

II

QUANTUM ALGORITHMS I



In this chapter, we will introduce two quantum algorithms, the first one is called the Deutsch-Jozsa (DJ) algorithm, and the second one is called the Bernstein-Vazirani (BV) algorithm. But, before getting into quantum algorithms, it is essential to explain why in quantum we need **reversible** gates?

III.1 REVERSIBLE AND IRREVERSIBLE CLASSICAL GATES

Quantum time evolution is unitary \Rightarrow invertible \Rightarrow reversible.

In standard quantum mechanics the state is $|\psi\rangle$ (Hilbert space). The time evolution is

$$|\psi(t)\rangle = U(t) |\psi(0)\rangle. \quad (\text{III.1})$$

$U(t)$ is unitary:

$$U^\dagger U = UU^\dagger = I. \quad (\text{III.2})$$

Unitary operators are one-to-one and onto (and satisfy $U^\dagger = U^{-1}$), so they never lose information.

IMPLICATIONS FOR QUANTUM CIRCUITS

In an ideal quantum computer, each gate implements a short-time evolution of a closed quantum system, and therefore must be represented by a unitary operator (i.e., a unitary matrix in a chosen basis).

1. Every quantum gate has an inverse gate (its inverse is its adjoint).
2. Consequently, the overall unitary implemented by a circuit is reversible: in principle, one can apply the inverse circuit to undo the computation.

Thus, reversibility is built into the physics of ideal (closed-system) quantum dynamics.

By contrast, ordinary classical logic gates such as AND are not reversible: they are many-to-one maps and therefore discard information. To implement classical logic within a quantum circuit, we must use only reversible operations (e.g., by embedding irreversible gates into larger reversible ones using ancilla bits and gates like NOT, CNOT, and Toffoli).

IRREVERSIBLE GATE → REVERSIBLE: HOW?

Let's explain this with an example of a classical gate, let's look at the classical 2-bit AND:

$$(a, b) \longmapsto a \wedge b.$$

With the Truth table:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

This map is many-to-one, so it is not reversible.

Quantum Computing

The trick is to keep the inputs and add an extra bit. So, we keep the original bits a, b and add one extra target bit c (an **ancilla**). And we define the reversible gate as:

$$F(a, b, c) = (a, b, c \oplus (a \wedge b)). \quad (\text{II.3})$$

If c is initialized to 0, then

$$F(a, b, 0) = (a, b, a \wedge b). \quad (\text{II.4})$$

Below is the complete Table (for all 8 possible cases for the three bits), and we check reversibility: every input has a distinct output. In other words, we were able to use an AND gate (the logic) with the help of adding an extra bit.

a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Now every input has a distinct output \Rightarrow reversible.

II.2 DEUTSCH ALGORITHM

Deutsch's algorithm is a simplified version of DJ algorithm. In Deutsch's algorithm, the problem set up is as follows: we have a “black box” that takes two bits, does something with them, and outputs the two bits. For example, we feed $(0, 0)$ and we get $(0, 1)$. Each time we do this, we say that we are *querying* the black box. The goal is to learn about the possible function being computed by the black box by querying the box as few times as possible.

In Deutsch's problem, the black box implements one of the following four functions:

$$\begin{aligned} f_1(x, y) &= (x, y), \\ f_2(x, y) &= (x, \bar{y}), \quad (\text{negation of } y) \\ f_3(x, y) &= (x, x \oplus y), \\ f_4(x, y) &= (x, x \oplus \bar{y}), \end{aligned} \tag{II.5}$$

where \oplus is XOR (addition mod 2):

$$0 \oplus 0 = 1 \oplus 1 = 0, \quad 0 \oplus 1 = 1 \oplus 0 = 1 \pmod{2}.$$

Define the two sets

$$S_1 = \{f_1, f_2\}, \quad S_2 = \{f_3, f_4\}.$$

Problem: Determine whether the function implemented by the black box is in S_1 or in S_2 .

