Introduction to Quantum Computing
Lecture 7: Complexity and Quantum Algorithms I

Petros Wallden

School of Informatics, University of Edinburgh

9th October 2018

Notation & Definitions

- Computational Complexity: Classification of problems according to their difficulty. We usually measure the **amount of resources** (e.g . time, space, gates) used by an algorithm as a function of the **input size** n.

Notation & Definitions

- Computational Complexity: Classification of problems according to their difficulty. We usually measure the **amount of resources** (e.g . time, space, gates) used by an algorithm as a function of the **input size** n.
- Complexity class: Is a set of problems with related resource-based complexity

Notation & Definitions

- Computational Complexity: Classification of problems according to their difficulty. We usually measure the **amount of resources** (e.g . time, space, gates) used by an algorithm as a function of the **input size** n.
- Complexity class: Is a set of problems with related resource-based complexity
- Notation:

- $f(n)$ is in $O(g(n))$ if for some constant m there exists a positive constant such that $f(n) \leq cg(n)$ for all $n \geq m$
- $f(n)$ is in $\Omega(g(n))$ if for some constant m there exists a positive constant c such that $f(n) \geq cg(n)$ for all $n \geq m$
- $f(n)$ is in $\Theta(g(n))$ if for some constant m there exists positive constants $c_1 \leq c_2$ such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq m$

Decision problems:

1. P: Solved by a deterministic Turing machine in polynomial time (classical deterministic computer solves efficiently)

Decision problems:

1. P: Solved by a deterministic Turing machine in polynomial time (classical deterministic computer solves efficiently)
2. NP: *Verifiable* in polynomial time by deterministic Turing machine.
   Alternatively, the "yes" instances can be accepted in polynomial time by a non-deterministic Turing machine

Decision problems:

1. P: Solved by a deterministic Turing machine in polynomial time (classical deterministic computer solves efficiently)
2. NP: *Verifiable* in polynomial time by deterministic Turing machine.
   Alternatively, the “yes” instances can be accepted in polynomial time by a non-deterministic Turing machine
3. PSPACE: Solved by Turing machine using polynomial amount of space (irrespective of the time needed)

Some Useful Complexity Classes

Decision problems:

1. P: Solved by a deterministic Turing machine in polynomial time (classical deterministic computer solves efficiently)
2. NP: *Verifiable* in polynomial time by deterministic Turing machine.
   Alternatively, the “yes” instances can be accepted in polynomial time by a non-deterministic Turing machine
3. PSPACE: Solved by Turing machine using polynomial amount of space (irrespective of the time needed)
4. BPP: Solved by probabilistic TM in poly time with bounded error (classical probabilistic computer solves efficiently)

Some Useful Complexity Classes

Decision problems:

1. P: Solved by a deterministic Turing machine in polynomial time (classical deterministic computer solves efficiently)
2. NP: *Verifiable* in polynomial time by deterministic Turing machine.
   Alternatively, the “yes” instances can be accepted in polynomial time by a non-deterministic Turing machine
3. PSPACE: Solved by Turing machine using polynomial amount of space (irrespective of the time needed)
4. BPP: Solved by probabilistic TM in poly time with bounded error (classical probabilistic computer solves efficiently)
5. BQP: Solved by probabilistic *quantum computer* in poly time with bounded error (quantum computer solves efficiently)

Complexity Classes Relations

Proven Relations:

- $\mathrm{BQP} \subseteq \mathrm{PSPACE}$. Problems solved by quantum computers can be solved (potentially inefficiently) by classical computers

Complexity Classes Relations

Proven Relations:

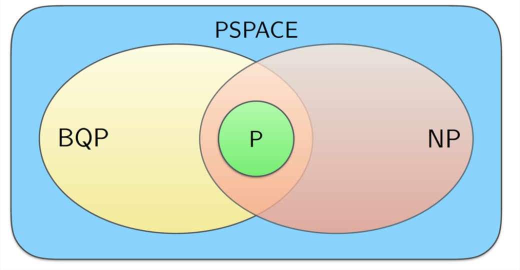
- $\mathrm{BQP} \subseteq \mathrm{PSPACE}$. Problems solved by quantum computers can be solved (potentially inefficiently) by classical computers
- $\mathrm{BPP} \subseteq \mathrm{BQP}$. Quantum computers are at least as efficient as probabilistic Turing machines

