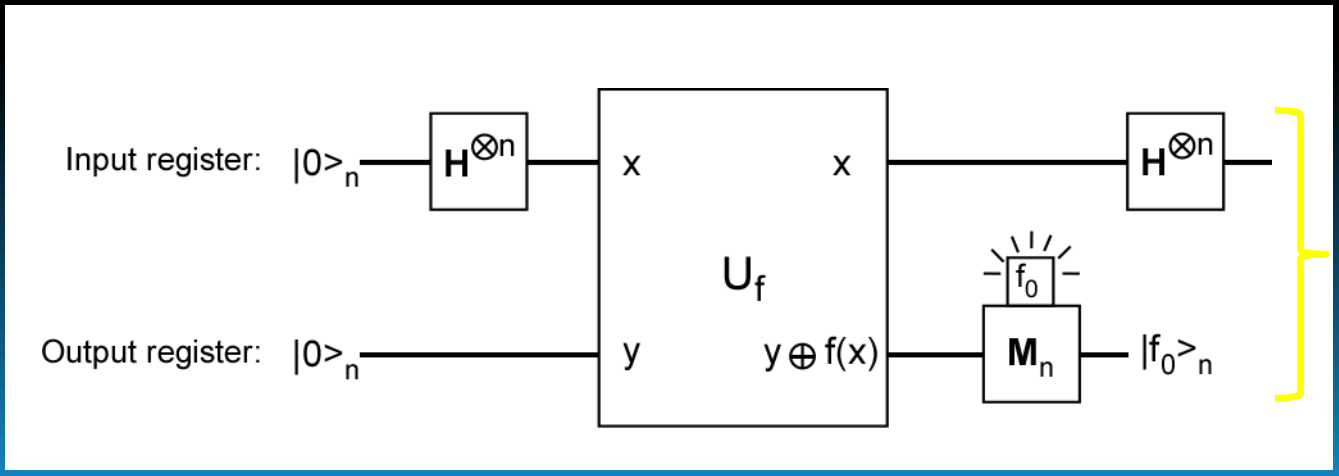
Hadamard the input register, and remember the rule:

$$\mathbf{H}^{\otimes n}|x\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{0 \le z < 2^n} (-1)^{x \cdot z} |z\rangle_n$$

Which gives:

$$|\psi_4\rangle = \frac{1}{\sqrt{2}} \left\{ H^{\otimes n}|x_0\rangle_n + H^{\otimes n}|y_0\rangle_n \right\} |f_0\rangle_n$$

$$= \left\{ \frac{1}{\sqrt{2^{n+1}}} \sum_{0 \le z < 2^n} \left[(-1)^{x_0 \cdot z} + (-1)^{y_0 \cdot z} \right] |z\rangle_n \right\} |f_0\rangle_n$$



There are two parts to the output state, and they can either add to two or subtract to zero.

$$|\psi_4\rangle = \left\{ \frac{1}{\sqrt{2^{n+1}}} \sum_{0 \le z < 2^n} \left[(-1)^{x_0 \cdot z} + (-1)^{y_0 \cdot z} \right] |z\rangle_n \right\} |f_0\rangle_n$$

They add if:

$$x_0 \cdot z = y_0 \cdot z \pmod 2$$

$$\Downarrow$$

$$x_0 \cdot z - y_0 \cdot z = 0 \pmod 2$$

$$\Downarrow$$

$$x_0 \cdot z + y_0 \cdot z = 0 \pmod 2$$

$$\Downarrow$$

$$(x_0 + y_0) \cdot z = 0 \pmod 2$$

$$\Downarrow$$

$$(x_0 \oplus y_0) \cdot z = 0 \pmod 2$$

Recall that dot products are defined modulo 2.

$$a \cdot z = 0 \pmod 2$$

There are two parts to the output state, and they can either add to two or subtract to zero.

$$|\psi_4\rangle = \left\{ \frac{1}{\sqrt{2^{n+1}}} \sum_{0 \le z < 2^n} \left((-1)^{x_0 \cdot z} + (-1)^{y_0 \cdot z} \right) |z\rangle_n \right\} |f_0\rangle_n$$

They add if: $x_0 \cdot z = y_0 \cdot z \pmod 2$

$$\Downarrow$$

$$x_0 \cdot z - y_0 \cdot z = 0 \pmod 2$$

$$\Downarrow$$

$$x_0 \cdot z + y_0 \cdot z = 0 \pmod 2$$

$$\Downarrow$$

$$(x_0 + y_0) \cdot z = 0 \pmod 2$$

$$\Downarrow$$

$$(x_0 \oplus y_0) \cdot z = 0 \pmod 2$$

For every state $|z\rangle$ that is part of the sum!

$$a \cdot z = 0 \pmod 2$$

with a coefficient of 2.

