QuIC Seminar 10

The Bernstein-Vazirani Algorithm

Contents

10.1 The Bernstein-Vazirani Problem 46
10.2 Classical Query Complexity 47
10.3 The Bernstein-Vazirani Algorithm 47
10.4 Concluding Remarks 48

Last time we introduced the query model of quantum computing and saw one problem which a quantum computer could solve *twice* as fast as a classical computer, with respect to the same oracle. Twice as fast is pretty good. If it took an hour originally, you could do it an a *half hour*. Two weeks? *One* week. This could make a huge difference!

But we're not always concerned about the constant factors of algorithms[1]. Often, we're concerned with how the problem *scales* in the input to the algorithm. For example, if I give you a list of n numbers, how long does the algorithm take to run, roughly? Does the runtime t scale polynomially, i.e. $t = O(n^k)$ for some integer k? Or does it scale exponentially, i.e., $t = O(b^t)$ for some base exponent b?

This is often the metric that computer scientists use to determine whether an algorithm is *efficient*.

Definition 10.1 (Efficient algorithm.). An **efficient** algorithm is one in which the number of steps scales (at most) polynomially in the input size.

In the Deustch-Jozsa problem, there is necessarily no scaling—the function of interest only acts on *one* bit! We can't scale the problem up to larger bits and see how the complexity scales.

The Bernstein-Vazirani (BV) algorithm presents us with a quantum algorithm whose complexity scales better than the best classical algorithm. Namely, the quantum algorithm will give us a *linear* speedup in the number of queries relative to the best classical algorithm. (If "linear speedup" doesn't make sense to you now, it will by the end of the seminar.)

First, note that the complexity of the BV algorithm is in terms of the number of queries to an *oracle*, so the speedup is also in terms of the number of queries. We talked about justifying the query model in the last seminar. If you're still losing sleep over the query model setting, maybe the following analogy will help[2]. Imagine there's two graduate students, Alice and Bob, working for you. They both sit in the same room, and every time you come by you find out they haven't done anything! They're hardly doing any work, not making any progress, etc. Then someone asks you: Who's the better graduate student, Alice or Bob?

[1] Well, *computational* scientists emphatically *do* care about constant factors—parallelizing an algorithm to get even a 35% speedup, say, is a huge deal. On the other hand, *computer* scientists largely *don't* care about these constant factors. The coarse-grained measure of polynomial vs exponential, say, is the key characteristic here.

[2] I didn't come up with this, but I forgot where I heard it. When I remember, I'll add a citation.

How could you possibly answer this?! You can't say anything about either—as you haven't seen them do *anything*!

Then you have an idea. You give them each a computer, the same computer, and you let them get back to work. When you come back in a week, you find that Alice has made enormous progress, while Bob is still spinning his wheels.

So you go back to the person who asked you the question and say, *With respect to the computer,* Alice is the better graduate student.

This seems natural, doesn't it? This is the same thing we're saying in the query model setting. It's hard to answer general questions about "which is better: quantum or classical?" So we give them each the same tool, namely an *oracle*, and ask which is better *with respect to the oracle.* (Or, as a computer scientist would say, *relative to an oracle.*)

Alright, end analogy! It's time to see the real deal.

10.1 The Bernstein-Vazirani Problem

The problem that Bernstein and Vazirani considered in order to show their quantum speedup is the following.

Definition 10.2 (Bernstein-Vazirani problem.). Let f be a function from bit strings of length n to a single bit,

$$f : \{0,1\}^n \to \{0,1\}, \tag{10.1}$$

such that for all input bit strings $x \in \{0,1\}^n$, there exists a secret string $s \in \{0,1\}^n$ such that

$$f(x) = x \cdot s. \tag{10.2}$$

Here, $\cdot$ denotes the inner product modulo two. The problem is to find s by querying f as few times as possible.

To be absolutely clear about the problem, let's consider the following examples.

Example 1: Inner products modulo 2.
Let

$$x_1 = 100, \qquad x_2 = 001, \qquad x_3 = 111, \tag{10.3}$$

and consider $s = 101$. Then,

$$x_1 \cdot s = (1)(1) + (0)(0) + (0)(1) \pmod 2 = 1. \tag{10.4}$$

Here, we multiply the jth bit of x_1 with the jth bit of s and sum up the terms, taking the answer modulo two. This is exactly the same as the standard dot product of vectors over the field $\mathbb{Z}_2^n$. For example, we could have written the dot product above as

$$x_1 \cdot s = (1,0,0) \cdot (1,0,1) = 1. \tag{10.5}$$

It's easy to verify that $x_2 \cdot s = 1$ and $x_3 \cdot s = 0$.

Now that we're clear on what the inner product means, let's consider an example of such a function f with this property.

Example 2: Function with the Bernstein-Vazirani property.
Consider the following function f on two bits which obeys the Berstein-Vazirani property $f(x) = s \cdot x$.

x	f(x)
00	0
01	0
10	1
11	1

Given just this table of values, can you determine what the secret string s is to solve the Bernstein-Vazirani problem?

10.2 Classical Query Complexity

There's a method for solving the BV problem in n classical queries. Here, in the classical setting, we send one input string x into our "black box" oracle and learn the value $f(x)$. How can we do this with n queries?

