Quantum Programming Algorithms: Bernstein-Vazirani

Jens Palsberg

Apr 17, 2019

Outline

Hook: Our second quantum algorithm is the Bernstein-Vazirani algorithm. It solves a natural problem and does so faster on a quantum computer than we can ever do on a classical computer.

Purpose: Persuade you that you can understand this algorithm.

Preview:
1. We will make sure that everybody understands the problem.
2. We will cover full details of how and why the algorithm works.
3. We will show an additional algorithm that uses similar ideas.

Transition to Body: Now let us talk about the problem that this algorithm solves.

Main Point 1: We will make sure that everybody understands the problem.
[Most of you have done the homework on solving the problem on a classical computer]
[We can represent the black-box function in multiple ways]
[We can program the solution in any classical language]

Transition to MP2: Now let us look at how to solve the problem.

Main Point 2: We will cover full details of how and why the algorithm works.
[We will mix classical and quantum computing]
[The structure of the quantum part is the same as in Deutsch-Josza]
[We can use some of the lemmas about Deutsch-Josza]

Transition to MP3: Now let us apply our knowledge to a different case.

Main Point 3: We will show an additional algorithm that uses similar ideas.
[The problem is a variation of what we have seen before]
[The solution uses a different circuit]
[The reasoning uses some of the same lemmas that we proved earlier]

Transition to Close: So there you have it.

Review: The Bernstein-Vazirani algorithm combines classical and quantum computing.

Strong finish: The Bernstein-Vazirani algorithm is the second indication that a quantum computer can be faster than a classical computer. We will see more examples of that in this course.

Call to action: Look for other natural problems that may be solvable on quantum computers.

Detailed presentation

Hook: Our second quantum algorithm is the Bernstein-Vazirani algorithm. It solves a natural problem and does so faster on a quantum computer than we can ever do on a classical computer.

Purpose: Persuade you that you can understand this algorithm.

Preview:

1. We will make sure that everybody understands the problem.
2. We will cover full details of how and why the algorithm works.
3. We will show an additional algorithm that uses similar ideas.

Transition to Body: Now let us talk about the problem that this algorithm solves.

Main Point 1: We will make sure that everybody understands the problem.

[Most of you have done the homework on solving the problem on a classical computer]
Here is the homework problem.

The Bernstein-Vazirani problem:
Input: a function $f : \{0,1\}^n \to \{0,1\}$.
Assumption: $f(x) = a \times x + b$.
Output: a, b.
Notation: $\{0,1\}^n$ is the set of bit strings of length n, a is an unknown bit string of length n, $\times$ is inner product mod 2, $+$ is addition mod 2, and b is an unknown single bit.

The assignment:
On a classical computer, in a classical language of your choice (such as C, Java, Python, etc), program solutions to the Bernstein-Vazirani problem. Treat the input function f as black box that you can call but cannot inspect in any way at all. Each solution will be code that includes one or more calls of f.

Notice that if $a = 00\ldots0$, then f is *constant*. Additionally, if $a \neq 00\ldots0$, then f is *balanced*. So, if we can solve Bernstein-Vazirani, we can solve Deutsch-Josza.

[We can represent the black-box function in multiple ways]
How do we keep the black-box function at an arms length such that we cannot get tempted to inspect it? In C we can use a function pointer. In Java we can use a lambda expression. In Python we can use an anonymous function. Other languages have similar constructs that will enable us to call the function, while giving us no way to inspect it.

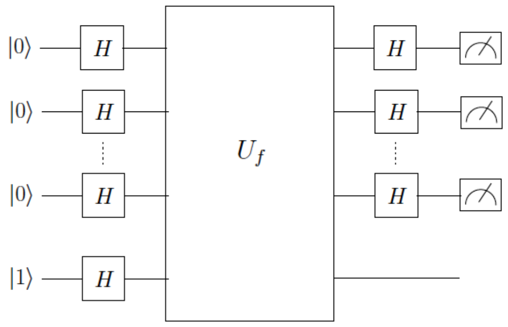
[We can program the solution in any classical language]
You wrote the solutions to the homework in several languages. We need to make several calls to f: a total of $n+1$ calls to f are needed. The reason is that we need to call $f(00\ldots0) = b$, and then we need to do n calls to determine a.

Transition to MP2: Now let us look at how to solve the problem.

Main Point 2: We will cover full details of how and why the algorithm works.

[We will mix classical and quantum computing]
First do a classical call $f(00\dots0) = b$. Then use a quantum circuit to determine a.