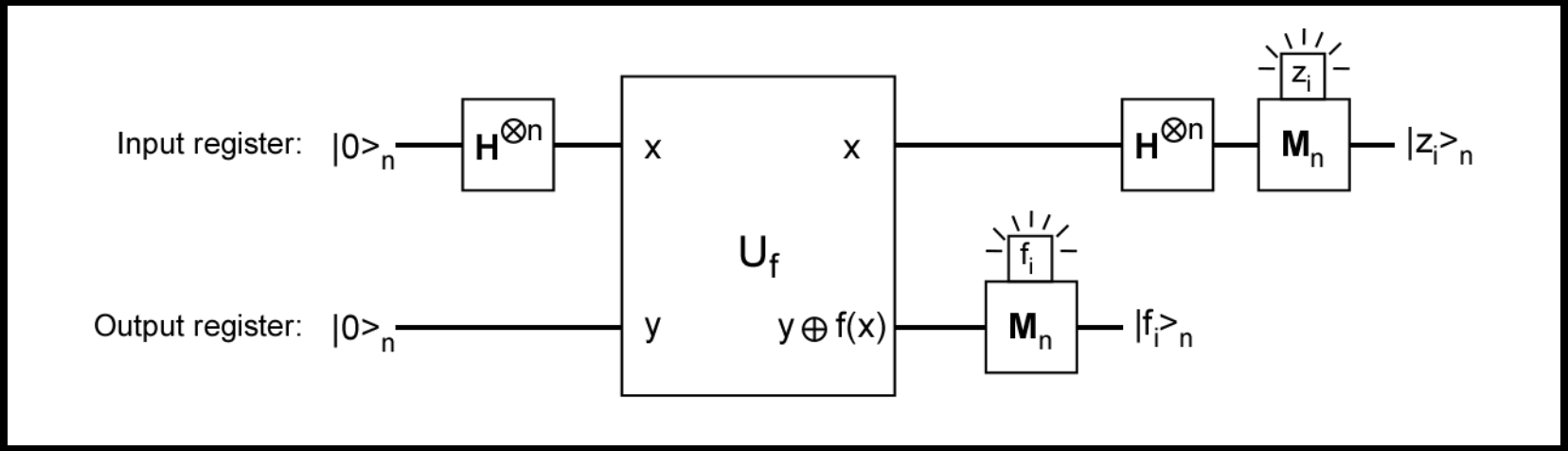
But every state $|z\rangle$ with

$$a \cdot z = 1 \pmod 2$$

will have a zero coefficient.

e. Measuring again to get part of the solution

Measure the input register to determine a value z_i with $z_i \oplus a = 0$.



Each state $|z_i\rangle$ is found with equal probability

$$P_{z_i} = \left| \frac{2}{\sqrt{2^{n+1}}} \right|^2 = \frac{1}{2^{n-1}}$$

f. Using the parts

Repeat the experiment determining at least (n-1) distinct times to get n-1 *linearly independent equations*

$$z_i \oplus a = 0 \qquad (0 \le i \le n-1)$$

Solve the equations using a classical computing algorithm (Gaussian Elimination) *Which takes, O(n^3) classical operations. (Polynomial in n)*

Note 1: If you already have k independent values of z, the probability that the next one you choose is independent is

$$P = \frac{2^{n-1} - 2^k}{2^{n-1}} = 1 - \frac{2^k}{2^{n-1}} > 0.5$$

So, you will get an independent set in O(n) queries of the oracle.

Note 2: Mermin shows (Appendix G) that if you do n+x queries, the probability that you will have n-1 independent equations is

$$P > 1 - \frac{1}{2^{x+1}}$$

Which can be made arbitrarily close to 1 for a constant x (independent of n)