Week 3 : Oracle Based Algorithms

Contents

1 Introduction

An oracle is a "black box" containing information about an unknown binary string $X = X_1X_2\cdots X_N$. Accessing the oracle with the input i will provide us with the output bit X_i. However, given an arbitrary oracle, we do not have any information of the inner workings of the oracle and so we say that that oracle contains or hides the information about X (hence the name "black box"). An oracle is generally represented as O. In the quantum setting, an oracle containing an unknown string $X = X_1X_2\cdots X_N$ acts as follows:

$$O\left|i\right\rangle\left|a\right\rangle \longrightarrow \left|i\right\rangle\left|a \oplus X_i\right\rangle.$$

We can observe that an oracle is in essence a linear operator (just like the quantum gates). If an oracle is queried with the superposition $\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\left|i\right\rangle\left|a\right\rangle$ as input, then the action of the oracle is given by

$$O\Big(\tfrac{1}{\sqrt{N}}\textstyle\sum_{i=0}^{N-1}\left|i\right\rangle\left|a\right\rangle\Big) \longrightarrow \tfrac{1}{\sqrt{N}}\textstyle\sum_{i=0}^{N-1}\left|i\right\rangle\left|a \oplus X_i\right\rangle.$$

If we set the ancilla qubit $\left|a\right\rangle = \left|-\right\rangle = \frac{\left|0\right\rangle-\left|1\right\rangle}{\sqrt{2}}$ for an input, then the oracle acts as

$$O_f\left|i\right\rangle\left|-\right\rangle \longrightarrow (-1)^{X_i}\left|i\right\rangle\left|-\right\rangle.$$

i.e., the queried bit is kicked back as a phase. This is called "Phase Kickback". Depending on the application, an oracle is implemented either as bit flip operator or as a phase flip operator.

Now say we have an n-bit input 1-bit output Boolean function f. Is it possible to somehow encode f in an oracle? Indeed we can. Given an f, we would have the complete description of the function f if we knew the values of the function at all 2^n inputs. So in order for an oracle to represent the function f, the oracle should contain information about all the 2^n function output values. Consequently, we can say that an oracle O corresponds to the function f if the oracle O contains the string $X = f(0)f(1)\cdots f(2^n-1)$ where $f(i)$ is the output of the function f when the binary string representation of i is provided as input.

In quantum computing, we have a large set of problems based on oracles and we will look into a few such problems in the later sections. In most of the oracle based problems, the ultimate objective is to compute a certain function, say $F(X)$ of the unknown string X with the least number of queries to the oracle.

2 Implementation of Oracles

When we introduced oracle, we asserted that the inner functioning of the oracles are not known when we are provided with an arbitrary oracle. But what if we want to construct our own oracle for a function or a string? A closer look at the oracles tells us that oracles are black boxes that mark (through bit flip or phase flip) those items with certain properties. For example, an oracle corresponding to a Boolean function f, marks those inputs x for which $f(x) = 1$. So an oracle is essentially a quantum circuit that implements a bit flip (or a phase flip) subject to certain conditions. The circuit corresponding to the oracle encoding a function f is denoted by U_f.

Let us take, for example, the 2-bit Boolean function $f(x) = x_1 \cdot x_2$ which is the two bit AND function. When an oracle implements the 2-bit AND function, the oracle should perform a bit flip on the ancilla bit only when the input is the state $|11\rangle$. Constructing a quantum circuit that flips the ancilla bit, say $|b\rangle$, when the input is $|11\rangle$ is quite straight forward. It is just a CCNOT gate (this is not a single control CNOT gate but a double control gate) where the control bits are the two input bits and the target bit is the ancilla bit.

In a similar fashion, we can construct oracles corresponding to any arbitrary function or string. But, as the size of the string or the function increases, the construction of oracles becomes harder. So we resort to algorithms like Quine-McCluskey algorithm for minimizing the Boolean functions and then use the minimized Boolean function to construct the oracle. However, this is beyond the scope of this material.