Complexity Classes Relations

Proven Relations:

- $\mathrm{BQP} \subseteq \mathrm{PSPACE}$. Problems solved by quantum computers can be solved (potentially inefficiently) by classical computers
- $\mathrm{BPP} \subseteq \mathrm{BQP}$. Quantum computers are at least as efficient as probabilistic Turing machines

Conjectured Relations: (based on other plausible assumptions)



Complexity Classes Relations

Proven Relations:

- $\mathrm{BQP} \subseteq \mathrm{PSPACE}$. Problems solved by quantum computers can be solved (potentially inefficiently) by classical computers
- $\mathrm{BPP} \subseteq \mathrm{BQP}$. Quantum computers are at least as efficient as probabilistic Turing machines

Conjectured Relations: (based on other plausible assumptions)

1. There are problems outside NP that quantum computers **can** solve
2. There are problems in NP that quantum computers **cannot** solve (therefore NP-complete problems should be outside BQP)

The Church - Turing Theses

(1) Church-Turing Thesis (original)

All physical computable functions can be computed by a **Turing machine**

(1) Church-Turing Thesis (original)

All physical computable functions can be computed by a **Turing machine**

- No reference to the efficiency of the computation
- Quantum computers do not affect this statement ($\mathrm{BQP} \subseteq \mathrm{PSPACE}$)

The Church - Turing Theses

(1) Church-Turing Thesis (original)

All physical computable functions can be computed by a **Turing machine**

- No reference to the efficiency of the computation
- Quantum computers do not affect this statement ($\mathrm{BQP} \subseteq \mathrm{PSPACE}$)

(2) Church-Turing Thesis (computational complexity)

A probabilistic Turing machine can **efficiently** simulate any realistic model of computation

The Church - Turing Theses

(1) Church-Turing Thesis (original)

All physical computable functions can be computed by a **Turing machine**

- No reference to the efficiency of the computation
- Quantum computers do not affect this statement ($\mathrm{BQP} \subseteq \mathrm{PSPACE}$)

(2) Church-Turing Thesis (computational complexity)

A probabilistic Turing machine can **efficiently** simulate any realistic model of computation

- If as conjectured $\mathrm{BPP} \subset \mathrm{BQP}$ then **(2) is wrong**!

The Church - Turing Theses

(1) Church-Turing Thesis (original)

All physical computable functions can be computed by a **Turing machine**

- No reference to the efficiency of the computation
- Quantum computers do not affect this statement ($\mathrm{BQP} \subseteq \mathrm{PSPACE}$)

(2') Church-Turing Thesis (quantum)

A **quantum** Turing machine can **efficiently** simulate any realistic model of computation

- We are given a classical gate corresponding to an unknown function f as a **black box** (oracle)

The Oracle Model

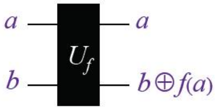
- We are given a classical gate corresponding to an unknown function f as a **black box** (oracle)

- Access: Query the oracle, i.e. insert x and obtain $f(x)$

- We are given a classical gate corresponding to an unknown function f as a **black box** (oracle)

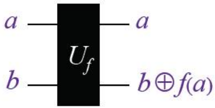
- Access: Query the oracle, i.e. insert x and obtain $f(x)$
- Goal: Determine properties of the function f with the fewest queries to the oracle

- We are given a quantum gate corresponding to an unknown classical function f as a **black box** (oracle) acting on two qubits in the following way:



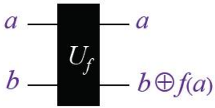
The Quantum Oracle Model

- We are given a quantum gate corresponding to an unknown classical function f as a **black box** (oracle) acting on two qubits in the following way:



- Access: Query the quantum oracle, i.e. insert $|a\rangle |b\rangle$ and obtain $|a\rangle |b \oplus f(a)\rangle$

- We are given a quantum gate corresponding to an unknown classical function f as a **black box** (oracle) acting on two qubits in the following way:



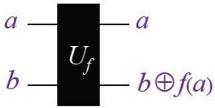
- Access: Query the quantum oracle, i.e. insert $|a\rangle\,|b\rangle$ and obtain $|a\rangle\,|b \oplus f(a)\rangle$

  By linearity, we can also query in superposition:

$$\sum_{a,b} C_{a,b}\,|a\rangle\,|b\rangle \to \sum_{a,b} C_{a,b}\,|a\rangle\,|b \oplus f(a)\rangle$$

The Quantum Oracle Model

- We are given a quantum gate corresponding to an unknown classical function f as a **black box** (oracle) acting on two qubits in the following way:



- Access: Query the quantum oracle, i.e. insert $|a\rangle\,|b\rangle$ and obtain $|a\rangle\,|b \oplus f(a)\rangle$

  By linearity, we can also query in superposition:

$$\sum_{a,b} C_{a,b}\,|a\rangle\,|b\rangle \rightarrow \sum_{a,b} C_{a,b}\,|a\rangle\,|b \oplus f(a)\rangle$$

- Goal: Determine properties of the classical function f with the fewest queries to the quantum oracle

The Deutsch - Jozsa Algorithm

- Inspiration for Shor's and Grover's algorithms

The Deutsch - Jozsa Algorithm

- Inspiration for Shor's and Grover's algorithms
- Initial protocol by **Deutsch** 1985, improved by **Jozsa**. Current version, is result of further research (**Cleve**, **Ekert**, **Macchiavello** and **Mosca**)

The Deutsch - Jozsa Algorithm

- **Input:** A boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$

- **Input:** A boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$
- **Promise:** The function is either **constant** or **balanced**

- **Input:** A boolean function $f : \{0,1\}^n \to \{0,1\}$
- **Promise:** The function is either **constant** or **balanced**

Constant: $f(x) = c \; \forall \; x$ and $c = 0$ or 1

Balanced: $|f^{-1}(0)| = |f^{-1}(1)|$ i.e. half inputs give 0 and half give 1

The Deutsch - Jozsa Algorithm

- **Input:** A boolean function $f : \{0,1\}^n \to \{0,1\}$
- **Promise:** The function is either **constant** or **balanced**

  Constant: $f(x) = c \; \forall \; x$ and $c = 0$ or 1

  Balanced: $|f^{-1}(0)| = |f^{-1}(1)|$ i.e. half inputs give 0 and half give 1
- **Output:** **Determine** whether the function is **constant** or is **balanced** with the **fewest queries**

The Deutsch - Jozsa Algorithm

- **Input:** A boolean function $f : \{0,1\}^n \to \{0,1\}$
- **Promise:** The function is either **constant** or **balanced**

  Constant: $f(x) = c \; \forall \; x$ and $c = 0$ or 1

  Balanced: $|f^{-1}(0)| = |f^{-1}(1)|$ i.e. half inputs give 0 and half give 1
- **Output:** **Determine** whether the function is **constant** or is **balanced** with the **fewest queries**
- **Performance:**
  1. Classical: To know with certainty we need at least $2^n/2 + 1$ queries
  2. Quantum: With a **single** oracle query

- Recall that U_f is defined as:

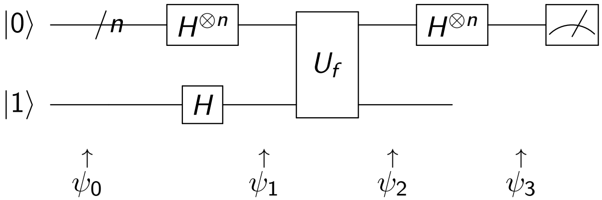
$$\sum_{x,y} C_{x,y} |x\rangle |y\rangle \rightarrow \sum_{x,y} C_{x,y} |x\rangle |y \oplus f(x)\rangle$$

