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:

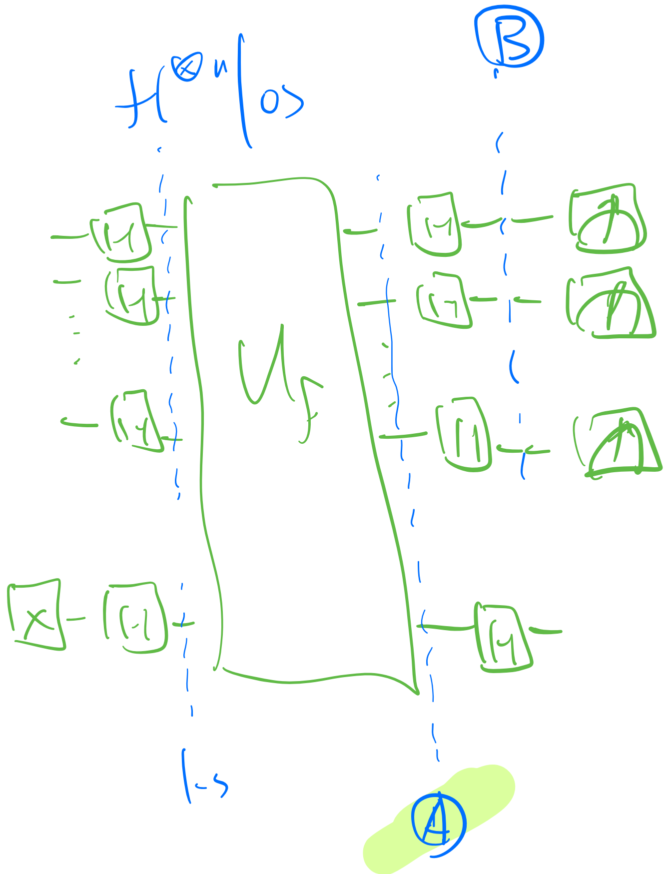
1. Deutsch's
2. Deutsch-Jozsa
3. Bernstein-Vazirani
4. Simon's
5. Shor's

Table of contents

Deutsch-Jozsa algorithm: Deutsch's algorithm for the $n > 1$ case

The state after the first set of Hadamards

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^{\otimes n} \otimes |1\rangle = |0 \ldots 0\rangle |1\rangle = |0 \ldots 01\rangle$
2. After first set of Hadamards: $|+\rangle^{\otimes n} \otimes |-\rangle = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} |c\rangle \otimes |-\rangle$



if f is constant
will measure "0"
if balanced,
will measure
something else

Deutsch's algorithm: Deutsch-Jozsa for the $n = 1$ case

The state after applying oracle U

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^{\otimes n} \otimes |1\rangle = |0\dots0\rangle\,|1\rangle = |0\dots01\rangle$
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:

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

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

Lemma: the Hadamard transform

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

$c \cdot m = (c_0 m_0 + c_1 m_1 + \cdots + c_{n-1} m_{n-1}) \mod 2.$

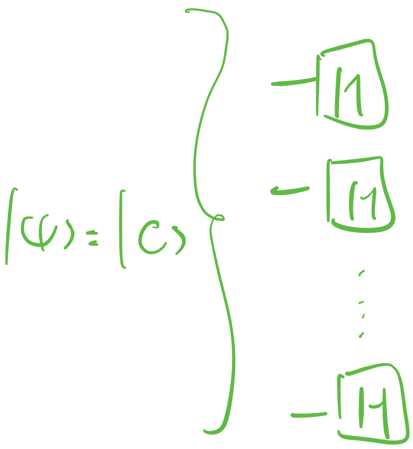
$= c_0 m_0 \oplus c_1 m_1 \oplus \cdots \oplus c_{n-1} m_{n-1}$

- $$\begin{aligned} & H^{\otimes n}|c\rangle \\ &= H|c_0\rangle \otimes H|c_1\rangle \otimes \ldots \otimes H|c_{n-1}\rangle \\ &= \frac{1}{\sqrt{2}}\Big(|0\rangle + (-1)^{c_0}|1\rangle\Big) \otimes \frac{1}{\sqrt{2}}\Big(|0\rangle + (-1)^{c_1}|1\rangle\Big) \otimes \ldots \otimes \frac{1}{\sqrt{2}}\Big(|0\rangle + (-1)^{c_{n-1}}|1\rangle\Big) \\ &= \frac{1}{2^{n/2}} \sum_{m=0}^{2^n-1} (-1)^{c_0 m_0 + c_1 m_1 + \ldots + c_{n-1} m_{n-1} \mod 2} |m\rangle \end{aligned}$$