CLASSICAL QUERY COUNT (WHY I QUERY IS NOT ENOUGH)

Suppose we query the function with $(0, 0)$:

$$\text{If } f \in S_1 : \quad f_1(0, 0) = (0, 0), \quad f_2(0, 0) = (0, 1),$$

$$\text{If } f \in S_2 : \quad f_3(0, 0) = (0, 0), \quad f_4(0, 0) = (0, 1).$$

So we are not able to distinguish between S_1 and S_2 with one classical query, and we need to query two times to classically distinguish between the two sets.

QUANTUM WAY: UNITARY (ORACLE) IMPLEMENTATIONS

To make a quantum black box, represent each f_i Eq.(II.5) as a 2-qubit unitary U_i acting on basis states $|x, y\rangle$.

Quantum Computing

For example, $f_2(x, y) = (x, \bar{y})$ flips only the second bit, so its unitary matrix is:

$$U_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \Rightarrow U_2 |x, y\rangle = |x, \bar{y}\rangle. \quad (\text{II.6})$$

The four unitaries (in the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$) are:

$$U_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad U_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad (\text{II.7})$$

$$U_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad U_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (\text{II.8})$$

Now we want to distinguish between the sets $\{U_1, U_2\}$ and $\{U_3, U_4\}$ with only *one* query.

DEUTSCH CIRCUIT (I-QUERY QUANTUM SOLUTION)

The schematic of a quantum circuit is shown in Fig II.1. The steps taken to build the circuit are:

1. Start in $|0\rangle \otimes |1\rangle$.
2. Apply Hadamard to each qubit:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \quad (\text{II.9})$$

3. Apply the oracle U_i (one black-box query).
4. Apply H again to both qubits.
5. Measure.

After the first Hadamards, the state is

$$\begin{aligned} |\Phi\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle). \end{aligned} \quad (\text{II.10})$$

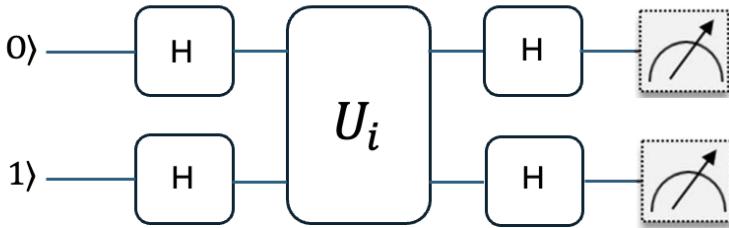


Figura II.1: Schematic of a quantum circuit for Deutsch's algorithm

Applying the oracle gives the following states:

$$|\psi_1\rangle = U_1 |\Phi\rangle = \frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle),$$

$$|\psi_2\rangle = U_2 |\Phi\rangle = \frac{1}{2} (-|00\rangle + |01\rangle - |10\rangle + |11\rangle),$$

$$|\psi_3\rangle = U_3 |\Phi\rangle = \frac{1}{2} (|00\rangle - |01\rangle - |10\rangle + |11\rangle),$$

$$|\psi_4\rangle = U_4 |\Phi\rangle = \frac{1}{2} (-|00\rangle + |01\rangle + |10\rangle - |11\rangle).$$

Now act with $H \otimes H$, where

$$H \otimes H = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \quad (\text{II.II})$$

Define $|X_i\rangle = (H \otimes H)U_i|\Phi\rangle$. Then:

$$\begin{aligned} |X_1\rangle &= (H \otimes H)U_1|\Phi\rangle = |01\rangle, & |X_2\rangle &= (H \otimes H)U_2|\Phi\rangle = -|01\rangle, \\ |X_3\rangle &= (H \otimes H)U_3|\Phi\rangle = |11\rangle, & |X_4\rangle &= (H \otimes H)U_4|\Phi\rangle = -|11\rangle. \end{aligned}$$

Hence, a measurement distinguishes the sets in **one query**:

$$\{U_1, U_2\} \longrightarrow |01\rangle, \quad \{U_3, U_4\} \longrightarrow |11\rangle. \quad (\text{II.I2})$$

So we can determine whether S_1 or S_2 was used with only one quantum query.