The Deutsch - Jozsa Algorithm

- Recall that U_f is defined as:

$$\sum_{x,y} C_{x,y} |x\rangle |y\rangle \rightarrow \sum_{x,y} C_{x,y} |x\rangle |y \oplus f(x)\rangle$$

- The Quantum Circuit of the algorithm is given by:



The Deutsch - Jozsa Algorithm

Property:

$$H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{x}|1\rangle\right)$$

Property:

$H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^x|1\rangle\right)$

The protocol's steps:

1. $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$. Note that the first register refers to string of n qubits, while the second register is a single qubit.

Property:

$H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle)$

The protocol's steps:

1. $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$. Note that the first register refers to string of n qubits, while the second register is a single qubit.
2. Apply H to all qubits:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|0\rangle - |1\rangle)$$

Property:

$H|x\rangle = \frac{1}{\sqrt{2}} \sum_{y\in\{0,1\}} (-1)^{xy} |y\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^x |1\rangle \right)$

The protocol's steps:

1. $|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle$. Note that the first register refers to string of n qubits, while the second register is a single qubit.
2. Apply H to all qubits:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \left(|0\rangle - |1\rangle \right)$$

3. Apply the oracle U_f:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \left(|f(x)\rangle - |1 \oplus f(x)\rangle \right)$$

Property:
$H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle)$

The protocol's steps:

1. $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$. Note that the first register refers to string of n qubits, while the second register is a single qubit.
2. Apply H to all qubits:
$$|\psi_1\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|0\rangle - |1\rangle)$$
3. Apply the oracle U_f:
$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle(|f(x)\rangle - |1\oplus f(x)\rangle)$$
it can be rewritten as:
$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$$

4. Apply $H^{\otimes n}$ to the first n qubits:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{y=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}\right)|y\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

where $x \cdot y = x_0 y_0 \oplus x_1 y_1 \oplus \cdots \oplus x_{n-1} y_{n-1}$ is the sum of the bitwise product.

4. Apply $H^{\otimes n}$ to the first n qubits:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{y=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}\right)|y\rangle\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

where $x\cdot y = x_0y_0\oplus x_1y_1\oplus\cdots\oplus x_{n-1}y_{n-1}$ is the sum of the bitwise product.

5. We measure the first n qubits in the computational basis and we examine the probability of obtaining all zero's ($|0\rangle^{\otimes n}$):

$$p(0) = |\frac{1}{2^n}\sum_{0}^{2^n-1}(-1)^{f(x)}|^2 \tag{1}$$

4. Apply $H^{\otimes n}$ to the first n qubits:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{y=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}\right)|y\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

   where $x \cdot y = x_0 y_0 \oplus x_1 y_1 \oplus \cdots \oplus x_{n-1} y_{n-1}$ is the sum of the bitwise product.

5. We measure the first n qubits in the computational basis and we examine the probability of obtaining all zero's ($|0\rangle^{\otimes n}$):

$$p(0) = |\frac{1}{2^n}\sum_{0}^{2^n-1}(-1)^{f(x)}|^2 \tag{1}$$

   If $f(x)$ is constant, all terms have the same sign and Eq. (1) gives $p(0) = 1$

4. Apply $H^{\otimes n}$ to the first n qubits:

$$|\psi_3\rangle = \frac{1}{2^n}\sum_{y=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot y}\right)|y\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

where $x \cdot y = x_0 y_0 \oplus x_1 y_1 \oplus \cdots \oplus x_{n-1}y_{n-1}$ is the sum of the bitwise product.

5. We measure the first n qubits in the computational basis and we examine the probability of obtaining all zero's ($|0\rangle^{\otimes n}$):

$$p(0) = |\frac{1}{2^n}\sum_{0}^{2^n-1}(-1)^{f(x)}|^2 \tag{1}$$

If $f(x)$ is constant, all terms have the same sign and Eq. (1) gives $p(0) = 1$

If $f(x)$ is balanced, half terms are $+1$ and half terms are -1 resulting to Eq. (1) giving $p(0) = 0$

- The algorithm is **deterministic**. Belongs to EQP (Exact Quantum Polynomial time) which is the quantum version of P (rather than in BQP which is the quantum version of BPP).

