Quantum algorithms

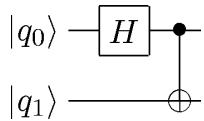
Quantum computing

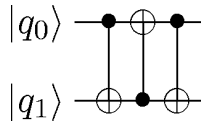
G. Chênevert

Feb. 5, 2021

JUNIA ISEN

Last time





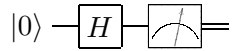
Quantum algorithms

First quantum algorithms

Quantum complexity

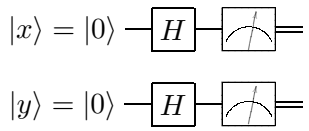
True random number generator

On 1 bit:



outputs 0 or 1 with probability $\frac{1}{2}$

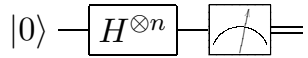
On 2 bits:



outputs 0, 1, 2 or 3 with probability $\frac{1}{4}$

True random number generator

In general



ouputs any integer in $[\![0, 2^n[\![$ with probability $\frac{1}{2^n}$.

$$\underbrace{H|0\rangle \otimes \cdots \otimes H|0\rangle}_{n} = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle$$

Boolean functions

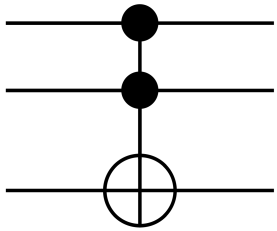
So far we know how to build quantum circuits computing NOT and XOR.

What about other logical gates, *e.g.* AND and OR?

Recall that quantum gates are represented by unitary matrices, *hence always reversible*.

$\Longrightarrow$ ancillary qubits are needed

The Toffoli (or CCNOT) gate



$$\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & & 1 \\ & & & & & & 1 & \end{bmatrix}$$

Reversible evaluation

If $f : \{0,1\}^n \to \{0,1\}$ is any boolean function

consider the gate U_f on $n+1$ qubits defined by

$$U_f : |\mathbf{x}\rangle \otimes |y\rangle \mapsto |\mathbf{x}\rangle \otimes |y \oplus f(\mathbf{x})\rangle.$$

You may check that this is a unitary operator on $\mathcal{V}_{2^n} \otimes \mathcal{V}_2 = (\mathcal{V}_2)^{\otimes(n+1)}$

that allows to evaluate f on $\mathbf{x}$ without losing $\mathbf{x}$:

$$U_f(|\mathbf{x}\rangle|0\rangle) = |x\rangle|f(\mathbf{x})\rangle.$$

Fact

For any boolean function $f : \{0,1\}^n \to \{0,1\}$

the quantum gate U_f can be implemented using only Toffoli and X gates.

Classical analogue: any boolean function can be generated using only the NAND gate

$$\text{CCNOT}(|x\rangle\,|y\rangle\,|1\rangle) = |x\rangle\,|y\rangle\,|\text{NAND}(x,y)\rangle$$

Example : the Deutsh algorithm

There exists 4 boolean functions f of a single variable: two of them are constant

$$f(x) \equiv 0, \qquad f(x) \equiv 1, \qquad \text{(type 0)}$$

while the two others are not:

$$f(x) = x, \qquad f(x) = 1 \oplus x. \qquad \text{(type 1)}$$

Example : the Deutsch algorithm

Suppose you are given one of those four functions f (as a black box: the only thing you can do is evaluate the function on inputs of your choice) and asked what type it is.

In the classical world, clearly two evaluations of f are needed (and sufficient).

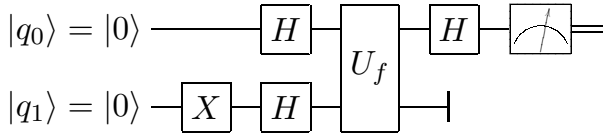
`input` f

`output` $f(0) \oplus f(1)$

The Deutsch algorithm finds out the type of f with *a single quantum evaluation.*

Example : the Deutsch algorithm

The following circuit computes the type of f:



(Can check how the circuit reduces for each of the 4 possible U_f)

Quantum algorithms

First quantum algorithms

Quantum complexity

Faster computations

The whole point of quantum computing is that some quantum algorithms exhibit

quantum advantage: *i.e.* run "faster" than the best known classical algorithms

In extreme cases, this leads to

quantum supremacy: *i.e.* the ability to compute things that could never practically be achieved with classical computers.

