Simon's Algorithm

Episode 29. Deutsch-Jozsa Algorithm

Episode 30. Bernstein-Vazirani Algorithm

Episode 31. Simon's Algorithm

Statement of the Problem

1. We are given a function $f:\{0,\ 1\}^n \to \{0,\ 1\}^n$ and a secret string s of n bits.
2. For all $x,\ y \in \{0,\ 1\}^n$, $f(x) = f(y)$ if and only if $y = x \oplus s$. Note that f is either one-to-one ($s = 0$) or two-to-one ($s \neq 0$).
3. The task is to find the secret string s with as few queries to function f as possible.

Classically, one needs queries to $f(x)$ with up to $2^{n-1} + 1$ different inputs.

Quantum Implementation

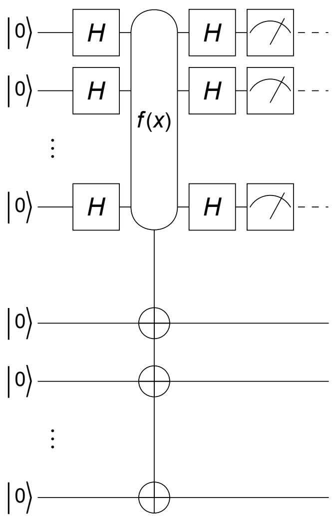


Figure 1. A quantum circuit to implement Simon's algorithm.

The quantum circuit in Figure 1 summarizes Simon's algorithm. The first `Hadamard` gate on the native register transforms the input state of the whole system as

$$|0\rangle \otimes |0\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0\rangle.$$

The quantum oracle makes a copy of the image $\left|f(x)\right\rangle$ of the state $\left|x\right\rangle$ of the native register to the ancillary register, and leads to

$$\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}\left|x\right\rangle\otimes\left|f(x)\right\rangle.$$

Finally, the second set of Hadamard gates on the native register maps the above state into

$$\sum_{y=0}^{2^n-1}\left|y\right\rangle\otimes\frac{1}{2^n}\sum_{x=0}^{2^n-1}\left|f(x)\right\rangle(-1)^{x\cdot y}.$$

The measurement on the native register yields an n-bit string y.

The probability for a particular string y is determined by the squared norm,

$$P_y=\left\langle\psi_y\middle|\psi_y\right\rangle,$$

of the y-dependent state $\left|\psi_y\right\rangle$ of the ancillary register

$$\left|\psi_y\right\rangle:=\frac{1}{2^n}\sum_{x=0}^{2^n-1}\left|f(x)\right\rangle(-1)^{x\cdot y}.$$

For s = 0, function f is one - to - one.

$$\left|\psi_y\right\rangle:=\frac{1}{2^n}\sum_{x=0}^{2^n-1}\left|f(x)\right\rangle(-1)^{x\cdot y}=\frac{1}{2^n}\sum_{z=0}^{2^n-1}\left|z\right\rangle(-1)^{f^{-1}(z)\cdot y}.$$

$$P_y=\frac{1}{2^n}$$

- For the analysis of the quantum circuit, see Section 4.2.4 of the Quantum Workbook (2022, 2023) or the Q3 tutorial “Simon’s Algorithm”.

Examples

In[ ]:=
```
Let[Qubit, S]
Let[Complex, c]
```

Smaller System

Consider again a secrete bit string.

In[ ]:=
```
string = {1, 1};
```

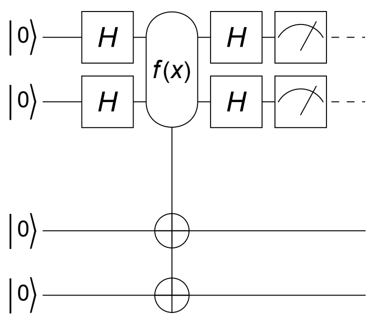
Consider a two-to-one function obeying the rule (specified in Simon’s problem).

In[ ]:=
```
Clear[f];
f[{0, 0}] = f[{1, 1}] = {0, 1};
f[{0, 1}] = f[{1, 0}] = {1, 1};
```

Here is an implementation of the corresponding quantum oracle.