The Deutsch - Jozsa Algorithm

- The algorithm is **deterministic**. Belongs to EQP (Exact Quantum Polynomial time) which is the quantum version of P (rather than in BQP which is the quantum version of BPP).
- It constitutes the first **exponential quantum speed-up**. From exponential many oracle calls for P algorithms, we succeed with a single oracle query in EQP!

The Deutsch - Jozsa Algorithm

- The algorithm is **deterministic**. Belongs to EQP (Exact Quantum Polynomial time) which is the quantum version of P (rather than in BQP which is the quantum version of BPP).
- It constitutes the first **exponential quantum speed-up**. From exponential many oracle calls for P algorithms, we succeed with a single oracle query in EQP!
- It does not give a speed-up compared to BPP, since if we allow for (small) probability of failure, there exist efficient classical algorithms with constant oracle calls (example?)

Property:

$$\sum_{y=0}^{N-1} \exp\left(\frac{2\pi i}{N}(x - x')y\right) = N\delta_{x,x'} \tag{2}$$

Property:

$$\sum_{y=0}^{N-1} \exp\left(\frac{2\pi i}{N}(x-x')y\right) = N\delta_{x,x'} \tag{2}$$

Notation:

1. $|x\rangle = |x_1 x_2 \cdots x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$, where $x := x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n 2^0$
2. $[0.x_1 \cdots x_m] = \sum_{k=1}^{m} x_k 2^{-k}$, e.g. $[0.x_1 x_2] = \frac{x_1}{2} + \frac{x_2}{2^2}$

Property:

$$\sum_{y=0}^{N-1} \exp\left(\frac{2\pi i}{N}(x-x')y\right) = N\delta_{x,x'} \qquad (2)$$

Notation:

1. $|x\rangle = |x_1 x_2 \cdots x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$, where $x := x_1 2^{n-1} + x_2 2^{n-2} + \cdots + x_n 2^0$
2. $[0.x_1 \cdots x_m] = \sum_{k=1}^{m} x_k 2^{-k}$, e.g. $[0.x_1 x_2] = \frac{x_1}{2} + \frac{x_2}{2^2}$

- **Definition** (classical): The **Discrete Fourier Transform** (DFT) takes a N-dimensional complex vector $(a_0, \cdots, a_{N-1})$ and maps it to $(b_0, \cdots, b_{N-1})$ in this way:

$$b_y = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} a_x \exp(2\pi i x y / N) \qquad (3)$$

Quantum Fourier Transform

- **Definition** (quantum): The **Quantum Fourier Transform** (QFT) is a unitary operation F that performs DFT to the **amplitudes** of a quantum state:

$$F\sum_{x=0}^{N-1} a_x |x\rangle = \sum_{y=0}^{N-1} b_y |y\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\sum_{x=0}^{N-1} a_x \exp(2\pi ixy/N) |y\rangle$$

where the amplitudes a_x, b_y are related as in DFT.

Quantum Fourier Transform

- **Definition** (quantum): The **Quantum Fourier Transform** (QFT) is a unitary operation F that performs DFT to the **amplitudes** of a quantum state:

$$F \sum_{x=0}^{N-1} a_x |x\rangle = \sum_{y=0}^{N-1} b_y |y\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} a_x \exp(2\pi i x y / N) |y\rangle$$

  where the amplitudes a_x, b_y are related as in DFT.
- To determine a quantum operation it suffices to know how it acts on computational basis state and then extend by linearity

