Simon's Algorithm

Simon's algorithm was the algorithm that inspired Shor in making the Shor's Algorithm. This is a great algorithm to have a look at hybrid algorithms, as this is a hybrid algorithm.

The problem is : –

```
Suppose there's a binary string of length n. Binary string is a string composed of 0
s and 1s. There's a function f, where f(x) = f(y), if and only if, y = x or y = x ⊕
 s. x and y are binary strings of length n. s is a “secret” binary string of length
 n, where s is not all 0s. So, s can be any one of the possible (2ⁿ - 1) binary stri
ngs.

We've to find the secret string s.

Example : -
    Suppose that n = 3.
    Suppose we find that f(000) = f(101), it means that 000 ⊕ s = 101.
    Remember, these are binary strings and not binary numbers. So 111 ⊕ 100 = 011.
    From the above information, we know that s is 101, as 000 ⊕ 101 = 101.

The question is, how many times do we need to evaluate f, to find s?
Also, we don't know what the secret string s or the function f are.

Classically, we need at least 5 evaluations. We evaluate any four strings of length
 n, on the the function f, and they might all give different results. But the evalua
tion of the fifth string, on the function f, is bound to repeat one of the values fr
om the previously evaluated four strings. Suppose f(010) and f(110) give the same re
sult. That means, f(010) = f(110), which means 010 ⊕ s = 110. Adding 010 to both sid
es, 010 ⊕ 010 ⊕ s = 010 ⊕ 110 => 000 ⊕ s = 100, s = 100.

In general, for binary string of length n, we need to make (2ⁿ⁻¹ + 1) evaluations.
```

Kronecker Product of Hadamard Gate

We know that the Hadamard Gate can be represented by the matrix, $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

- Applying H gate on 2 qubits, in the state $|00\rangle$, we get, $|00\rangle + |01\rangle + |10\rangle + |11\rangle$

- Applying H gate on 2 qubits, in the state $|01\rangle$, we get, $|00\rangle - |01\rangle + |10\rangle - |11\rangle$
- Applying H gate on 2 qubits, in the state $|10\rangle$, we get, $|00\rangle + |01\rangle - |10\rangle - |11\rangle$
- Applying H gate on 2 qubits, in the state $|11\rangle$, we get, $|00\rangle - |01\rangle - |10\rangle + |11\rangle$

The Matrix representation for $H^{\otimes 2}$ can be given as, $\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$

```
using Yao, YaoPlots
```

Which we can verify below

```
4×4 Array{Complex{Float64},2}:
 0.5+0.0im   0.5+0.0im   0.5+0.0im   0.5+0.0im
 0.5+0.0im  -0.5+0.0im   0.5+0.0im  -0.5+0.0im
 0.5+0.0im   0.5+0.0im  -0.5+0.0im  -0.5+0.0im
 0.5+0.0im  -0.5+0.0im  -0.5+0.0im   0.5-0.0im
```

```
Matrix(repeat(2, H, 1:2))
```

We can rewrite the above as

$$\frac{1}{\sqrt{2}}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Which can be again rewritten as

$$H^{\otimes 2} = \frac{1}{\sqrt{2}}\begin{bmatrix} H & H \\ H & -H \end{bmatrix}$$

Following this trend,

$$H^{\otimes 3} = \frac{1}{\sqrt{2}}\begin{bmatrix} H^{\otimes 2} & H^{\otimes 2} \\ H^{\otimes 2} & -H^{\otimes 2} \end{bmatrix}$$

$$H^{\otimes 4} = \frac{1}{\sqrt{2}}\begin{bmatrix} H^{\otimes 3} & H^{\otimes 3} \\ H^{\otimes 3} & -H^{\otimes 3} \end{bmatrix}$$

-
-
-

$$H^{\otimes n} = \frac{1}{\sqrt{2}}\begin{bmatrix} H^{\otimes n-1} & H^{\otimes n-1} \\ H^{\otimes n-1} & -H^{\otimes n-1} \end{bmatrix}$$

This is known as the Kronecker product of Hadamard Gate.

Dot Product of binary strings

The dot product of two binary strings, a and b, both of length n, where $a = a_0a_1a_2....a_{n-1}$, *and* $b = b_0b_1b_2....b_{n-1}$, *the **dot product** of* a *and* b, $a \cdot b$, *is defined as*

```
a · b = a₀ × b₀ ⊕ a₁ × b₁ ⊕ a₂ × b₂ .... aₙ₋₁ × bₙ₋₁
```

It's always equal to 0 or 1. If $a = 0010$ and $b = 0101$, then

```
a · b = 0 × 0 ⊕ 0 × 1 ⊕ 1 × 0 ⊕ 0 × 1 = 0 ⊕ 0 ⊕ 0 ⊕ 0 = 0
```

Lets check out the dot products of all possible combinations for binary strings where $n = 2$.