- Try it out for $n = 1$: $H^{\otimes 1}|c\rangle = \frac{1}{2^{1/2}} \sum_{m=0}^{2^1-1} (-1)^{c \cdot m} |m\rangle =$
$$\frac{1}{\sqrt{2}}(-1)^0|0\rangle + \frac{1}{\sqrt{2}}(-1)^c|1\rangle = \begin{cases} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle & \text{if } |c\rangle = |0\rangle \\ \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |-\rangle & \text{if } |c\rangle = |1\rangle \end{cases}$$

$$\boxed{H} \otimes \boxed{H} \otimes \cdots \otimes \boxed{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{pmatrix}$$

$$|c\rangle - \boxed{H} - \boxed{H} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$



e.g. $|c\rangle = |00\cdots0\rangle,\ |00\cdots1\rangle \ldots |11\cdots1\rangle$

Deutsch-Jozsa algorithm: Deutsch's algorithm for the $n > 1$ case

The state after applying oracle U

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^{\otimes n} \otimes |1\rangle = |0\ldots0\rangle\,|1\rangle = |0\ldots01\rangle$
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: $U\Big(|+\rangle^{\otimes n} \otimes |-\rangle\Big) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$
4. After final set of Hadamards:

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

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

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

Deutsch-Jozsa algorithm: Deutsch's algorithm for the $n > 1$ case

Output of circuit is 0 iff f is constant

1. Initial state: $|c\rangle \otimes |t\rangle = |0\rangle^{\otimes n} \otimes |1\rangle = |0\ldots0\rangle\,|1\rangle = |0\ldots01\rangle$
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: $U\Big(|+\rangle^{\otimes n} \otimes |-\rangle\Big) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)} |c\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$
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: $|c\rangle \otimes |t\rangle = |0\rangle^{\otimes n} \otimes |1\rangle = |0\ldots0\rangle\,|1\rangle = |0\ldots01\rangle$
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: $U\Big(|+\rangle^{\otimes n} \otimes |-\rangle\Big) = \frac{1}{2^{n/2}} \sum_{c=0}^{2^n-1} (-1)^{f(c)}\,|c\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$
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 $n=1$

	f_0	f_1	f_2	f_3
$f(0)$	0	0	1	1
$f(1)$	0	1	0	1
	c	b	b	c

Deutsch Jozsa Algo, e.g. $n=2$

	f_0				f_4				f_8				f_{12}			
$f(00)$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$f(01)$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$f(10)$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$f(11)$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	c			b		b	b			b	b		b			c

D-J Algo e.g. $n=3$

	f_0	f_1	...	f_{15}	...	f_{255}
$f(000)$	0	0		0		1
$f(001)$	0	0		0		1
$f(010)$	0	0		0		1
$f(011)$	0	0		0		1
$f(100)$	0	0		1		1
$f(101)$	0	0		1		1
$f(110)$	0	0		1		1
$f(111)$	0	1		1		1
	c	x		b		c

140 balanced

Table of contents

1) conventional (analysis)
2) stabilizers
3) ZX
4) significance vs. DJ -

Table of contents

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.

- 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

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 $N = 15$.

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 $n = 15$.

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.

Order finding

Our discrete log problem is equivalent to order finding.

	$a^1 \mod 15$	$a^2 \mod 15$	$a^3 \mod 15$	$a^4 \mod 15$
a=2	2	4	8	1
a=4	4	1	4	1
a=7	7	4	13	1
a=8	8	4	2	1
a=11	11	1	11	1
a=13	13	4	7	1
a=14	14	1	14	1

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

a=2	r=4	$2^2 \pm 1 = 4 \pm 1$
a=4	r=2	$4^1 \pm 1 = 4 \pm 1$
a=7	r=4	$7^2 \pm 1 = 49 \pm 1$
a=8	r=4	$8^2 \pm 1 = 64 \pm 1$
a=11	r=2	$11^1 \pm 1 = 11 \pm 1$
a=13	r=4	$13^2 \pm 1 = 169 \pm 1$
a=14	r=2	$14^2 \pm 1 = 196 \pm 1$ (bad case)

Notice why we discarded 14.

Proof why this works and why factoring is modular square root

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

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

$$a^r - 1 \equiv 0 \mod N$$

$$(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \mod N$$

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 $\frac{a^{\frac{r}{2}}-1}{N}$ is an integer

that would imply

$$a^{\frac{r}{2}} - 1 \equiv 0 \mod N$$

$$a^{\frac{r}{2}} \equiv 1 \mod N$$

but we already defined r is the smallest such that $a^r \equiv 1 \mod N$, so there is a contradiction, so $\frac{a^{\frac{r}{2}}-1}{N}$ is not an integer.

Suppose $\frac{a^{\frac{r}{2}}+1}{N}$ is an integer

that would imply

$$a^{\frac{r}{2}} + 1 \equiv 0 \mod N$$

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