II.3 THE DEUTSCH-JOZSA ALGORITHM

Now that we have seen Deutsch's problem, we can move into our first, more serious algorithm, the the Deutsch–Jozsa problem.

DEUTSCH-JOZSA PROBLEM STATEMENT: CONSTANT vs. BALANCED

In the Deutsch–Jozsa (DJ) problem, there exist two classes of Boolean functions: *constant* functions and *balanced* functions.

Consider a function from n bits to 1 bit,

$$f : \{0, 1\}^n \rightarrow \{0, 1\}. \quad (\text{II.I3})$$

The function is **constant** if

$$f(x) = 0 \text{ for all } x \in \{0, 1\}^n, \quad \text{or} \quad f(x) = 1 \text{ for all } x \in \{0, 1\}^n.$$

(Equivalently, f is constant on all inputs.)

The function is **balanced** if it outputs 0 on half of the possible inputs and 1 on the other half. Equivalently speaking, there exists a subset $S \subseteq \{0, 1\}^n$ with

$$|S| = 2^n$$

such that

$$f(x) = 0 \quad \forall x \in S, \quad f(x) = 1 \quad \forall x \notin S.$$

QUESTION

How many queries could it take in the *worst case* to distinguish between **constant** and **balanced**? Classically, we know the worst-case scenario will be

$$|S| = 2^{n-1} + 1. \tag{II.14}$$

But can we solve it with Quantum Algorithms? To do so, first we define an Oracle as a unitary map:

$$U_f : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle. \tag{II.15}$$

Here $x \in \{0, 1\}^n$ and $y \in \{0, 1\}$.

We want to *query* U_f to determine whether $f(x)$ is **constant** or **balanced**.

We also introduce a trick called the Phase Kickback Trick, as explained in the next Section II.4. This trick is going to be used in the development of other algorithms as well. Moreover, as the Hadamard operator will be an occurring process, in Section II.5 we will summarize the operations within the Hadamard.

II.4 PHASE KICKBACK TRICK

We use the special state:

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{II.16}$$

Quantum Computing

Applying the oracle we have:

$$\begin{aligned}
 U_f |x\rangle |- \rangle &= U_f \left(|x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \right) \\
 &= \frac{1}{\sqrt{2}} (U_f |x\rangle |0\rangle - U_f |x\rangle |1\rangle)
 \end{aligned} \tag{II.17}$$

Which leads to:

$$\begin{aligned}
 U_f |x\rangle |0\rangle &= |x\rangle |f(x)\rangle \\
 U_f |x\rangle |1\rangle &= |x\rangle |1 \oplus f(x)\rangle = |x\rangle |\overline{f(x)}\rangle
 \end{aligned} \tag{II.18}$$

So:

$$\begin{aligned}
 U_f |x\rangle |- \rangle &= \frac{1}{\sqrt{2}} \left(|x\rangle |f(x)\rangle - |x\rangle |\overline{f(x)}\rangle \right) \\
 &= |x\rangle \otimes \frac{1}{\sqrt{2}} \left(|f(x)\rangle - |\overline{f(x)}\rangle \right)
 \end{aligned} \tag{II.19}$$

Considering the two cases:

- If $f(x) = 0$, then

$$U_f |x\rangle |- \rangle = |x\rangle |- \rangle$$

- If $f(x) = 1$, then

$$U_f |x\rangle |- \rangle = - |x\rangle |- \rangle$$

Conclusion:

$$U_f |x\rangle |- \rangle = (-1)^{f(x)} |x\rangle |- \rangle \tag{II.20}$$

—

II.5 HADAMARD OPERATOR

The Hadamard matrix can be written in the following way:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (\text{II.21})$$