$\begin{bmatrix} 00 \cdot 00 & 00 \cdot 01 & 00 \cdot 10 & 00 \cdot 11 \\ 01 \cdot 00 & 01 \cdot 01 & 01 \cdot 10 & 01 \cdot 11 \\ 10 \cdot 00 & 10 \cdot 01 & 10 \cdot 10 & 10 \cdot 11 \\ 11 \cdot 00 & 11 \cdot 01 & 11 \cdot 10 & 11 \cdot 11 \end{bmatrix}$. Which calculates to $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$.

Remember $H^{\otimes 2}$, which could be represented by, $\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$?

It can also be represented by, $\frac{1}{2}\begin{bmatrix} (-1)^{00\cdot 00} & (-1)^{00\cdot 01} & (-1)^{00\cdot 10} & (-1)^{00\cdot 11} \\ (-1)^{01\cdot 00} & (-1)^{01\cdot 01} & (-1)^{01\cdot 10} & (-1)^{01\cdot 11} \\ (-1)^{10\cdot 00} & (-1)^{10\cdot 01} & (-1)^{10\cdot 10} & (-1)^{10\cdot 11} \\ (-1)^{11\cdot 00} & (-1)^{11\cdot 01} & (-1)^{11\cdot 10} & (-1)^{11\cdot 11} \end{bmatrix}$

So yeah, we can use dot products to denote Knonecker products of Hadamard Gates

Now, assume $s = 11$. We're going to add the columns with the pairs `x` and `x ⊕ s`. In other words, in this case, columns 1 and 4, and, columns 2 and 3.

Adding columns 1 and 4,

$$\frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 \\ 0 \\ 0 \\ 2 \end{bmatrix}$$

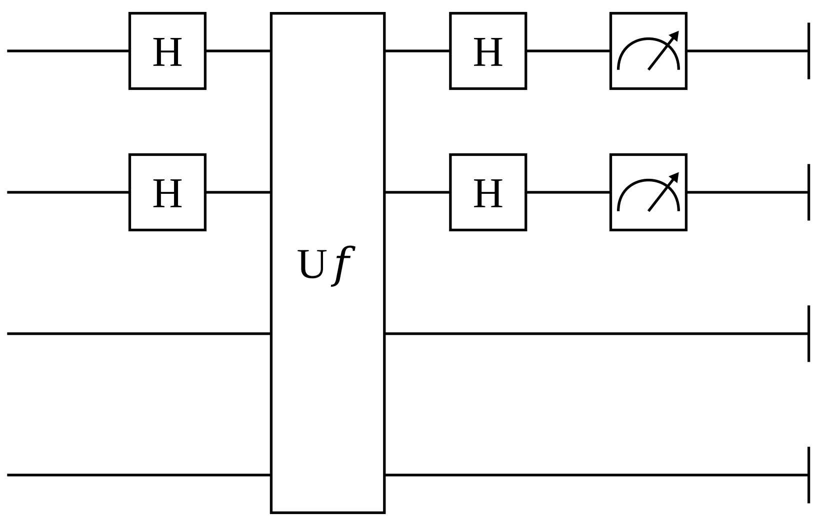
Similarly adding columns 2 and 3,

$$\frac{1}{2}\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}+\frac{1}{2}\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}=\frac{1}{2}\begin{bmatrix}2\\0\\0\\-2\end{bmatrix}$$

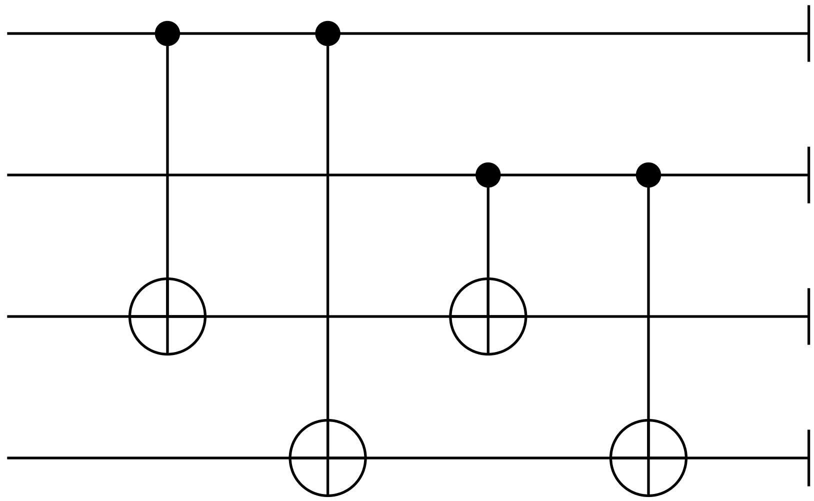
As the above vectors are state vectors, when doing the operation `x ⊕ s`, we see that some probability amplitudes are getting amplified and some are getting cancelled. If you've studied exponents, you know that $(-1)^{a\cdot(b\oplus s)}=(-1)^{a\cdot b}(-1)^{a\cdot s}$. It means, if $a\cdot s=0$, then $(-1)^{a\cdot(b\oplus s)}=(-1)^{a\cdot b}$, hence they get added, and if, $a\cdot s=1$, then $(-1)^{a\cdot(b\oplus s)}=-(-1)^{a\cdot b}$, hence they get cancelled out.