- **Definition** (quantum): The **Quantum Fourier Transform** (QFT) is a unitary operation F that performs DFT to the **amplitudes** of a quantum state:

$$F\sum_{x=0}^{N-1} a_x |x\rangle = \sum_{y=0}^{N-1} b_y |y\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\sum_{x=0}^{N-1} a_x \exp(2\pi i xy/N)|y\rangle$$

  where the amplitudes a_x, b_y are related as in DFT.
- To determine a quantum operation it suffices to know how it acts on computational basis state and then extend by linearity
- The QFT acts as (note that $N = 2^n$):

$$F|x_1x_2\cdots x_n\rangle = \frac{1}{\sqrt{N}}\left(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle\right)\otimes\left(|0\rangle + e^{2\pi i[0.x_{n-1}x_n]}|1\rangle\right)\otimes \otimes\cdots\otimes\left(|0\rangle + e^{2\pi i[0.x_1x_2\cdots x_n]}|1\rangle\right) \quad (4)$$

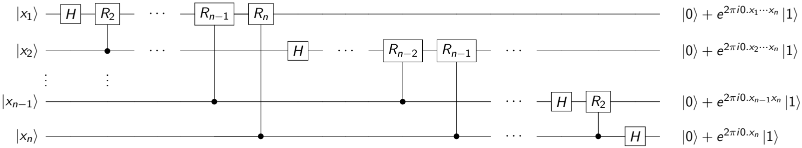
- We will use Eq. (4) as **definition** for QFT

  Is not hard to see that it is essentially the same with Eq. (3) of the DFT (see Appendix)

Quantum Fourier Transform

- We will use Eq. (4) as **definition** for QFT

  Is not hard to see that it is essentially the same with Eq. (3) of the DFT (see Appendix)
- The Quantum Circuit for F is:

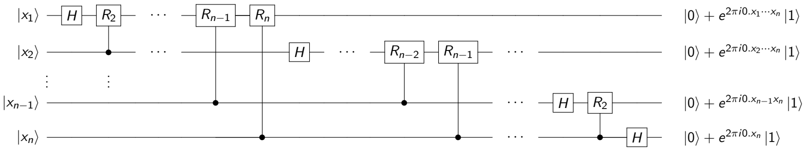


where the gate $R_k := \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$ and we omitted a swap of the qubits and a factor $\frac{1}{\sqrt{2}}$ at the end of the circuit

Quantum Fourier Transform

- We will use Eq. (4) as **definition** for QFT

  Is not hard to see that it is essentially the same with Eq. (3) of the DFT (see Appendix)
- The Quantum Circuit for F is:



  where the gate $R_k := \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$ and we omitted a swap of the qubits and a factor $\frac{1}{\sqrt{2}}$ at the end of the circuit
- It is simple to see that F is unitary since the circuit consists of unitary gates (see also Appendix)

Quantum Fourier Transform

Example: Three qubits

$$F\,|x_1x_2x_3\rangle = |\psi_1\psi_2\psi_3\rangle$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3]}\,|1\rangle\right)$$
$$|\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2x_3]}\,|1\rangle\right)$$
$$|\psi_3\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1x_2x_3]}\,|1\rangle\right)$$

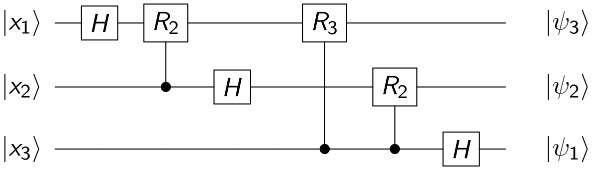
Quantum Fourier Transform

Example: Three qubits

$$F\,|x_1x_2x_3\rangle = |\psi_1\psi_2\psi_3\rangle$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3]}\,|1\rangle\right)$$
$$|\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2x_3]}\,|1\rangle\right)$$
$$|\psi_3\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1x_2x_3]}\,|1\rangle\right)$$

The corresponding circuit is:



Quantum Fourier Transform

- The number of gates in QFT (including the final swaps) is $\Theta(n^2)$

Quantum Fourier Transform

- The number of gates in QFT (including the final swaps) is $\Theta(n^2)$
- To implement the **classical Fast Fourier Transform** $\Theta(n2^n)$ gates are needed