In which when acting on the states $|0\rangle$ and $|1\rangle$ gives us:

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned}$$

The general qubit can be written as:

$$H|x\rangle = \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}} (-1)^{x \cdot y} |y\rangle, \quad (\text{II.22})$$

in which the general Hadamard is written as:

$$H = \frac{1}{\sqrt{2}} \sum_{x,y \in \{0,1\}} (-1)^{x \cdot y} |y\rangle \langle x|, \quad (\text{II.23})$$

and the tensor product form is:

$$H^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \langle x|$$

Example:

Applying Hadamard on the state zero gives us:

$$H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} |z\rangle \quad (\text{II.24})$$

Applying $H^{\otimes n}$ on both sides again gives us:

$$H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_z |z\rangle \right) = |0\rangle \quad (\text{II.25})$$

$$H^{\otimes n} H^{\otimes n} = I \quad \Rightarrow \quad \text{Applying twice returns original state.}$$

Now we can get back into solving the DJ algorithm, based on the explanation in Section II.4 we know:

If the second register is prepared in

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

then the oracle acts as a phase on the first register:

$$U_f(|x\rangle \otimes |-\rangle) = (-1)^{f(x)} |x\rangle \otimes |-\rangle.$$

QUERYING THE ORACLE IN SUPERPOSITION

Let us query U_f with the input

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |-\rangle. \quad (\text{II.26})$$

Then

$$U_f |\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \otimes |-\rangle. \quad (\text{II.27})$$

Since the second register $|-\rangle$ is unchanged, we can focus on the first register:

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle. \quad (\text{II.28})$$

IF f IS CONSTANT

If $f(x) = 0$ for all x , then

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle.$$

If $f(x) = 1$ for all x , then

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1) |x\rangle = -\frac{1}{\sqrt{2^n}} \sum_x |x\rangle.$$

Hence, for a constant function,

$$|\phi\rangle = \pm \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle. \quad (\text{II.29})$$

IF f IS BALANCED

If f is balanced, let

$$S_+ = \{x : f(x) = 0\}, \quad S_- = \{x : f(x) = 1\}, \quad |S_+| = |S_-| = 2^{n-1}.$$

Then

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \left(\sum_{x \in S_+} |x\rangle - \sum_{x \in S_-} |x\rangle \right) = 0. \quad (\text{II.30})$$

ORTHOGONALITY WITH THE UNIFORM SUPERPOSITION

Define the uniform state

$$|\phi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle, \quad |\phi_f\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle, \quad (\text{II.31})$$

then

$$\langle \phi_0 | \phi_f \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)}.$$

For balanced f , half the terms are $+1$ and half are -1 , so the sum is 0:

$$\langle \phi_0 | \phi_f \rangle = 0 \quad (\text{balanced case}).$$

AFTER APPLYING $H^{\otimes n}$

- Suppose that the function we are querying is a constant. If we apply Hadamard gates to the state produced after the phase kick back trick then:

$$H^{\otimes n} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle =$$

$$(-1)^{f(x)} H^{\otimes n} \frac{1}{\sqrt{2^n}} \sum |x\rangle = (-1)^{f(x)} |0\rangle \quad (\text{II.32})$$

Hence if we measure the computational basis after performing the n Hadamards, and the function is constant then the measurement outcome will always be $|0\rangle$.