In[ ]:=
```
cc = {1, 2};
tt = {3, 4};
all = Join[cc, tt];
qc = QuantumCircuit[Ket[S@all], S[cc, 6],
  Oracle[f, S@cc, S@tt], S[cc, 6], Measurement[S[cc, 3]],
  "Invisible" → S@{2.5}]
```

Out[ ]=



In[ ]:=
```
out = ExpressionFor[qc]
result = Readout[S[cc, 3]]
```

Out[ ]=

$$\frac{\left|1_{S_1} 1_{S_2} 0_{S_3} 1_{S_4}\right\rangle}{\sqrt{2}} - \frac{\left|1_{S_1} 1_{S_2} 1_{S_3} 1_{S_4}\right\rangle}{\sqrt{2}}$$

Out[ ]=

{1, 1}

In[ ]:=
```
mat = Table[ExpressionFor[qc]; Readout[S[cc, 3]], {2}]
```

Out[ ]=

{{0, 0}, {1, 1}}

Larger System

Now, let us examine a lager system. Suppose that we are given a secrete bit string.

In[ ]:=
```
string = {1, 1, 0};
```

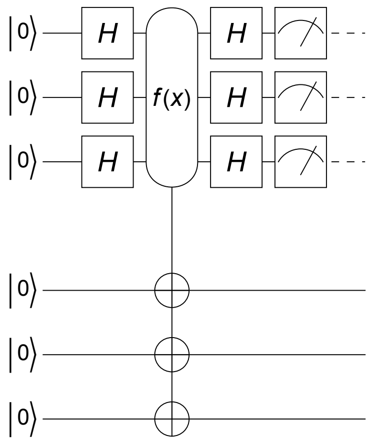
This is a function consistent with the above secrete bit string.

In[ ]:=
```
Clear[f];
f[0] = f[6] = 3;
f[1] = f[7] = 7;
f[2] = f[4] = 4;
f[3] = f[5] = 1;
```

Here is an implementation of the corresponding quantum oracle.

In[ ]:=
```
cc = {1, 2, 3};
tt = {4, 5, 6};
all = Join[cc, tt];
qc1 = QuantumCircuit[Ket[S@all], S[cc, 6], Oracle[f, S@cc, S@tt], S[cc, 6]];
qc2 = QuantumCircuit[qc1, Measurement[S[cc, 3]],
  "Invisible" → S@{3.5}]
```

Out[ ]=



This is one way to get the measurement outcome.

In[ ]:=
```
out = ExpressionFor[qc2]
result = Readout[S[cc, 3]]
```

Out[ ]=

$$\frac{1}{2}\left|1_{S_1}1_{S_2}1_{S_3}0_{S_4}0_{S_5}1_{S_6}\right\rangle + \frac{1}{2}\left|1_{S_1}1_{S_2}1_{S_3}0_{S_4}1_{S_5}1_{S_6}\right\rangle - \frac{1}{2}\left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}0_{S_5}0_{S_6}\right\rangle - \frac{1}{2}\left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}1_{S_5}1_{S_6}\right\rangle$$

Out[ ]=

{1, 1, 1}

To make repeated measurements, it is more efficient to first compute the state just before the measurement.

In[ ]:=
```
new = ExpressionFor[qc1];
```

Now we perform the measurement repeatedly.

In[ ]:=
```
data = Table[Measurement[S[cc, 3]]@new; Readout[S[cc, 3]], {12}];
data // TableForm
```

Out[ ]//TableForm=

1	1	1
1	1	1
1	1	0
0	0	0
0	0	0
1	1	0
1	1	1
0	0	1
0	0	0
1	1	1
0	0	1
1	1	1

As two linearly independent vectors (bit strings), we choose these:

In[ ]:=
```
mat = {{1, 1, 0}, {0, 0, 1}}
```

Out[ ]=
```
{{1, 1, 0}, {0, 0, 1}}
```

Then, the linear equation, `mat.ss=0 (mod 2)`, for the Boolean variables `ss:={s1,s2,s3}` is given by the following, which agrees with the given secrete bit string.

In[ ]:=
```
ss = {1, 1, 0}
```

Out[ ]=
```
{1, 1, 0}
```

In[ ]:=
```
Mod[mat.ss, 2]
```

Out[ ]=
```
{0, 0}
```

Summary

Keywords

- Oracle
- Decision making
- Simon’s problem

Functions

- `Oracle`

Related Links

- Section 4.2 of the Quantum Workbook (2022, 2023).

- Tutorial: Simon's Algorithm