Basic quantum algorithms: Deutsch-Jozsa, Bernstein-Vazirani, Shor factoring classical part

Yipeng Huang

Rutgers University

February 14, 2024
Promise algorithms vs. unstructured search

Quantum algorithms offer exponential speedup in “promise” problems

A progression of related algorithms:
1. Deutsch’s
2. Deutsch-Jozsa
3. Bernstein-Vazirani
4. Simon’s
5. Shor’s

\[ f(x) = f(x \cdot \theta) \]
Table of contents

Deutsch-Jozsa algorithm: extending Deutsch’s algorithm to more qubits
    The state after applying oracle $U$
    Lemma: the Hadamard transform
    The state after the final set of Hadamards
    Probability of measuring upper register to get 0

Bernstein-Vazirani algorithm: examining the Deutsch-Jozsa outputs in more detail

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
    Factoring to modular square root
    Modular square root to discrete logarithm
    Discrete logarithm to order finding
    Order finding to period finding
Deutsch-Jozsa algorithm: Deutsch's algorithm for the $n > 1$ case

The state after the first set of Hadamards

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^\otimes n \otimes |1\rangle = |0\ldots0\rangle |1\rangle = |0\ldots01\rangle$

2. After first set of Hadamards: $|+\rangle^\otimes n \otimes |-\rangle = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes |-\rangle$
If \( f \) is constant, it will measure 0.
If balanced, it will measure something else.
Deutsch’s algorithm: Deutsch-Jozsa for the $n = 1$ case

The state after applying oracle $U$

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^\otimes n \otimes |1\rangle = |0\ldots0\rangle |1\rangle = |0\ldots01\rangle$

2. After first set of Hadamards: $|+\rangle^\otimes n \otimes |-\rangle = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes |-\rangle$

3. After applying oracle $U$:

$$U\left(|+\rangle^\otimes n \otimes |-\rangle\right) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes \left(\frac{|f(c)\rangle - |\bar{f}(c)\rangle}{\sqrt{2}}\right)$$

$$= \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$
Lemma: the Hadamard transform

\[ H^\otimes n |c\rangle = \frac{1}{2^{n/2}} \sum_{m=0}^{2^n-1} (-1)^{c \cdot m} |m\rangle \]

\[ C \cdot M \equiv (C_0 M_0 + C_1 M_1 + \ldots + C_{n-1} M_{n-1}) \mod 2 \]

\[ = C_0 \otimes C_1 \otimes \ldots \otimes C_{n-1} \]

**Try it out for** \( n = 1 \):

\[ H^\otimes 1 |c\rangle = \frac{1}{2^{1/2}} \sum_{m=0}^{2^1-1} (-1)^{c \cdot m} |m\rangle = \]

\[ \frac{1}{\sqrt{2}} (-1)^0 |0\rangle + \frac{1}{\sqrt{2}} (-1)^c |1\rangle = \begin{cases} \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle = |+\rangle & \text{if } |c\rangle = |0\rangle \\ \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle = |--\rangle & \text{if } |c\rangle = |1\rangle \end{cases} \]
\[ |c> = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \]
\[ |c\rangle = \left\{ |0\rangle, |00\cdots 0\rangle, |00\cdots 1\rangle, \ldots, |11\cdots 1\rangle \right\} \]
Deutsch-Jozsa algorithm: Deutsch’s algorithm for the \( n > 1 \) case

The state after applying oracle \( U \)

1. Initial state: \( |c\rangle \otimes |t\rangle = |0\rangle^\otimes n \otimes |1\rangle = |0 \ldots 0\rangle |1\rangle = |0 \ldots 01\rangle \)

2. After first set of Hadamards: \( |+\rangle^\otimes n \otimes |-\rangle = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes |-\rangle \)

3. After applying oracle \( U \): \( U\left(|+\rangle^\otimes n \otimes |-\rangle\right) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \)

4. After final set of Hadamards:

\[
\left( H^\otimes n \otimes I \right) \left( \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \right) \\
= \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} \left( \frac{1}{2^{n/2}} \sum_{m=0}^{2^n-1} (-1)^{c \cdot m} |m\rangle \right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \\
= \frac{1}{2^n} \sum_{c=0}^{2^n-1} \sum_{m=0}^{2^n-1} (-1)^{f(c) + c \cdot m} |m\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)
\]
Deutsch-Jozsa algorithm: Deutsch’s algorithm for the $n > 1$ case

Output of circuit is 0 iff $f$ is constant

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^\otimes n \otimes |1\rangle = |0\ldots 0\rangle |1\rangle = |0\ldots 01\rangle$

2. After first set of Hadamards: $|+\rangle^\otimes n \otimes |-\rangle = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes |-\rangle$

3. After applying oracle $U$: $U(|+\rangle^\otimes n \otimes |-\rangle) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left( \frac{|0\rangle-|1\rangle}{\sqrt{2}} \right)$