- If the function we are querying is balanced:

$$H^{\otimes n} \frac{1}{\sqrt{2^n}} \sum (-1)^{f(x)} |x\rangle =$$

$$\frac{1}{2^n} \sum_{x',y' \in \{0,1\}^n} \underbrace{(-1)^{x' \cdot y'} |y'\rangle \langle x'|}_{H^{\otimes n}} \sum (-1)^{f(x)} |x\rangle =$$

$$\frac{1}{2^n} \sum_{x,y' \in \{0,1\}^n} (-1)^{x' \cdot y' + f(x)} \delta_{xx'} |y'\rangle = \frac{1}{2^n} \sum_{x,y' \in \{0,1\}^n} (-1)^{x \cdot y' + f(x)} |y'\rangle \quad (\text{II.33})$$

If now we measure the computational basis

$$\left| \langle 0 | \frac{1}{2^n} \sum (-1)^{x \cdot y' + f(x)} |y'\rangle \right|^2 = \left(\frac{1}{2^n} \sum (-1)^{x \cdot y' + f(x)} \right)^2$$

as there exists equal numbers of input the result is 0.

II.6 BERNSTEIN-VAZIRANI

ALGORITHM

- In the BV algorithm, we are given an n -bit function

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

which outputs a single bit. This function is of the form

$$f_s(x) = s \cdot x,$$

where s is an unknown n -bit string. The goal of BV is to find s .

-

$$U_s = \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}} |x\rangle\langle x| \otimes |y \oplus f_s(x)\rangle\langle y|.$$

- **Similar to the Deutsch–Jozsa algorithm by the help of the kick back trick, we create a state:**

$$|\psi_s\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f_s(x)} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot s} |x\rangle.$$

- **The states are all orthogonal:**

$$\langle \psi_t | \psi_s \rangle = \delta_{st} .$$

The proof of orthogonality is shown as below:

$$\begin{aligned} \langle \psi_s | \psi_t \rangle &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot s} \langle x | \sum_{y \in \{0,1\}^n} (-1)^{y \cdot t} |y\rangle \\ &= \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot s + y \cdot t} \langle x | |y\rangle \\ &= \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot s + y \cdot t} \delta_{xy} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot (s \oplus t)}. \end{aligned}$$

Now consider the sum for a fixed value $k = s \oplus t$:

$$\sum_{x \in \{0,1\}^n} (-1)^{x \cdot k} = \sum_{x \in \{0,1\}^n} (-1)^{x_1 k_1} (-1)^{x_2 k_2} \dots (-1)^{x_n k_n} = 2^n \delta_{k,0}.$$

So we summarize as:

$$\langle \psi_s | \psi_t \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot (s \oplus t)} = \frac{1}{2^n} 2^n \delta_{k,0}$$

This concludes the orthogonality.

Quantum Computing

Now, that we showed the states are all orthogonal, we can apply Hadamard. A reminder that:

$$H^{\otimes n} = \sum_{y \in \{0,1\}^n} |y\rangle \langle \psi_y|.$$

Hence we have:

$$H^{\otimes n} |\psi_s\rangle = \sum_{y \in \{0,1\}^n} |y\rangle \langle \psi_s | \psi_t\rangle = |s\rangle.$$

We can express it as:
$$H^{\otimes n} = \sum_{y \in \{0,1\}^n} |y\rangle \langle \psi_y|.$$

$$H^{\otimes n} |\psi_s\rangle = \sum_{y \in \{0,1\}^n} |y\rangle \langle \psi_y| \psi_s = |s\rangle.$$

This is the case of non-recursive BV algorithm. We can also look at the recursive algorithm which is layered problem as is explained in the next Section.

RECURSIVE BV

You are given access to the function

$$f(x, y) = s_x \cdot y \pmod{2}.$$

- For each $x \in \{0, 1\}^n$ there is a hidden $s_x \in \{0, 1\}^n$.
- You can query the oracle to get: $f(x, y) = s_x \cdot y$.
- You are told: $g(s_x) = x \cdot s \pmod{2}$.