Consider what happens when we send in the bit string $x = 100\cdots0$. (That is, the bit string with a one as its first bit and zeros elsewhere.) The query then sends us back

$$f(x) = x \cdot s = s_1, \tag{10.6}$$

where s_1 is the first bit of s.

Exercise 48: Prove that

$$x_j \cdot s = s_j \tag{10.7}$$

here x_j is the bit string with a one in the jth component and zeroes elsewhere.

Thus, we have an algorithm for determining s explicitly with n queries. Can we do better than this? The answer is no, simply because each bit of s is independent. It takes *at least* n bits of information to specify s, and we learn exactly one bit from each query. Thus, n classical queries is a lower bound.

10.3 The Bernstein-Vazirani Algorithm

The circuit for the Bernstein-Vazirani algorithm is shown below.

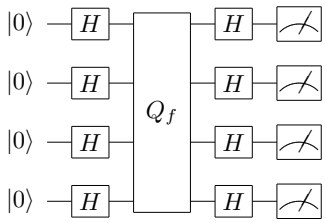

Figure 10.1: Quantum circuit for the Bernstein-Vazirani algorithm. Here, four qubits are shown, but the general construction is with n qubits. The query used is a phase query.

The algorithm uses n qubits and outputs each bit of the secret string s once measurements are made, thereby solving the BV problem in one query—a linear speedup! How does this work?

We start in the all zero state, as we always do, then perform a Hadamard gate on each qubit:

$$|0\rangle^{\otimes n} \longmapsto |+\rangle^{\otimes n} = \sum_{x\in\{0,1\}^n} |x\rangle. \tag{10.8}$$

At this point, we make a query to our function f, which writes an appropriate phase into the state, as follows:

$$\sum_{x\in\{0,1\}^n} |x\rangle \longmapsto \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \tag{10.9}$$

Using the BV property that $f(x) = x \cdot s$, we can of course write

$$\sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle = \sum_{x\in\{0,1\}^n} (-1)^{s\cdot x}|x\rangle. \tag{10.10}$$

What does this state do for us? Note that we can rewrite this state in the following way:

$$\sum_{x\in\{0,1\}^n} (-1)^{s\cdot x}|x\rangle = (|0\rangle + (-1)^{s_1}|1\rangle) \otimes (|0\rangle + (-1)^{s_2}|1\rangle) \otimes (|0\rangle + (-1)^{s_n}|1\rangle). \tag{10.11}$$

Exercise 49: Prove the equality (10.11) by expanding the right-hand side and using (10.7). It may be instructive to start with a small number of qubits, say $n = 2$, to see what's going on, then generalize it.

Note that the state (10.11) is extremely useful for us. It tells us that the jth qubit is in the plus state $|+\rangle$ if the jth bit of s is 0, otherwise the j qubit is in the minus state $|-\rangle$. Thus, all we need to ask is whether each qubit is in the plus state or the minus state, and this tells us s exactly!

Of course, we can't directly measure in the Hadamard basis—we only have access to computational basis measurements. That is, we can only ask if the qubits are $|0\rangle$ or $|1\rangle$. This isn't much of a problem though—we can just change bases to the Hadamard basis and then make a computational basis measurement there. This is what the latter part of the circuit (after the query) is doing.

If this doesn't make sense to you, we can still carry out the calculations explicitly. Let's write the state (10.11) as

$$(|0\rangle + (-1)^{s_1}|1\rangle) \otimes (|0\rangle + (-1)^{s_2}|1\rangle) \otimes (|0\rangle + (-1)^{s_n}|1\rangle) = \bigotimes_{j=1}^{n}(|0\rangle + (-1)^{s_j}|1\rangle). \tag{10.12}$$

By performing a Hadamard gate on all the qubits, we get the state

$$\bigotimes_{j=1}^{n}(|0\rangle + (-1)^{s_j}|1\rangle) \longmapsto H^{\otimes n} \bigotimes_{j=1}^{n}(|0\rangle + (-1)^{s_j}|1\rangle) = \bigotimes_{j=1}^{n} H(|0\rangle + (-1)^{s_j}|1\rangle) = \bigotimes_{j=1}^{n} |s_j\rangle. \tag{10.13}$$

Here, we use the fact that $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$. Thus, by measuring the jth qubit, we get the value s_j exactly. By measuring all the qubits, we get the secret string s exactly, thereby solving the BV problem with exactly one query in the quantum setting!

10.4 Concluding Remarks

One thing to note is that, although we used only one query, we used $O(n)$ gates. That is, we used a number of gates that scales linearly in the size of the input (which is the number of bits in the secret string s, or equivalently the domain of the function f.) Apparently, Daniel Simon didn't like this feature and set out to prove there couldn't be a "real" quantum speedup. He discovered a quantum algorithm that obtains an *exponential* speedup in the query model setting, however.

Exercise 50: State the Deustch-Jozsa problem and algorithm as a particular case of the Bernstein-Vazirani algorithm.

Exercise 51: For the Deustch-Jozsa algorithm, we presented a classical simulation using standard optical equipment to see exactly how the interference comes into play. What would an optical simulation of the Bernstein-Vazirani algorithm look like for general $n > 1$?