3 Deutsch Algorithm

Consider the following problem. Suppose we are provided with an oracle O_f of a Boolean function $f : \{0,1\} \longrightarrow \{0,1\}$. A function g is said to be constant if $g(x) = c$ for $c \in 0, 1$ for all input x and balanced if $g(x) = 0$ for exactly half of the 2^n inputs x and $g(x) = 1$ for the rest of the inputs. Say our objective is to find if the given function f is balanced or constant. Classically, what is the minimum number of queries that is required to determine if the function is balanced or constant? (Think before you read ahead.) It is exactly two queries. We have to compute both $f(0)$ and $f(1)$ by making one query for each and if both values are same then the function f is constant else the function is balanced. But it just takes one query for a quantum computer to find that! This is done by the Deutsch algorithm.

Even though Deutsch algorithm is not of much practical use, it serves as a very good tool to understand quantum parallelism and quantum interference. In order to determine if an 1-bit Boolean function is balanced or constant, we are actually not interested in the individual f values but the value of $f(0) \oplus f(1)$ which is a global property of f. Deutsch algorithm intelligently exploits this condition and uses quantum interference to obtain such a global property of f in just a single query.

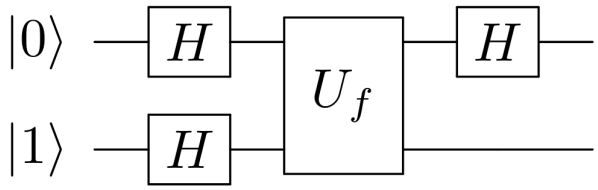


Figure 1: Circuit of Deutsch Algorithm

Deutsch algorithm is implemented using the quantum circuit provided in Figure 1.

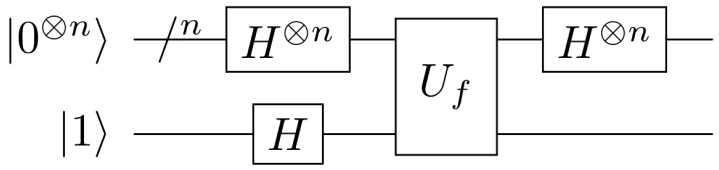


Figure 2: Circuit of Deutsch Jozsa Algorithm

The state of the qubits after application of each layer of the circuit is as follows:

$$
\begin{aligned}
|0\rangle\,|1\rangle &\xrightarrow{H\otimes H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|-\rangle \\
&\xrightarrow{U_f} \frac{1}{\sqrt{2}}((-1)^{f(0)}\,|0\rangle + (-1)^{f(1)}\,|1\rangle)\,|-\rangle \\
&\xrightarrow{H\otimes I} \frac{1}{2}\Big[\Big((-1)^{f(0)} + (-1)^{f(1)}\Big)\,|0\rangle + \Big((-1)^{f(0)} - (-1)^{f(1)}\Big)\,|1\rangle\Big]\,|-\rangle
\end{aligned}
$$

We can see from the above equation that the amplitude of $|0\rangle$ becomes 0 when the function f is balanced and 1 when f is constant. So we obtain $|0\rangle$ on measurement with probability 1 if f is constant and $|1\rangle$ with probability 1 if f is balanced. In the next section, we will see an extension to this algorithm which solves the problem of determining if an n-bit Boolean function f is balanced or constant given a prior promise that f is either balanced or constant.

4 Deutsch-Jozsa Algorithm

Deutsch-Jozsa algorithm (or shortly DJ algorithm) is an extension of Deutsch algorithm to any n-bit Boolean function. Similar to the Deutsch algorithm, we are provided with an oracle of an n-bit Boolean function f and the promise that f is either balanced or constant. The problem is to determine if f is balanced or constant. Before getting into the implementation of DJ algorithm, let us first consider the case of classically solving the this problem. For any n-bit Boolean function f there are 2^n possible functional values $f(i)$. Since we were given the promise that f is either constant or balanced, we need atleast $(2^n/2)+1$ function values to determine the condition of f. Classically in order to obtain these values we need atleast $\frac{2^n}{2}+1$ queries to the function f. So classically we need number of queried exponential in n. However, we will now show that DJ algorithm makes just one query to determine the condition of f.