Circuit Implementation

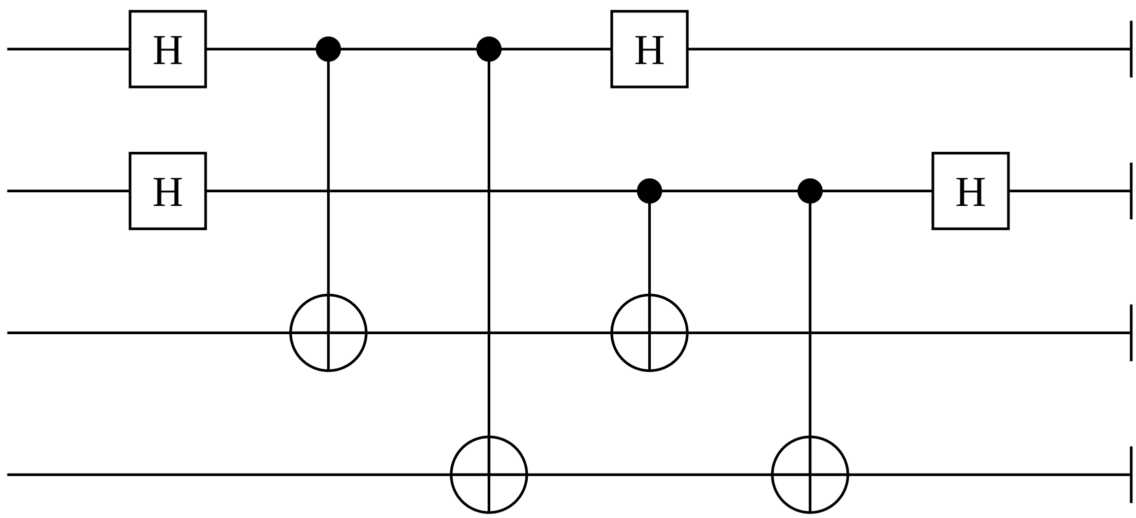
The circuit looks like this, where it takes a string of 0^{2n} as input, and the first n inputs, i.e. x, return x after passing through the circuit, and the next n inputs, i.e. y, return `y ⊕ f(x)`, after passing through the circuit. Let's call this circuit U_f. This is for $n=2$.




```
s = "11"
s = string(rand(1 : (2^2 - 1)), base=2, pad=2)
```



```
begin
    if s == "11"
        U𝑓 = chain(4, control(1,3=>X), control(1,4=>X), control(2,3=>X),
control(2,4=>X))
    elseif s == "01"
        U𝑓 = chain(4, control(1,3=>X), control(1,4=>X))
    elseif s == "10"
        U𝑓 = chain(4, control(2,3=>X), control(2,4=>X))
    end
    plot(U𝑓)
end
```




```
begin
    SimonAlgoCircuit_for_n_2 = chain(4, repeat(H, 1:2), put(1:4=>U𝑓), repeat(H, 1:2))
    plot(SimonAlgoCircuit_for_n_2)
end
```

```
output =
▶BitBasis.BitStr{2,Int64}[11 ₍₂₎, 00 ₍₂₎, 11 ₍₂₎, 00 ₍₂₎, 11 ₍₂₎, 11 ₍₂₎, 00 ₍₂₎, 00
```

```
output = zero_state(4) |> SimonAlgoCircuit_for_n_2 |> r->measure(r, 1:2, nshots=1024)
```

The reason it's a hybrid algorithm, is that we got two states, for $n = 2$, which have equal chances of being the secret string s. From here on, we've to classically deduce which of the measured states can be the output. Since s can't be 00, s = 11.

Note that this implementation is specific to $n = 2$

Deduction gets really complicated as n increases, and while its very very unlikely, on real quantum machines, there's a chance that you'll never get the secret string s for any number of runs, or nshots. This algorithm doesn't have much use/application cases either. Shor was inspired by this algorithm to make a general period finding algorithm.