The Bernstein-Vazirani Algorithm

Peter Young

(Dated: October 29, 2019)

Like the Deutsch algorithm, the Bernstein-Vazirani algorithm finds information about a black box function, but has a bigger speedup. It is very similar to the Deutsch-Josza algorithm which is set as a homework assignment.

Consider a function

$$f(x) = a \cdot x \tag{1}$$

where a and x have n bits while the function itself, f, has one bit. The dot indicates a bitwise inner product with modulo 2 addition:

$$a \cdot x \equiv a_0 x_0 \oplus a_1 x_1 \oplus \cdots \oplus a_{n-1} x_{n-1} \, . \tag{2}$$

The problem is to determine a.

Let's make sure that we understand the "dot". For example for $n = 4$, if the bits of a are 1101 and the bits of x are 1110 (recall that the zeroth bit is the least significant, i.e. the rightmost one) then[1]

$$a \cdot x = (1 \times 0) + (0 \times 1) + (1 \times 1) + (1 \times 1) \mod 2 = 0 + 0 + 1 + 1 \mod 2 = 2 \mod 2 = 0. \tag{3}$$

Hence, for these values of a and x, $f(x) = 0$. If we take $x = 1000$ then $f(x) = 0 + 0 + 0 + 1 \mod 2 = 1$.

Classically we can only determine the bits of a one at a time. The k-th bit of a can be determined by feeding in $x = 2^k$. To see this, consider the binary representation of a:

$$a = a_0 + a_1 2^1 + \cdots + a_k 2^k + \cdots a_{n-1} 2^{n-1} \, , \tag{4}$$

and similarly for x. Hence if $x = 2^k$ then $x_k = 1$ while, for $l \neq k, x_l = 0$, so $a \cdot x = a_k$. Consequently $f(2^k) = a_k$. We have to do this for each bit, $k = 0, 1, 2, \cdots, n-1$, so it requires n calls of the function.

[1] One can either do the mod 2 operation after each addition or add up in the normal way and apply the mod 2 operation at the end. In either case, the result is 0 if an even number of terms in the sum are 1, and 1 if an odd number of terms are 1.

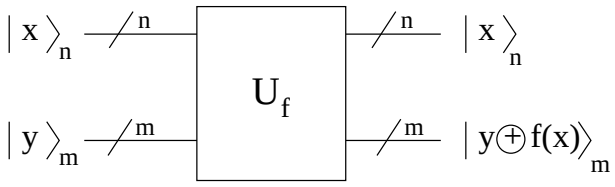

FIG. 1: Schematic diagram of a general unitary transformation U_f for an n-bit input x and an m-bit output $f(x)$. The upper register in the figure has n qubits and contains the input value x. The lower register has m qubits and contains information about the function value $f(x)$. The registers are shown as single lines. To ensure the transformation is reversible we have $n + m$ qubits in both the initial state (to the left) and final state (to the right). For the Bernstein-Vazirani Algorithm $m = 1$.

The quantum algorithm succeeds in determining a with just *one* call!

For the Bernstein-Vazirani Algorithm there are n input qubits and 1 output qubit. A schematic diagram of a general reversible unitary transformation which takes an n-bit input and generates an m-bit output is shown in Fig. 1. (For the Bernstein-Vazirani algorithm, $m = 1$.)

In the Bernstein-Vazirani algorithm the unitary U_f is surrounded by Hadamards, as shown in Fig. 2. The upper register is set to $|0\rangle_n$ and the lower qubit to $|1\rangle$. This is the same circuit as for the Deutsch-Josza algorithm.

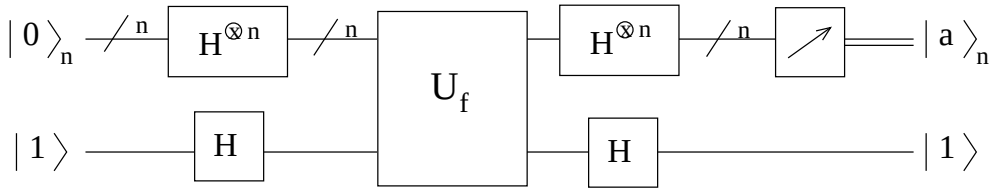

FIG. 2: Circuit diagram for the Bernstein-Vazirani algorithm. In the final state the upper (input) register contains $|a\rangle$ while the lower (output) qubit reverts to its initial state $|1\rangle$. The desired value of a can therefore be read off by measuring the upper register.