The implementation of DJ algorithm as a quantum circuit is provided in Figure 2. The evolution of the input state post implementation of each layer of circuit is given below.

$$\left|0^{\otimes n}\right\rangle |1\rangle \xrightarrow{H^{\otimes n}\otimes H} \sum_x |x\rangle |-\rangle$$
$$\xrightarrow{U_f} \sum_x (-1)^{f(x)} |x\rangle |1\rangle$$
$$\xrightarrow{H^{\otimes n}\otimes I} \sum_y \left[\sum_x (-1)^{f(x)\oplus x.y}\right] |y\rangle$$

where $x.y = x_1y_1 + x_2y_2 + \cdots + x_ny_n$.

It remains to prove the correctness of the algorithm. Now, we have two cases.

Case (i) : Let us assume that the function f is balanced. Now, the probability of obtaining $|0\rangle$ on measurement is $Pr[|0\rangle] = \frac{1}{2^n}\left[\sum_x (-1)^{f(x)\oplus x.y}\right]^2 = \frac{1}{2^n}\left[\sum_x (-1)^{f(x)}\right]^2$ since we have $x \cdot y = 0$ for all n-bit string x for $y = 0^{\otimes n}$. Since f is balanced, we have exactly $2^n/2$ strings x such that $f(x) = 0$ and exactly $2^n/2$ strings x such that $f(x) = 1$. Hence the probability of obtaining $|0\rangle$ is $Pr[|0\rangle] = \frac{1}{2^n}\left[\sum_x (-1)^{f(x)}\right]^2 = 0$.

Case (ii) : Next let us assume that the function f is constant. Then, the probability of obtaining $|0\rangle$ on measurement is given as $Pr[|0\rangle] = \frac{1}{2^n}\left[\sum_x (-1)^{f(x)}\right]^2$. Now, since the function f is constant $Pr[|0\rangle] = \frac{1}{2^n}\left[\sum_x (-1)^{f(x)}\right]^2 = \frac{1}{2^n} \cdot 2^n = 1$.

We can clearly observe that, on measurement, we obtain $|0\rangle$ with probability 1 if f is constant and with probability 0 if f is balanced.

5 Grover's Algorithm and Amplitude Amplification

Grover's algorithm is one of the most popular quantum algorithms that is the backbone of many other quantum algorithms. It is a method that offers polynomial speed up over best known classical algorithms for solving a wide class of important problems. One such problem is the unordered search. It is well known that the best known classical algorithm for unordered search makes $O(N)$ queries to find an item of interest, also called a solution, from a list of N items. But using Grover's iterations, the same can be obtained in just $O(\sqrt{N})$ queries. At the first look $\sqrt{N}$ speedup may not look very big. But as n grows, the difference is observed significantly. For instance, for a list of size $N = 20$, the Grover's algorithm, also known as the quantum search algorithm and Grover's search, makes about 5 queries which is not significantly less. But say for a list of size $N = 100000$, the Grover's search needs just 100 calls to the oracle to obtain the solution. This is a huge improvement over 100000 calls made by the best known classical algorithm. However, Grover's algorithm is probabilistic, i.e, the algorithms gives the correct solution with some probability greater than 1/2. Although Grover's search is not as breathtaking as other quantum algorithms with exponential speedup like Deutsch-Jozsa algorithm and Shor's factoring algorithm, its extensive applicability in the field of search problems makes it very interesting and noteworthy.