4. After final set of Hadamards: $(H^\otimes n \otimes I) \left( \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left( \frac{|0\rangle-|1\rangle}{\sqrt{2}} \right) \right) = \frac{1}{2^n} \sum_{c=0}^{2^n-1} \sum_{m=0}^{2^n-1} (-1)^{f(c)+c\cdot m} |m\rangle \otimes \left( \frac{|0\rangle-|1\rangle}{\sqrt{2}} \right)$

5. Amplitude of upper register being $|m\rangle = |0\rangle$:

\[
\frac{1}{2^n} \sum_{c=0}^{2^n-1} (-1)^{f(c)}
\]
Deutsch-Jozsa algorithm: Deutsch’s algorithm for the $n > 1$ case

Output of circuit is 0 iff $f$ is constant

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^\otimes n \otimes |1\rangle = |0 \ldots 0\rangle |1\rangle = |0 \ldots 01\rangle$

2. After first set of Hadamards: $|+\rangle^\otimes n \otimes |-\rangle = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes |-\rangle$

3. After applying oracle $U$: $U \left( |+\rangle^\otimes n \otimes |-\rangle \right) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$

4. After final set of Hadamards: $(H^\otimes n \otimes I) \left( \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \right) = \frac{1}{2^n} \sum_{c=0}^{2^n-1} \sum_{m=0}^{2^n-1} (-1)^{f(c)+c \cdot m} |m\rangle \otimes \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$

5. Amplitude of upper register being $|m\rangle = |0\rangle$: $\frac{1}{2^n} \sum_{c=0}^{2^n-1} (-1)^{f(c)}$

6. Probability of measuring upper register to get $m = 0$:

$$\left| \frac{1}{2^n} \sum_{c=0}^{2^n-1} (-1)^{f(c)} \right|^2 = \begin{cases} \left| (-1)^{f(c)} \right|^2 = 1 & \text{if } f \text{ is constant} \\ 0 & \text{if } f \text{ is balanced} \end{cases}$$
Deutsch: Algo \( n=1 \)

\[
\begin{array}{c|cccc}
& f_0 & f_1 & f_2 & f_3 \\
\hline
f(0) & 0 & 0 & 1 & 1 \\
f(1) & 0 & 1 & 0 & 1 \\
c & b & b & c
\end{array}
\]

Deutsch: Jojza Algo, e.g. \( n=2 \)

D-J Algo, e.g. \( n=3 \)

140 balanced
Table of contents

Deutsch-Jozsa algorithm: extending Deutsch’s algorithm to more qubits
   The state after applying oracle $U$
   Lemma: the Hadamard transform
   The state after the final set of Hadamards
   Probability of measuring upper register to get 0

Bernstein-Vazirani algorithm: examining the Deutsch-Jozsa outputs in more detail

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
   Factoring to modular square root
   Modular square root to discrete logarithm
   Discrete logarithm to order finding
   Order finding to period finding
Table of contents

Deutsch-Jozsa algorithm: extending Deutsch’s algorithm to more qubits
  The state after applying oracle $U$
  Lemma: the Hadamard transform
  The state after the final set of Hadamards
  Probability of measuring upper register to get 0

Bernstein-Vazirani algorithm: examining the Deutsch-Jozsa outputs in more detail

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
  Factoring to modular square root
  Modular square root to discrete logarithm
  Discrete logarithm to order finding
  Order finding to period finding
The factoring problem

One way functions for cryptography

1. Multiplying two $b$-bit numbers: on order of $b^2$ time.

2. Best known classical algorithm to factor a $b$-bit number: on order of about $2^{3\sqrt{b}}$ time.

- Makes multiplying large primes a candidate one-way function.
- It’s an open question of mathematics to prove whether one way functions exist.

Public key cryptography

Numberphile YouTube channel explanation of RSA public key cryptography:
https://www.youtube.com/watch?v=M7kEpw1tn50
The factoring problem

One way functions for cryptography

1. Multiplying two $b$-bit numbers: on order of $b^2$ time.
2. Best known classical algorithm to factor a $b$-bit number: on order of about $2^{\sqrt[3]{b}}$ time.

Quantum integer factoring algorithm

- Quantum algorithm to factor a $b$-bit number: $b^3$.
- Peter Shor, 1994.
- Important example of quantum algorithm offering exponential speedup.
Table of contents

Deutsch-Jozsa algorithm: extending Deutsch’s algorithm to more qubits
  The state after applying oracle $U$
  Lemma: the Hadamard transform
  The state after the final set of Hadamards
  Probability of measuring upper register to get 0

Bernstein-Vazirani algorithm: examining the Deutsch-Jozsa outputs in more detail

The factoring problem

Shor’s algorithm classical part: converting factoring to period finding
  Factoring to modular square root
  Modular square root to discrete logarithm
  Discrete logarithm to order finding
  Order finding to period finding
The classical part: converting factoring to order finding / period finding

General strategy for the classical part

1. Factoring
2. Modular square root
3. Discrete logarithm
4. Order finding
5. Period finding

The fact that a quantum algorithm can support all these primitives leads to additional ways that future quantum computing can be useful / threatening to existing cryptography.
Factoring

\[ N = pq \]

\[ N = 15 = 3 \times 5 \]
Modular square root

Finding the modular square root

\[ s^2 \mod N = 1 \]

\[ s = \sqrt{1} \mod N \]

Trivial roots would be \( s = \pm 1 \).