BLACK-BOX ACCESS AND PROMISE (EXAMPLE) AND THE CLASSICAL SOLUTION

Assume for the 3-bit secret string:

- The only black box access is:

$$f(x, y) = s_x \cdot y, \quad x, y \in \{0, 1\}^3.$$

- For each x there is a hidden 3-bit string s_x that the solver never sees directly. Also

$$g(s_x) = x \cdot s.$$

SOLVER STRATEGY CLASSICAL WAY

1. Query the oracle on three y -basis vectors to read three bits of s_x :
- $$s_x[0] = f(x, 100), \quad s_x[1] = f(x, 010), \quad s_x[2] = f(x, 001).$$
2. Compute $g(s_x) \rightarrow x \cdot s.$
 3. Collect those values for three linearly-independent choices of x ; solve the resulting linear system.
 4. Finally output $g(s).$

QUERY 1: $x = 100$

y	oracle answer
100	I
010	O
001	O

$$s_{100}[0] = 1, \quad s_{100}[1] = 0, \quad s_{100}[2] = 0 \Rightarrow s_{100} = 100.$$

$$g(s_x) = g(s_{100}) = 1.$$

$$100 \cdot s = 1 \Rightarrow s_0 = 1.$$

Quantum Computing

QUERY 2: $x = 010$

y	oracle answer
100	I
010	O
001	I

Query 3) $x = 001$

y	oracle answer
100	I
010	I
001	O

$$s_{001}[0] = 1, \quad s_{001}[1] = 1, \quad s_{001}[2] = 0.$$

$$s_{110} = 110 \Rightarrow g(s_{110}) = 0 \Rightarrow 001 \cdot s = 0 \Rightarrow (s_2 = 1).$$

$$\boxed{\text{Therefore } s = 100} \quad \Rightarrow \quad \boxed{g(s_x) = 1}.$$

Okay! what is the classical query complexity? $O(n^k)$
 where k is the number of layers; in this example it is 2. Now lets move into the quantum approach.

Quantum Solution (Level 2 RBV)

We re-state the problem as:

$$f(x_1, x_2) = s_{x_1} \cdot x_2, \quad x_1 \in \{0, 1\}^n, \quad x_2 \in \{0, 1\}^n.$$

$$g(s_{x_1}) = x_1 \cdot s.$$

The goal is to find $g(s)$.

Step 1: The Unitary Oracle

$$U_f = \sum_{x_1, x_2} \sum_y \left(|x_1\rangle \langle x_1| \right) \otimes \left(|x_2\rangle \langle x_2| \right) \otimes \left(|y \oplus f(x_1, x_2)\rangle \langle y| \right),$$

which is:

$$|x_1\rangle |x_2\rangle |y\rangle \longrightarrow |x_1\rangle |x_2\rangle |y \oplus f(x_1, x_2)\rangle.$$

where,

$$f(x_1, x_2) = s_{x_1} \cdot x_2.$$

Step 2: The Phase kickback trick

$$|y\rangle \rightarrow |- \rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Prepare the initial state:

$$|\Psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1, x_2} |x_1\rangle |x_2\rangle \otimes |- \rangle.$$

Apply the oracle (phase kick back trick is used):

$$U_f |\Psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1, x_2} (-1)^{f(x_1, x_2)} |x_1\rangle |x_2\rangle |- \rangle = \frac{1}{\sqrt{2^n}} \sum_{x_1, x_2} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle |x_2\rangle |- \rangle$$

Okay a reminder on Hadamard:

$$H|b\rangle = \frac{1}{\sqrt{2}} \sum_{z \in \{0,1\}} (-1)^{bz} |z\rangle,$$

and applying Hadamard on n bits will be:

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle.$$

Hadamard is supposed to be applied to the second register $|x_2\rangle$:

$$\sum_{x_2} (-1)^{s_{x_1} \cdot x_2} |x_2\rangle.$$

Quantum Computing

You compute:

$$H^{\otimes n} \sum_{x_2} (-1)^{s_{x_1} \cdot x_2} |x_2\rangle = \sum_{x_2} (-1)^{s_{x_1} \cdot x_2} H^{\otimes n} |x_2\rangle \\ = \sum_{x_2} (-1)^{s_{x_1} \cdot x_2} \left(\frac{1}{\sqrt{2^n}} \sum_z (-1)^{x_2 \cdot z} |z\rangle \right) = \frac{1}{\sqrt{2^n}} \sum_z \sum_{x_2} (-1)^{x_2 \cdot (s_{x_1} \oplus z)} |z\rangle.$$