Now let us formally define the quantum search algorithm. Let n be such that $N = 2^n$. Let us assume that we have an oracle, U_f, that marks the solution of our interest, i.e, it acts as

$$U_f |x\rangle |a\rangle = |x\rangle |a \oplus f(x)\rangle$$

where $f(x) = 1$ is x is a solution and $f(x) = 0$ otherwise. Let $|\phi\rangle = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} |x\rangle$ be the superposition of all possible query values (which are essentially the items in the given list). Define the states

$$|good\rangle = \frac{1}{\sqrt{t}} \sum_{x:f(x)=1} |x\rangle \ , \ |bad\rangle = \frac{1}{\sqrt{N-t}} \sum_{x:f(x)=0} |x\rangle$$

as the superposition of good and bad states respectively. Then we can write

$$|\phi\rangle = \sin(\theta) |good\rangle + \cos(\theta) |bad\rangle$$

where $\theta = sin^{-1}(\sqrt{t/N})$. Let us assume, for simplicity, that we have exactly one solution, say u. Then we have $|good\rangle$ and $|bad\rangle$ states as

$$|good\rangle = |u\rangle \ , \ |bad\rangle = \frac{1}{\sqrt{N-1}} \sum_{x:f(x)=0} |x\rangle$$

and the query register as

$$|\phi\rangle = \frac{1}{\sqrt{N}} |u\rangle + \sqrt{\frac{N-1}{N}} \sum_{x:f(x)=0} |x\rangle.$$

We can observe that the probability of obtaining the solution state $|u\rangle$ on measuring $|\phi\rangle$ is $\frac{1}{N}$. The goal of the quantum search algorithm is to increase the probability of obtaining the good state $|u\rangle$ from $\frac{1}{N}$ to close to 1. This is achieved using the operator $G = H^{\otimes n} U_{\overline{0}} H^{\otimes n} U_f$ which is called the *Grover iterate* where $U_{\overline{0}}$ operates as follows

$$U_{\overline{0}} |x\rangle = \begin{cases} -|x\rangle & \text{if } x \neq 0 \\ |x\rangle & \text{if } x = 0 \end{cases}$$

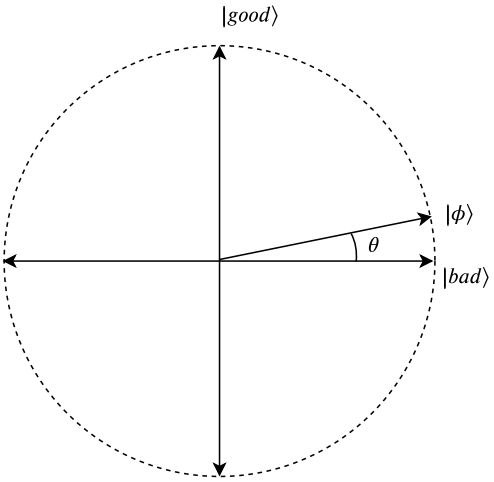
With a closer look at the Grover iterate, we can observe that it can be written as

$$H^{\otimes n} U_{\overline{0}} H^{\otimes n} U_f = H^{\otimes n} (2 |0\rangle \langle 0| - I) H^{\otimes n} U_f = (2 |S\rangle \langle S| - I) U_f = U_{Ref_S} U_f$$

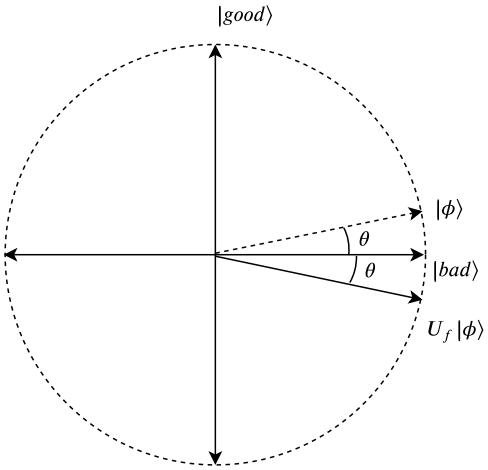
where state $|S\rangle = H^{\otimes n} |0^{\otimes n}\rangle$ is the equal superposition of all the input states. The operator $U_{Ref_S} = 2 |S\rangle \langle S| - I$ is essentially a reflection about the state $|S\rangle$. Again the oracle U_f is a reflection about the bad state. Consequently, the Grover iterate is an operator of two reflections U_{Ref_S} and U_f. Now, the quantum search algorithm is given as follows:

1. Start with the state $|0^{\otimes n}\rangle$.
2. Apply the Hadamard gate on all the qubits to obtain the state $|S\rangle$.
3. For $k = \lceil \frac{\pi}{4} - \frac{1}{2} \rceil$ times perform:
   (a) Apply the operator U_f to reflect about the state $|bad\rangle$.
   (b) Apply the operator U_{Ref_S} to reflect about the state $|S\rangle$.
4. Measure the first register to check if the output is the solution.

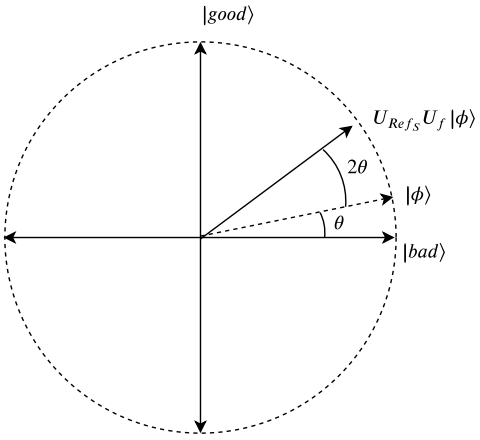
To have a better understanding of the algorithm and the chosen value of k, let us look into the geometric interpretation of the quantum search algorithm.



(a) Initial state



(b) Reflection about $|bad\rangle$



(c) Reflection about $|S\rangle$

Figure 3: Grover's Iteration

Geometric Interpretation

Suppose that we have a two dimensional plane spanned by the states $|good\rangle$ and $|bad\rangle$. The states $|good\rangle$ and $|bad\rangle$ are orthogonal to each other. The state $|S\rangle = H^{\otimes n}|0^{\otimes n}\rangle$ will be a superposition of $|good\rangle$ and $|bad\rangle$ as

$$|S\rangle = sin(\theta)|good\rangle + cos(\theta)|bad\rangle \text{ where } \theta = sin^{-1}(\tfrac{1}{\sqrt{N}})$$

as mentioned earlier. The representation of the state $|S\rangle$ on the plane spanned by $|good\rangle$ and $|bad\rangle$ is given in the Figure 3a.

On applying the oracle U_f, a negative phase is acquired only by the $|good\rangle$ state and so the resultant state $|\phi'\rangle = U_f|S\rangle$ is a reflection of the state $|S\rangle$ about the state $|bad\rangle$ as in Figure 3b. The state $|\phi'\rangle$ is at an angular distance of θ from $|bad\rangle$ similar to $|S\rangle$.

Finally, on applying the operator U_{Ref_S} on $|\phi'\rangle$, the state $|\phi'\rangle$ gets reflected over the state $|S\rangle$ as in Figure 3c. Now, since the angle between $|S\rangle$ and $|\phi'\rangle$ was 2θ, on final reflection the resultant state $|\phi''\rangle = U_{Ref_S}|\phi'\rangle$ makes an angle 3θ with the state $|bad\rangle$. Hence one Grover iteration takes the state that makes an angle θ with the bad state and transforms it into a state that makes an angle 3θ with the bad state with just one call to the oracle U_f. There is an increase of 2θ in a single Grover's iteration. So the probability of obtaining the good state gets amplified from $sin^2(\theta)$ to $sin^2(3\theta)$.

