The supremacy of quantum algorithms: The Bernstein-Vazirani Algorithm

Introduction

Imagine you're in an arcade, and you've discovered a mysterious slot machine. Here's how it works: you have three coins – some dimes and some quarters. The machine allows you to throw in these coins in various combinations, and it will either give you nothing or exactly $1. Your goal is to figure out the exact hidden combination of dimes and quarters that unlocks the dollar jackpot.

This arcade game is a playful analogy for the Bernstein-Vazirani algorithm, which solves a similar type of puzzle – but with quantum superpowers! Just as the slot machine has a hidden rule based on your coin choices, this algorithm is all about discovering a secret pattern (a hidden string of 0s and 1s) that determines the output of a special function. The challenge? Doing it as efficiently as possible.

How to Play: (Students and teacher)

1. Students have three coins to start with, represented as bits (0 for dimes and 1 for quarters).
2. The slot machine (teacher) has the key (hidden string) and evaluates the input combination and gives the result: 0 (no dollar) or 1 (jackpot!).
3. Students mission is to guess the exact rule (the hidden string) that determines the result.

Mathematical formulation of the problem

Imagine we have a mystery box represented by a function f_a. This box is special because when you give a binary number with n bits, it returns either 0 or 1. The function f_a is defined using a fixed secret binary number $a = (a_1, a_2, \ldots, a_n)$, where each a_i is either 0 or 1.

To compute $f_a(x)$, the function follows these steps:

- Take a n-bit input $x = (x_1, x_2, \ldots, x_n)$, where each x_i is 0 or 1.
- Multiply each input bit by the corresponding bit in a: $a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_n \cdot x_n$
- Check the result of the sum:
    - If the sum is even, the function outputs 0.
    - If the sum is odd, the function outputs 1.

Task 1 Use the hidden binary number $\boldsymbol{a = (1, 0, 1)}$ for the case $\boldsymbol{n = 3}$ and complete the following table by computing $\boldsymbol{f_a(x)}$.

Input x	$a \cdot x = a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_3$	Odd	Even	Output $f_a(x)$
$(0,0,0)$	$1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 = 0$		x	0
$(0,0,1)$	$1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 = 1$	x		1
$(0,1,0)$				
$(1,0,0)$				
$(0,1,1)$				
$(1,0,1)$				
$(1,1,0)$				
$(1,1,1)$				

Analyse the Results: How many and which combinations of quarters and dimes will result in a 1$ win. Notice how the outputs depend on the hidden pattern. This pattern is the key to understanding the function's behaviour.

Task 2: Now we focus on the actual problem to find the hidden string of a function $f_a(x_1,x_2,x_3)$ from the table below:

x_1	x_2	x_3	$f_a(x_1,x_2,x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

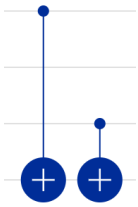
Find the most efficient strategy to determine the string $\mathbf{a}=(a_1,a_2,a_3)$:

- How many queries do we need?
- Which input values (x_1,x_2,x_3) would you use to determine the string $\mathbf{a}=(a_1,a_2,a_3)$ most efficiently?
- What is the hidden string?

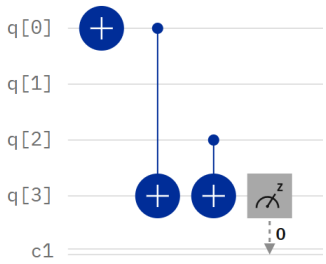
Quantum computer simulation with the IBM composer

1. In this first approach we you use a classical algorithm to find the hidden string.

Open https://quantum.ibm.com/composer/files/new.The circuit below implements a particular string, which we are trying to figure out.

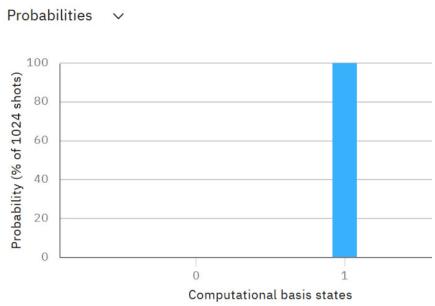


Example: Input x_1=(1,0,0).



- The input values x_1,x_2,x_3 are the inputs from the first three qubits q[0],q[1],q[2].
- The function value is the output from q[3].
- All qubits are initialized to input values 0. In order to change the input value from 0 to 1, apply ⊕ to the respective qubit.
- Measurement  of the last qubit q[3]= $f(x_1,x_2,x_3)$.