Quantum Fourier Transform

- The number of gates in QFT (including the final swaps) is $\Theta(n^2)$
- To implement the **classical Fast Fourier Transform** $\Theta(n2^n)$ gates are needed
- It appears we obtained an exponential speed-up for a task (DFT) that has many application

Quantum Fourier Transform

- The number of gates in QFT (including the final swaps) is $\Theta(n^2)$
- To implement the **classical Fast Fourier Transform** $\Theta(n2^n)$ gates are needed
- It appears we obtained an exponential speed-up for a task (DFT) that has many application
- However, we **cannot access**(read-out) the amplitudes of a quantum state, so we cannot extract the classical values of the DFT.
- To achieve real speed-up, we need to use QFT as part of larger algorithm (see later!)

Appendix: QFT proof details

- Rewrite Eq. (4):

$$
\begin{aligned}
F\left|x_1 x_2 \cdots x_n\right\rangle &= \left|\psi_1 \psi_2 \cdots \psi_n\right\rangle \\
&= \frac{1}{\sqrt{N}} \left(\sum_{y_1 \in \{0,1\}} e^{2\pi i [0.x_n] y_1} \left|y_1\right\rangle \right) \otimes \left(\sum_{y_2 \in \{0,1\}} e^{2\pi i [0.x_{n-1} x_n] y_2} \left|y_2\right\rangle \right) \otimes \\
&\quad \otimes \cdots \otimes \left(\sum_{y_n \in \{0,1\}} e^{2\pi i [0.x_1 x_2 \cdots x_n] y_n} \left|y_n\right\rangle \right) \\
&= \frac{1}{\sqrt{N}} \left(\sum_{y_1 \in \{0,1\}} e^{2\pi i x (y_1/2^1)} \left|y_1\right\rangle \right) \otimes \left(\sum_{y_2 \in \{0,1\}} e^{2\pi i x (y_2/2^2)} \left|y_2\right\rangle \right) \otimes \\
&\quad \otimes \cdots \otimes \left(\sum_{y_n \in \{0,1\}} e^{2\pi i x (y_n/2^n)} \left|y_n\right\rangle \right) \\
&= \frac{1}{\sqrt{N}} \otimes_{l=1}^{N} \left(\sum_{y_l \in \{0,1\}} e^{2\pi i x (y_l/2^l)} \left|y_l\right\rangle \right) \qquad (5)
\end{aligned}
$$

Appendix: QFT proof details

It follows that

$$\begin{aligned} F\left|x_1x_2\cdots x_n\right\rangle &= \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi i x \sum_{l=1}^{n} y_l/2^l}\left|y\right\rangle \\ &= \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{2\pi i xy/N}\left|y\right\rangle \qquad (6) \end{aligned}$$

where we used that $y = y_1 2^{n-1} + y_2 2^{n-2} + \cdots + y_n 2^0$, similarly for x and $N = 2^n$.

Appendix: QFT proof details

- Express F as an operator: $F = \frac{1}{\sqrt{N}} \sum_{x,y=0}^{N-1} e^{2\pi i xy/N} |y\rangle \langle x|$
- Show that is unitary:

$$
\begin{aligned}
F^\dagger F &= \frac{1}{N} \sum_{x,y,x',y'=0}^{N-1} e^{-2\pi i x'y'/N} |x'\rangle \langle y'| e^{2\pi i xy/N} |y\rangle |x\rangle \\
&= \frac{1}{N} \sum_{x,y,x',y'=0}^{N-1} e^{-2\pi i (x'y'-xy)/N} |x'\rangle \langle x| \langle y'| y\rangle \\
&= \frac{1}{N} \sum_{x,x'=0}^{N-1} \left(\sum_{y=0}^{N-1} e^{-2\pi i y(x'-x)/N} \right) |x'\rangle \langle x| \\
&= \frac{1}{N} \sum_{x,x'=0}^{N-1} N\delta_{x,x'} |x'\rangle \langle x| = \sum_{x=0}^{N-1} |x\rangle \langle x| = \mathbf{1} \ \square
\end{aligned}
$$