When the Grover's iteration is applied again, we can see that the angle between the resultant state and the bad state increases from 3θ to 5θ. So in general, the angle gain on applying Grover's iteration k-times is $(2k+1)\theta$. The ultimate goal of the quantum search algorithm is to transform the initial state into a final state which is outputs $|good\rangle$ with probability 1. In terms of θ, the angle difference between the final state and the bad state should be $\pi/2$. However since the angle gained is an integral multiple of θ itself and since $\pi/2$ is not necessarily a multiple of θ, we obtain a final state that is as close to $|good\rangle$ as possible, i.e, $(2k+1)\theta \approx \pi/2$. The optimal number of Grover's iterations required to obtain such a final state is left as an exercise to the reader. On solving the exercise, we will see that a final state close to the good state is obtained with just $O(\sqrt{N})$ calls to the oracle.

A natural extension to the Grover's search algorithm is the 'Amplitude Amplification' algorithm. Unlike Grover's search which uses an equal superposition of all possible states as the initial state, the amplitude amplification algorithm works for any arbitrary initial state $\mathcal{A}|0^{\otimes n}\rangle$ for any algorithm $\mathcal{A}$. In essence, Grover's search is the specialized case of amplitude amplification algorithm where $\mathcal{A} = H^{\otimes n}$.

The amplitude amplification algorithm is given as follows:

1. Start with the state $|0^{\otimes n}\rangle$.
2. Apply $\mathcal{A}$ on $|0^{\otimes n}\rangle$ to obtain $\mathcal{A}|0^{\otimes n}\rangle$.
3. For $k = \left\lceil \frac{\pi}{4} - \frac{1}{2} \right\rceil$ times perform:
   (a) Apply the operator U_f to reflect about the state $|bad\rangle$.
   (b) Apply the operator $U_{Ref_{\mathcal{A}}}$ to reflect about the state $|S\rangle$.
4. Measure the first register to check if the output is the solution.

where $U_{Ref_{\mathcal{A}}} = \mathcal{A}U_{\overline{0}}\mathcal{A}^{-1}$ is the reflection operator that reflects a state about $\mathcal{A}|0^{\otimes n}\rangle$.

Similar to the Grover's search algorithm, the number of queries required by amplitude amplification algorithm to obtain a final state close to the good state is $O(\sqrt{N})$ where $N = 2^n$.

6 Simon's Period-Finding Algorithm

Assume that we are given an n-bit input k-bit output Boolean function f. Say, we are also given a promise that the function f is such that for any two n-bit inputs x and y, $f(x) = f(y)$ if and only if $x = y \oplus s$ for some $s \in \{0,1\}^n$. Is it possible to determine the shift s? If possible, how many queries to the function f are required to determine s? This problem is called the Simon's problem.

Let us first find the number of queries required classically. We know that we need to query the function f once per input. Now we need to find two input strings x and y which have the same functional values, i.e, $f(x) = f(y)$. Since we have no information about the shift s, in the worst case we would have to query for atleast one more than half the number of total inputs, i.e, we need to make $\frac{2^n}{2} + 1 = O(2^n)$ number of queries to the function f so that by pigeon-hole principle we will get a colliding pair. We can see that the number of queries is exponential in n. Even if we use birthday bounds, we would still need $O(2^{n/2})$ queries to f before obtaining a collision pair which is still exponential in n.

Can we do better in a quantum setting? Turn outs we can do better in a quantum setting. But how much better? Is it just quadratically faster than the best known classical algorithms or even faster than that? The successive discussion on Simon's algorithm will help up answer these questions.

Simon's algorithm efficiently uses the exclusively quantum concepts of constructive and destructive interference to solve the Simon's problem in $O(n)$ queries to the oracle of the function f. Despite the fact that solving this problem is not of immediate practical use, Simon's algorithm shows that there does exist a class of problems for which the quantum algorithms can provide solutions exponentially faster than the best known classical algorithms. Simon's algorithm uses a circuit similar to that of the Deutsch algorithm. One of the prime differences between them is that the Deutsch algorithm processes Boolean function with 1-bit output whereas Simon's algorithm processes Boolean functions with arbitrary output size pertaining to certain conditions (you will derive the conditions as a part of the exercise). The quantum circuit corresponding to Simon's algorithm is presented in Figure 4.