The measured value, in this case 1, is the function value $f(x_1,x_2,x_3)$which can be read from the graph that shows the simulation of the measurement probability:



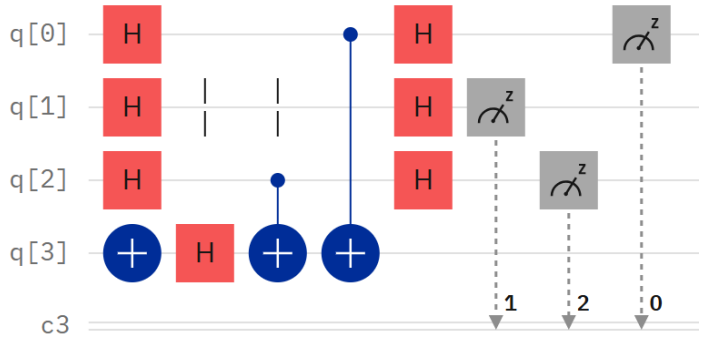
Task 3: Build the above quantum circuit and use the quantum composer to complete the table below:

x_1,x_2,x_2	0,0,0	1,0,0	0,1,0	0,0,1	1,1,0	1,0,1	0,1,1	1,1,1
$f(x_1,x_2,x_2)$		1						

Find the hidden string from your table.

2. In this second approach we use the Bernstein Vazirani quantum algorithm to find the hidden string

The quantum algorithm uses the circuit below. The function is, as before, implemented by the 2 CNOT gates. The hidden output string is obtained after measuring the output of the first 3 qubits. Similar as in the case of the Deutsch Algorithm, the initial Hadamard gates create a superposition of all possible input states.

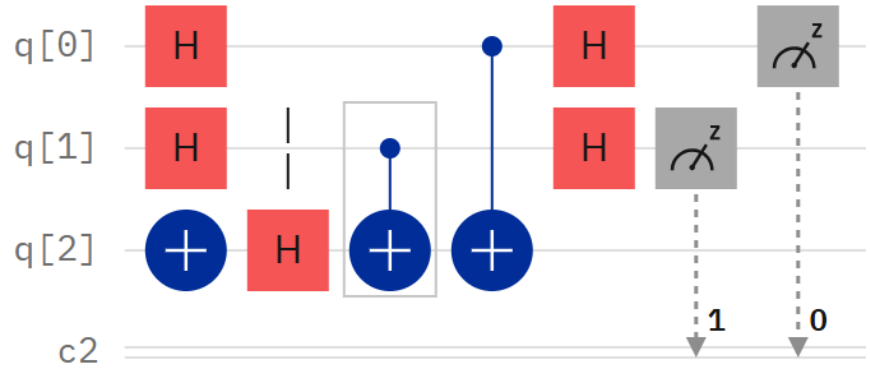


Build the Quantum algorithm and confirm that the simulation indeed reveals the string you found classically in problem 3.

Notice: It took 3 queries classically, but only one query using the quantum algorithm.

In order to explore how the quantum algorithm solves the problem, let's look at the two dimensional example, $f(x_1,x_2)$, which state evolution we can easily track with help from an Excel sheet.

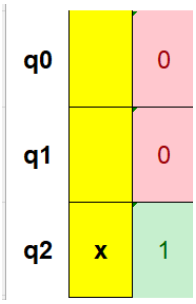
Task 4: Build the circuit below one gate at a time as you fill in the Excel sheet one gate at a time. Confirm that your Excel agrees with the state evolution from the composer.



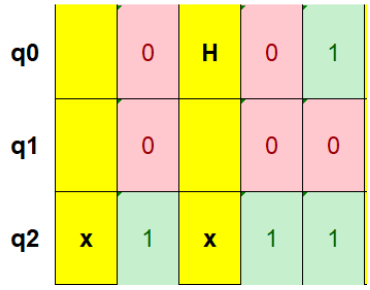
Some help how to get started. Use ket notation, explain that I drop the normalisation in the excel

Initially all qubits are in state $|0\rangle$: $|q_0\rangle|q_1\rangle|q_2\rangle \equiv |q_0, q_1, q_2\rangle = |0,0,0\rangle$.

The $\oplus$-Gate is acting on $|q_2\rangle$ and flips the total state to $|0,0,1\rangle$, which we note in our table:



The Hadamard gate acting on $|q_0\rangle$ and creates a superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The total state is therefore $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle|1\rangle = \frac{1}{\sqrt{2}}(|0,0,1\rangle + |1,0,1\rangle)$. For simplicity, we drop the normalisation factor and denote the final state in our table as



Finish the Excel table and answer the following questions:

a) What do the two Hadamard gates in the beginning accomplish?

b) Why is there an ⊕-Gate at the beginning of the output qubit? Look what would happen if the ⊕-Gate was missing.

c)What is the purpose of the last two Hadamard gates?

d)What string (a_1,a_2) does the measurement reveal?