- Are there other (nontrivial) square roots?
- For \( N = 15 \), \( s = \pm 4, s = \pm 11, s = \pm 14 \) are all nontrivial square roots. (Show this).
- Later in these slides, we will see how nontrivial square roots are useful for factoring.
Discrete log

1. Pick a that is relatively prime with N.
2. Efficient to test if relatively prime by finding GCD using Euclid’s algorithm. For example, a=6 and n=15.

Exercise: list the possible a’s for N = 15.
1. Pick $a$ that is relatively prime with $N$.
2. Efficient to test if relatively prime by finding GCD using Euclid’s algorithm. For example, $a = 6$ and $n = 15$.

So now our factoring problem is:

$$a^r \mod N = 1$$

$$a^r \equiv 1 \mod N$$

In fact, this algorithm for finding discrete log even more directly attacks other crypto primitives such as Diffie-Hellman key exchange.
Our discrete log problem is equivalent to order finding.

<table>
<thead>
<tr>
<th></th>
<th>( a^1 \mod 15 )</th>
<th>( a^2 \mod 15 )</th>
<th>( a^3 \mod 15 )</th>
<th>( a^4 \mod 15 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a=2 )</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>( a=4 )</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>( a=7 )</td>
<td>7</td>
<td>4</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>( a=8 )</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( a=11 )</td>
<td>11</td>
<td>1</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>( a=13 )</td>
<td>13</td>
<td>4</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>( a=14 )</td>
<td>14</td>
<td>1</td>
<td>14</td>
<td>1</td>
</tr>
</tbody>
</table>

Find smallest \( r \) such that \( a^r \equiv 1 \mod N \)
Period finding

In other words, the problem by now can also be phrased as finding the period of a function.

\[ f(x) = f(x + r) \]

Where

\[ f(x) = a^x = a^{x+r} \mod N \]

Find \( r \).
What to do after quantum algorithm gives you $r$

- If $r$ is odd or if $a^{\frac{r}{2}} + 1 \equiv 0 \mod N$, abandon.
- There is separate theorem saying no more than a quarter of trials would have to be tossed.

Exercise: try for $a = 14$. 
What to do after quantum algorithm gives you $r$

- If $r$ is odd or if $a^\frac{r}{2} + 1 \equiv 0 \mod N$, abandon.
- There is separate theorem saying no more than a quarter of trials would have to be tossed.

Exercise: try for $a = 14$.

Otherwise, factors are $\text{GCD}( a^\frac{r}{2} \pm 1, N )$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$r$</th>
<th>$a^\frac{r}{2} \pm 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>$2^2 \pm 1 = 4 \pm 1$</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>$4^1 \pm 1 = 4 \pm 1$</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>$7^2 \pm 1 = 49 \pm 1$</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>$8^2 \pm 1 = 64 \pm 1$</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>$11^1 \pm 1 = 11 \pm 1$</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>$13^2 \pm 1 = 169 \pm 1$</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>$14^2 \pm 1 = 196 \pm 1$ (bad case)</td>
</tr>
</tbody>
</table>

Notice why we discarded 14.
Proof why this works and why factoring is modular square root

\[ a^r \equiv 1 \pmod{N} \]

So now \( a^{\frac{r}{2}} \) is a nontrivial square root of 1 mod N.

\[ a^r - 1 \equiv 0 \pmod{N} \]

\[ (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \pmod{N} \]

The above implies that

\[ \frac{(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)}{N} \]

is an integer. So now we have to prove that

1. \( \frac{a^{\frac{r}{2}} - 1}{N} \) is not an integer, and
2. \( \frac{a^{\frac{r}{2}} + 1}{N} \) is not an integer.
Proof why this works and why factoring is modular square root

Suppose \( \frac{a^r - 1}{N} \) is an integer

that would imply

\[
a^r - 1 \equiv 0 \pmod{N} \\
a^r \equiv 1 \pmod{N}
\]

but we already defined \( r \) is the smallest such that \( a^r \equiv 1 \pmod{N} \), so there is a contradiction, so \( \frac{a^r - 1}{N} \) is not an integer.

Suppose \( \frac{a^r + 1}{N} \) is an integer

that would imply

\[
a^r + 1 \equiv 0 \pmod{N}
\]

but we already eliminated such cases because we know this doesn’t give us a useful result.