D.2 Simon


1. For breaking RSA we will see that its useful to know the period of a function: that \( r \) such that \( f(x + r) = f(x) \). Simon’s problem is a warmup for this.

2. **Simon’s Problem:** Suppose we are given an unknown function \( f : 2^n \rightarrow 2^n \) and we are told that it is *two-to-one.*
   This means \( f(x) = f(y) \) iff \( x \oplus y = r \) for some fixed \( r \in 2^n \).
   The vector \( r \) can be considered the *period* of \( f \), since \( f(x \oplus r) = f(x) \).

3. The problem is to determine the period \( r \) of an unknown \( f \).

4. **Classical solution:** Since we don’t know anything about \( f \), the best we can do is evaluate it on random inputs.
   If we are ever lucky enough to find \( x \) and \( x' \) such that \( f(x) = f(x') \), then we have our answer, \( r = x \oplus x' \).

5. On the average you need to do \( 2^{n/2} \) function evaluations, which is exponential in the size of the input.
   For \( n = 100 \), it would require about \( 2^{50} \approx 10^{15} \) evaluations. “At 10 million calls per second it would take about three years.” [MQCS 55]

6. **Quantum algorithm:** We will see that a quantum computer can determine \( r \) with high probability (\( > 1 - 10^{-6} \)) in about 120 evaluations.
   At 10 million calls per second, this would take about 12 microseconds!

7. **Input superposition:** As before, start by using the Walsh-Hadamard transform to create a superposition of all possible inputs:
   \[
   |\psi_1\rangle \overset{\text{def}}{=} H^\otimes n |0\rangle^\otimes n = \frac{1}{2^{n/2}} \sum_{x \in 2^n} |x\rangle.
   \]

8. **Function evaluation:** Suppose that \( U_f \) is the quantum gate array implementing \( f \) and recall \( U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle \). Therefore:
   \[
   |\psi_2\rangle \overset{\text{def}}{=} U_f|\psi_1\rangle|0\rangle^\otimes n = \frac{1}{2^{n/2}} \sum_{x \in 2^n} |x\rangle|f(x)\rangle.
   \]
Therefore we have an equal superposition of corresponding input-output values.

\[ \text{¶9. Output measurement:} \] Measure the output register (in the computational basis) to obtain some \(|z\rangle\).

Since the function is two-to-one, the projection will have a superposition of two inputs:

\[
\frac{1}{\sqrt{2}} (|x_0\rangle + |x_0 \oplus r\rangle) |z\rangle,
\]

where \( f(x_0) = z = f(x_0 \oplus r) \).

\[ \text{¶10. The information we need is contained in the input register,} \]

\[
|\psi_3\rangle \overset{\text{def}}{=} \frac{1}{\sqrt{2}} (|x_0\rangle + |x_0 \oplus r\rangle),
\]

but it cannot be extracted directly.

If we measure it, we will get either \( x_0 \) or \( x_0 \oplus r \), but not both, and we need both to get \( r \).

(We cannot make two copies, due to the no-cloning theorem.)

\[ \text{¶11. Suppose we apply the Walsh-Hadamard transform to this superposition:} \]

\[
H^{\otimes n}|\psi_3\rangle = H^{\otimes n} \frac{1}{\sqrt{2}} (|x_0\rangle + |x_0 \oplus r\rangle) = \frac{1}{\sqrt{2}} (H^{\otimes n}|x_0\rangle + H^{\otimes n}|x_0 \oplus r\rangle).
\]

\[ \text{¶12. Now, recall (¶13, p. 146) that} \]

\[
H^{\otimes n}|x\rangle = \frac{1}{2^{n/2}} \sum_{y \in 2^n} (-1)^{x \cdot y} |y\rangle.
\]

Therefore,

\[
H^{\otimes n}|\psi_3\rangle = \frac{1}{\sqrt{2}} \left[ \frac{1}{2^{n/2}} \sum_{y \in 2^n} (-1)^{x_0 \cdot y} |y\rangle + \frac{1}{2^{n/2}} \sum_{y \in 2^n} (-1)^{(x_0 \oplus r) \cdot y} |y\rangle \right]
= \frac{1}{2^{(n+1)/2}} \sum_{y \in 2^n} \left[ (-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus r) \cdot y} \right] |y\rangle.
\]

\[ \text{¶13. Note that} (-1)^{(x_0 \oplus r) \cdot y} = (-1)^{x_0 \cdot y} (-1)^{r \cdot y}. \]

Therefore, if \( r \cdot y = 1 \), then the bracketed expression is 0 (since the
terms have opposite sign and cancel).
However, if $r \cdot y = 0$, then the bracketed expression is $2(-1)^{x_0 \cdot y}$ (since they don’t cancel).

¶14. Hence the result of the Walsh-Hadamard transform is

$$|\psi_4\rangle = H^{\otimes n}|\psi_3\rangle = \frac{1}{2^{(n-1)/2}} \sum_{y \text{ s.t. } r \cdot y = 0} (-1)^{x_0 \cdot y}|y\rangle.$$ 

¶15. **Measurement:** Measuring the input register (in the computational basis) will collapse it with equal probability into a state $|y^{(1)}\rangle$ such that $r \cdot y^{(1)} = 0$.

¶16. **First equation:** Since we know $y^{(1)}$, this gives us some information about $r$, expressed in the equation:

$$y_1^{(1)} r_1 + y_2^{(1)} r_2 + \cdots + y_n^{(1)} r_n = 0 \pmod{2}.$$

¶17. **Iteration:** The quantum computation can be repeated, producing a series of bit strings $y^{(1)}, y^{(2)}, \ldots$ such that $y^{(k)} \cdot r = 0$.
From them we can build up a system of $n$ linearly-independent equations and solve for $r$.
(If you get a linearly-dependent equation, you have to try again.)

¶18. Note that each quantum step (involving one evaluation of $f$) produces an equation (except in the unlikely case $y^{(k)} = 0$ or that it’s linearly dep.), and therefore determines one of the bits in terms of the other bits.
That is, each iteration reduced the candidates for $r$ by approximately one-half.

¶19. **Probability:** A mathematical analysis (Mermin, 2007, App. G) shows that with $n + m$ iterations the probability of having enough information to determine $r$ is $> 1 - \frac{1}{2^{m+1}}$.
“Thus the odds are more than a million to one that with $n + 20$ invocations of $U_f$ we will learn $|r\rangle$, no matter how large $n$ may be.” (Mermin, 2007, p. 57)

¶20. **Exponential speedup:** Therefore Simon’s problem can be solved in linear time on a quantum computer, but requires exponential time on a classical computer.