Oct. 2019: Sampling random quantum circuits on 53 qubits (Google)

Dec. 2020: Quantum computational advantage using photons

Complexity of an algorithm

Classically: the complexity of an algorithm $\mathcal{A}$ is a bound on the number of computing steps needed for an input of a given size

i.e. the function

$$n \mapsto \max_{|x|=n} N_{\mathcal{A}}(x)$$

where $N_{\mathcal{A}}(x)$ denotes the number of steps used to perform $\mathcal{A}$ on x in a given computing model (usually: Turing machines)

Quantum computing model

There exist a (rather inconvenient) notion of quantum Turing machine

Most people prefer to use the (equivalent) **quantum circuit model**:

- given input x: a circuit $U_{\mathcal{A},x}$ made out of quantum gates taken from a standard generating set is prepared
- the output y is the result of the measure of $U_{\mathcal{A},x}|0\rangle$
- (then some classical post-processing may be applied)

The **complexity** of the quantum algorithm is a bound on the number of gates needed:

$$n \mapsto \max_{|x|=n} |U_{\mathcal{A},x}|.$$

The Deutsch-Jozsa problem

Given a boolean function $f : \{0,1\}^n \to \{0,1\}$ assumed to be either

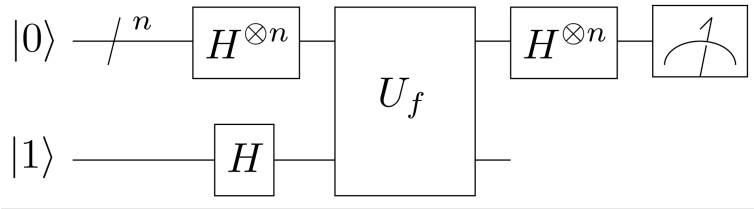
- constant: $f(x) \equiv 0$ or $f(x) \equiv 1$ ("type 0") or
- balanced: $|\{x \,|\, f(x) = 0\}| = |\{x \,|\, f(x) = 1\}| = 2^{n-1}$ ("type 1"),

problem: compute its type.

A classical algorithm needs at least $2^{n-1} + 1$ evaluations of f to decide

$\Longrightarrow$ **EXP** (EXPonential time) complexity class

The Deutsch-Jozsa algorithm



Proposition: output is $|0\rangle^{\otimes n} \iff f$ is constant

EQP (Exact Quantum Polynomial time) complexity class

$\implies$ exponential speedup !

Deutsch-Jozsa: remarks

- Here U_f is considered an **oracle** for f (black box implementation)
- Classical decision algorithm: $2^{n-1}+1$ evaluations are required to guarantee the answer... but we can get a probable answer with much less evaluations.
- With k evaluations, assuming constant and balanced functions are equiprobable:
  - if not all values are equal, f is certainly balanced
  - if all values are equal, f is constant with probability

$$\geq \frac{1}{1+2^{1-k}} \geq \frac{2}{3} \qquad \text{for } k \geq 2$$

Probabilistic algorithms

In practice: we prefer a fast algorithm with a good probability of giving a right answer to a slow algorithm that is always right!

Example: Rabin-Miller *vs* AKS primality tests

The Deutsch-Jozsa problem is in the

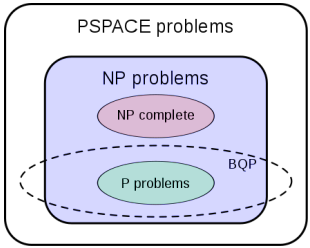
BPP (Bounded-error Probabilistic Polynomial time) complexity class

But since quantum algorithms are typically also probabilistic...

The speedup here is not so impressive after all!

Aside: the Complexity Zoo

- We know that **P** $\subseteq$ **NP** (Nondeterministic Polynomial time)
- Whether the inclusion is strict is an open question ($1,000,000)
- We also know that **P** $\subseteq$ **BPP** $\subseteq$ **BQP** (Bounded-error Quantum Polynomial time)
- Reverse inclusions *also unknown*; complexity classes are a real zoo



Upcoming

Two algorithms displaying a clear quantum advantage over the classical counterparts:

- **Grover**'s algorithm

  unstructured search among n items in $\mathcal{O}(\sqrt{n})$

- **Shor**'s algorithm

  factoring a n-bit integer in $\mathcal{O}(n^3 \log n)$