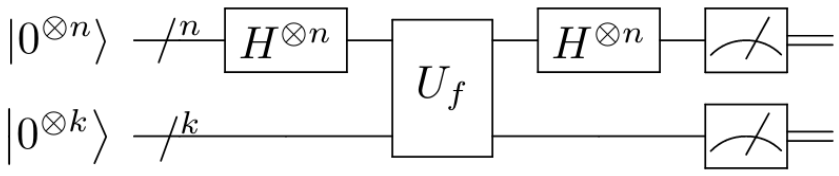


Figure 4: Circuit implementing Simon's Algorithm

The evolution of the states after implementation of each of the layers is as follows:

$$\begin{aligned}
\left|0^{\otimes n}\right\rangle \left|0^{\otimes k}\right\rangle &\xrightarrow{H^{\otimes n}\otimes I} \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \left|0^{\otimes k}\right\rangle \\
&\xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle \\
&\xrightarrow{H^{\otimes n}\otimes I} \frac{1}{2^n} \sum_x \sum_y (-1)^{x\cdot y} |y\rangle |f(x)\rangle \\
&= \sum_y |y\rangle \otimes \Big(\frac{1}{2^n} \sum_x (-1)^{x\cdot y} |f(x)\rangle \Big)
\end{aligned}$$

Now, let us analyze the probability of obtaining some $|y\rangle$ in cases.
Case i : Let $s = 0 \otimes n$. Then the probability of obtaining any particular $|y\rangle$ is given by

$$\begin{aligned}
Pr(|y\rangle) &= \Big|\Big| \sum_x (-1)^{x\cdot y} |f(x)\rangle \Big|\Big|^2 \\
&= \Big|\Big| \sum_x (-1)^{x\cdot y} |x\rangle \Big|\Big|^2 \text{ (since all } f(x) \text{ are distinct)} \\
&= \frac{1}{2^n}
\end{aligned}$$

So when the shift $s = 0^{\otimes n}$, then we observe all the 2^n strings $|y\rangle$ with equal probability.
Case ii : Let $s \neq 0^{\otimes n}$. Then the probability of obtaining any particular $|y\rangle$ is given by

$$Pr(|y\rangle) = \Big|\Big| \frac{1}{2^n} \sum_x (-1)^{x\cdot y} |f(x)\rangle \Big|\Big|^2$$

Let A be the set of all the unique $f(x)$s. Then we can write

$$Pr(|y\rangle) = \Big|\Big| \frac{1}{2^n} \sum_{z\in A} \Big((-1)^{x_1\cdot y} + (-1)^{x_2\cdot y} \Big) |z\rangle \Big|\Big|^2$$

where $x_1 = x_2 \oplus s$ and $f(x_1) = f(x_2) = z$. Since, $x_1 = x_2 \oplus s$ and $(x_1 \oplus s)\cdot y = (x_1 \cdot y) \oplus (s \cdot y)$.

$$\begin{aligned}
Pr(|y\rangle) = 0 &= \Big|\Big| \frac{1}{2^n} \sum_{z\in A} \Big((-1)^{x_1\cdot y} + (-1)^{(x_1\oplus s)\cdot y} \Big) |z\rangle \Big|\Big|^2 \\
&= \Big|\Big| \frac{1}{2^n} \sum_{z\in A} (-1)^{x_1\cdot y} \Big(1 + (-1)^{s\cdot y} \Big) |z\rangle \Big|\Big|^2
\end{aligned}$$