Acting with H on $|0\rangle$ gives an equal linear superposition of the two basis states. Similarly acting with $H^{\otimes n}$ on $|0\rangle_n$ gives an equal superposition of the 2^n basis states. Hence, including the lower register, the state inputted to U_f is

$$H^{\otimes n}|0\rangle_n \otimes H|1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^{n-1}} |x\rangle_n \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \,. \tag{5}$$

For each term in the superposition, the function U_f acts in the same way as for the Deutsch algorithm described in `https://young.physics.ucsc.edu/150/deutsch.pdf`. The lower qubit is

flipped if $f(x) = 1$, which is the same as changing the sign of the state. This sign change is called *phase kickback* by Vathsan[1]. If $f(x) = 0$ there is no change. Hence each term in the superposition acquires a factor of $(-1)^{f(x)}$, so the state of the system immediately the action of U_f is

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^{n-1}} (-1)^{f(x)} |x\rangle_n \otimes \frac{(\,|0\rangle - |1\rangle\,)}{\sqrt{2}} \,. \tag{6}$$

Next consider the effect of the Hadamards acting after U_f. The action on the lower qubit is to convert $(\,|0\rangle - |1\rangle\,)/\sqrt{2}$ to $|1\rangle$. However, the effect of $H^{\otimes n}$ acting on an arbitrary computational basis state $|x\rangle_n$ needs more thought. Consider first just one qubit. Then

$$H|x\rangle_1 = \frac{1}{\sqrt{2}} (\,|0\rangle + (-1)^x |1\rangle\,) = \frac{1}{\sqrt{2}} \sum_{y=0}^{1} (-1)^{xy} |y\rangle \,. \tag{7}$$

Hence the effect of applying $H^{\otimes n}$ on an n-qubit computational basis state is

$$\begin{aligned} H^{\otimes n}|x\rangle_n &= \sum_{y_{n-1}=0}^{1} \cdots \sum_{y_1=0}^{1} \sum_{y_0=0}^{1} (-1)^{\sum_{j=0}^{n-1} x_j y_j} |y_{n-1}\rangle \cdots |y_1\rangle |y_0\rangle \,, \\ &= \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y} |y\rangle_n \,, \end{aligned} \tag{8}$$

where $x \cdot y$ is the bitwise inner product defined in Eq. (2), and we have used that we only need to know whether $\sum_{j=0}^{n-1} x_j y_j$ is even or odd. Hence, combining Eqs. (6) and (8), the amplitude to find the upper register in state $|y\rangle \equiv |y_{n-1}\rangle \cdots |y_1\rangle |y_0\rangle$ is

$$\begin{aligned} c_y =& \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)+x\cdot y} \\ =& \frac{1}{2^n} \prod_{j=0}^{n-1} \left[\sum_{x_j=0}^{1} (-1)^{(a_j+y_j)x_j} \right] . \end{aligned} \tag{9}$$

Let us evaluate this for the state where $y_j = a_j$ for all j. Then $(a_j + y_j)x_j = 2$ or 0 and, since this quantity is the power of -1 we only need it modulo 2, so all the factors of $(-1)^{(a_j+y_j)x_j}$ are $+1$, and consequently all 2^n terms from the sums over the x_j add up in phase. The result is that the amplitude for $y = a$ is 1. Since the total probability must add up to 1 this means that all the other amplitudes must be zero. To see that this is indeed the case, note that for each qubit where $y_j \neq a_j$, $a_j + y_j = 1$ and so the sum over x_j for these qubits gives zero. The final result is a product over terms for each qubit and so we get zero, as required.

Including the output qubit, the final state is

$$|a\rangle_n \otimes |1\rangle \,, \tag{10}$$

and a measurement of the upper register in Fig. 2 gives a, even though we made just *one* call to the function.

Since a classical computation of a requires n function calls, we have obtained a "*quantum speedup*" of n. Note that the procedure is analagous to Deutsch's algorithm. The first set of Hadamards generates a superposition of inputs to the gate U_f which "evaluates"[2] the function for all 2^n inputs using *quantum parallelism*, and then the second set of Hadamards destroys all the outputs apart from a, using quantum *interference*.