Hence:

$$H^{\otimes n} |\Psi_0\rangle = H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{x_1, x_2} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle |x_2\rangle |- \rangle \right)$$

becomes:

$$\frac{1}{2^n} \sum_z 2^n \delta_{s_{x_1}, z} |z\rangle = |s_{x_1}\rangle.$$

Hence, we are able to resolve s_{x_1} .

Now we know:

$$g(s_{x_1}) = x_1 \cdot s.$$

So we use another unitary operator G :

$$G = \sum_x \sum_y |x\rangle \langle x| \otimes |y \oplus g(x)\rangle \langle y|.$$

$$G |x\rangle |y\rangle = |x\rangle |y \oplus g(x)\rangle.$$

Okay, now we don't have direct access to $g(s_{x_1})$, so we introduce an extra ancilla qubit initialized to $|0\rangle$.

$$\frac{1}{\sqrt{2^n}} \sum_{x_1} |x_1\rangle \otimes |s_{x_1}\rangle \otimes |- \rangle \otimes |0\rangle.$$

Now applying G : $G |s_{x_1}\rangle |0\rangle = |s_{x_1}\rangle |g(s_{x_1})\rangle$.

$$\Rightarrow \frac{1}{\sqrt{2^n}} \sum_{x_1} |x_1\rangle \otimes |s_{x_1}\rangle \otimes |- \rangle \otimes |g(s_{x_1})\rangle.$$

(Use kickback trick again.)

If we prepare the last qubit as $|- \rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, then

$$G(|s_{x_1}\rangle |-\rangle) = (-1)^{g(s_{x_1})} |s_{x_1}\rangle |-\rangle.$$

$$\Rightarrow \frac{1}{\sqrt{2^n}} \sum_{x_1} (-1)^{g(s_{x_1})} |x_1\rangle \otimes |s_{x_1}\rangle \otimes |-\rangle \otimes |-\rangle.$$

Question I want to use $H^{\otimes n}$ to resolve s_{x_1} , but I can't now because $|s_{x_1}\rangle$ is also there and entangled to $g(s_{x_1})$. What can I do?

We apply H to $|s_{x_1}\rangle$ first, and then apply the oracle U :

$$H^{\otimes n} |s_{x_1}\rangle = \frac{1}{\sqrt{2^n}} \sum_{x_2 \in \{0,1\}^n} (-1)^{x_2 \cdot s_{x_1}} |x_2\rangle.$$

$$\text{So we get } \frac{1}{2^n} \sum_{x_1, x_2 \in \{0,1\}^n} (-1)^{g(x_1)} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle |x_2\rangle |-\rangle |-\rangle.$$

And then we apply U :

$$\begin{aligned} & \frac{1}{2^n} \sum_{x_1, x_2 \in \{0,1\}^n} (-1)^{g(x_1)} (-1)^{s_{x_1} \cdot x_2} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle |x_2\rangle |-\rangle |-\rangle \\ &= \frac{1}{2^n} \sum_{x_1, x_2 \in \{0,1\}^n} (-1)^{g(x_1)} |x_1\rangle |x_2\rangle |-\rangle |-\rangle. \end{aligned}$$

Now if we apply $H^{\otimes n}$ to the first register, we will obtain $g(x_1)$ and consequently s .

This approach can be extended to k levels of BV. So we need $O(2^k)$ queries. If we set $k = \log(n)$, we see that quantum algorithm takes $O(n)$ while classically $O(n^{\log n})$ queries are needed.