We can see that for y such that $s \cdot y = 1$, we have $Pr(|y\rangle) = 0$. This is a case of destructive interference. For any y such that $s \cdot y = 0$, we have $1 + (-1)^{y \cdot s} = 2$ which implies $Pr(|y\rangle) = \frac{2}{2^n} = \frac{1}{2^{n-1}}$. This is a case of constructive interference. On measuring the final state of the circuit in Figure 4, we obtain $|y\rangle$ such that $y \cdot s = 0$. We then perform $O(n)$ measurements to obtain $n-1$ strings $y_1, y_2, ..., y_n$ such that $y_i \cdot s = 0$ for all i. This gives us a system of n linear equations with n unknowns. If $y_1, y_2, ..., y_n$ are linearly independent then we obtain the unknown string s by solving this system of linear equations. The probability of obtaining such a set of linearly independent y_i s is atleast $\frac{1}{4}$. Hence the overall query complexity of finding the unknown shift s is $O(n)$.

In this chapter we have discussed about oracles and some important oracle based algorithms. Many algorithms use these algorithms as subroutines to solve many other computational problems. Work has also been done to generalize the discussed algorithms. An interested reader can explore quantum algorithms for Walsh spectrum, forrelation problem and hidden shit problems for further reading.

7 Exercises

1. Verify that if the ancilla bit is set as $|a\rangle = |-\rangle$, then the oracle kicks back the queried bit as phase.

2. Let f be a 4-bit Boolean function given by $f(x) = (x_1 \cdot x_2) \oplus (x_3 \cdot x_4)$. Construct a quantum oracle (a circuit) that when given an input state $|x\rangle\,|b\rangle$ flips the ancilla qubit $|b\rangle$ if and only if $f(x) = 1$.

3. Given an oracle for an n-bit Boolean function O_f, construct a circuit such that on providing $|0^n\rangle$ and $|a\rangle$ as input, outputs $|y\rangle$ with probability $Pr(|y\rangle) = \left[\sum_x (-1)^{f(x \oplus a)} (-1)^{(x \oplus a) \cdot y}\right]^2$.

4. Compute the maximum number of amplitude amplification iterations required for a state $|\psi\rangle = sin\theta\,|0\rangle + cos\theta\,|1\rangle$ to become $|\psi'\rangle = sin\theta'\,|0\rangle + cos\theta'\,|1\rangle$ where $sin^2\theta' \approx 1$ for some small initial θ. (Hint: Remember that k iterations of amplitude amplification will change the angle θ to $(2k+1)\theta$).

5. Let A be an algorithm whose matrix representation is given by

$$A = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

   and let $|\psi\rangle = A\,|0\rangle$. Also let the good state be $|1\rangle$. Show that on applying $-(2\,|\psi\rangle\,\langle\psi| - I)S_0$ on $|\psi\rangle$ (where S_0 acts on $|0\rangle$ as $S_0\,|0\rangle = -\,|0\rangle$ and trivially on $|1\rangle$) amplifies the probability of obtaining $|1\rangle$ from $sin^2\theta$ to $sin^2(3\theta)$.

6. Let $f : \{0,1\}^n \longrightarrow \{0,1\}^k$ be a Boolean function. We know that if the function f is such that $f(x) = f(y)$ if and only if $x = y \oplus s$ for some shift s, then Simons's algorithm can be used to obtain the shift s. What are the values of k for which the function f can be of the above form?

7. *We know that an oracle O can be implemented as a phase kickback operator as $O|x\rangle|-\rangle \longrightarrow (-1)^{f(x)}|x\rangle|-\rangle$. The first-order derivative of an n-bit Boolean function f at a point $a \in \{0,1\}^n$ is defined as

$$\Delta f_a(x) = f(x \oplus a) \oplus f(x).$$

Given an oracle O_f, construct a quantum circuit to obtain a state such that the probability of obtaining the state $|y\rangle$ when measured equals the square of the Walsh coefficient of the first-order derivative of the function $f(x)$ at the point a, i.e.,

$$Pr(|y\rangle) = \frac{1}{2^n}\Big[\sum_x (-1)^{\Delta f_a(x)}(-1)^{x\cdot y}\Big]^2.$$