[The structure of the quantum part is the same as in Deutsch-Josza]



[We can use some of the lemmas about Deutsch-Josza]
After the second round of uses of H, we know that the state of the first n qubits is:

$$\begin{aligned}
& \frac{1}{2^n}\ \Sigma_{x\in\{0,1\}^n}\ \Sigma_{y\in\{0,1\}^n}\ (-1)^{x\cdot y+f(x)}\ |y\rangle \\
= & \frac{1}{2^n}\ \Sigma_{x\in\{0,1\}^n}\ \Sigma_{y\in\{0,1\}^n}\ (-1)^{x\cdot y+a\times x+b}\ |y\rangle \\
= & \frac{1}{2^n}\ (-1)^b\ \Sigma_{x\in\{0,1\}^n}\ \Sigma_{y\in\{0,1\}^n}\ (-1)^{(a+y)\times x}\ |y\rangle \\
= & (-1)^b\ |a\rangle
\end{aligned}$$

In the third step, we rely on this calculation:

$$\begin{aligned}
& \text{The amplitude of } |a\rangle \\
= & \frac{1}{2^n}\ (-1)^b\ \Sigma_{x\in\{0,1\}^n}\ (-1)^{(a+a)\times x} \\
= & \frac{1}{2^n}\ (-1)^b\ \Sigma_{x\in\{0,1\}^n}\ 1 \\
= & (-1)^b
\end{aligned}$$

So, when we measure the first n qubits, we will observe $|a\rangle$ with probability $|(-1)^b|^2 = 1^2 = 1$.

Transition to MP3: Now let us apply our knowledge to a different case.

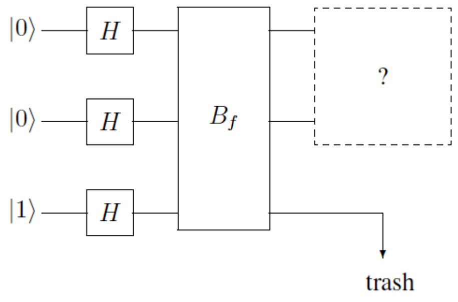
Main Point 3: We will show an additional algorithm that uses similar ideas.

[The problem is a variation of what we have seen before]

Suppose we are given $f : \{0,1\}^2 \to \{0,1\}$ and we are promised that f returns 1 for a single input and that f returns 0 otherwise. Our goal is to determine which input makes f return 1.

In classical computing, we have to make three calls of f to solve the problem.

[The solution uses a different circuit]



Above, the diagram uses B_f instead of U_f. The box with the "?" is occupied by the following matrix:

$$U = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

[The reasoning uses some of the same lemmas that we proved earlier]

Let us give names to four particular superpositions:

$$\begin{aligned}
\phi_{00} &= \frac{1}{2}(-|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\
\phi_{01} &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle + |11\rangle) \\
\phi_{10} &= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle + |11\rangle) \\
\phi_{11} &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle)
\end{aligned}$$

Lemma 1. $\forall c, d \in \{0,1\} : \; U(\ \phi_{cd}\ ) = |cd\rangle.$

Proof. The proof has four cases. We will show one of the cases; the others are similar.

$$U(\ \phi_{00}\ ) = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} (\ \frac{1}{2}\begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix}\ ) = \frac{1}{4}\begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

□

Theorem 2. *The algorithm works.*

Proof. The initial state of the three qubits is $|001\rangle$.

After using the three H, followed by the use of U_f, followed by the use of U, the state of the three qubits is:

$$\begin{aligned}
& (U \otimes I) \circ U_f \circ (H \otimes H \otimes H)\ |001\rangle \\
= \ & (U \otimes I) \circ U_f\ |{+}{+}{-}\rangle \\
= \ & (U \otimes I) \circ U_f\ \frac{1}{2}(|00-\rangle + |01-\rangle + |10-\rangle + |11-\rangle) \\
= \ & (U \otimes I)\frac{1}{2}((-1)^{f(00)}|00-\rangle + (-1)^{f(01)}|01-\rangle + (-1)^{f(10)}|10-\rangle + (-1)^{f(11)}|11-\rangle) \\
= \ & (U \otimes I)\frac{1}{2}(((-1)^{f(00)}|00\rangle + (-1)^{f(01)}|01\rangle + (-1)^{f(10)}|10\rangle + (-1)^{f(11)}|11\rangle) \otimes |-\rangle) \\
= \ & |cd\rangle|-\rangle \quad \text{where } f(cd) = 1
\end{aligned}$$

In the third step, we used the phase-kickback lemma. In the fifth step, we used Lemma 1.

Now we measure the first two qubits; with probability $|1|^2 = 1$, we get cd where $f(cd) = 1$. □

Transition to Close: So there you have it.

Review: The Bernstein-Vazirani algorithm combines classical and quantum computing.

Strong finish: The Bernstein-Vazirani algorithm is the second indication that a quantum computer can be faster than a classical computer. We will see more examples of that in this course.

Call to action: Look for other natural problems that may be solvable on quantum computers.