Following Mermin[2] and Vathsan[1] it is useful to give an alternative derivation of how the circuit in Fig. 2 works, by giving an *explicit* construction of the black box U_f. It is convenient to illustrate by a specific example. We take $n = 5$ and $a = 11010$ so $x_0 = 0, x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$. The function $a \cdot x$ can be implemented by the gates shown in Fig. 3.

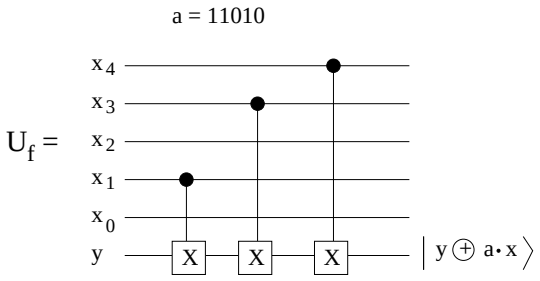

FIG. 3: A circuit diagram for $n = 5$ to implement the function $f(x) = a \cdot x$ with $a = 11010$, i.e. $f(x) = x_1 + x_3 + x_4 \mod 2$. The circuit flips the output qubit, the lowest one, initialized to y, whenever x_1 or x_3 or x_4 is 1. (Note that flipping y is equivalent to adding 1 to y mod 2.) Hence the final value of the output qubit is $y \oplus (a \cdot x)$ as required.

To incorporate U_f into the Bernstein-Vazirani algorithm, we sandwich it in between Hadamards, see Fig. 2, and note that the Hadamards interchange control and target qubits in the CNOT (control-X) gates, see Fig. 5(f) in the handout on the Deutsch algorithm, `https://young.physics.ucsc.edu/150/deutsch.pdf`. As before, the initial upper register is $|0\rangle_n$ and the lower register is $|1\rangle$. We see immediately from Fig. 4 that a is *directly* imprinted in the final state of the input register. There does not appear to be any parallelism and interference.

Hence these two explanations of the Bernstein-Vazirani algorithm are quite different. To quote Mermin[2]:

---

[2] To understand the reason for the quotation marks see footnote 3 in the handout on the Deutsch algorithm, `https://young.physics.ucsc.edu/150/deutsch.pdf`

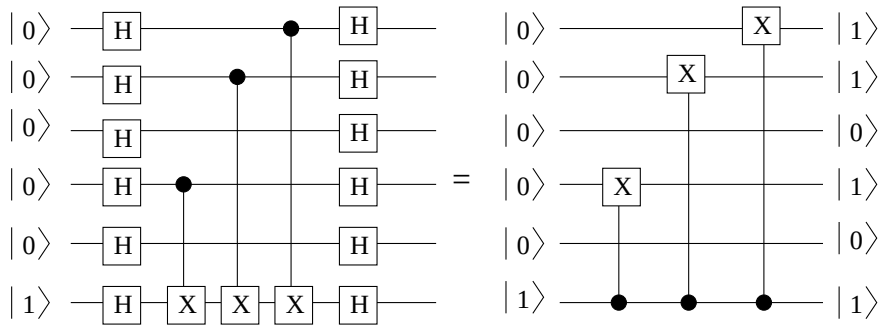

FIG. 4: Sandwiching the circuit for U_f in Fig. 3 between Hadamards, and realizing that the effect of the Hadamards is to interchange the control and target qubits in the CNOT (control-X) gates, we see immediately that the final state of the input register contains $a = 11010$.

> "The first applies U_f to the quantum superposition of all possible inputs and then applies operations which leads to perfect destructive interference of all states in the superposition except for the one in which the upper (input) register is in the state $|a\rangle$. The second suggests a specific mechanism for representing the subroutine that executes U_f and then shows that sandwiching such a mechanism between Hadamards *automatically* (my italics) imprints a on the upper register. Interestingly, quantum mechanics appears in the second method only because it allows the reversal of the control and target qubits of a cNOT operation solely by means of 1-qubit (Hadamard) gates."

(I have used the conventional spelling of "qubit" rather than Mermin's idiosyncratic "Qbit".)

---

[1] R. Vathsan, *Introduction to Quantum Physics and Information Processing* (CRC Press, Boca Raton, 2016).

[2] N. D. Mermin, *Quantum Computer Science* (Cambridge University Press, Cambridge, 2007).