Title: Quantum computation: where does the speedup come from?

Date: Feb 08, 2006 02:00 PM

URL: http://pirsa.org/06020011

Abstract: I look at the information-processing involved in a quantum computation, in terms of the difference between the Boolean logic underlying a classical computation and the non-Boolean logic represented by the projective geometry of Hilbert space, in which the subspace structure of Hilbert space replaces the set-theoretic structure of classical logic. I show that the original Deutsch XOR algorithm, Simon's algorithm, and Shor's algorithm all involve a similar geometric formulation. In terms of this picture, I consider the question of where the speedup relative to classical algorithms comes from.

Quantum Computation: Where Does the Speedup Come From?

Jeffrey Bub

Department of Philosophy
University of Maryland

PI, February 8, 2006

Outline

General strategy

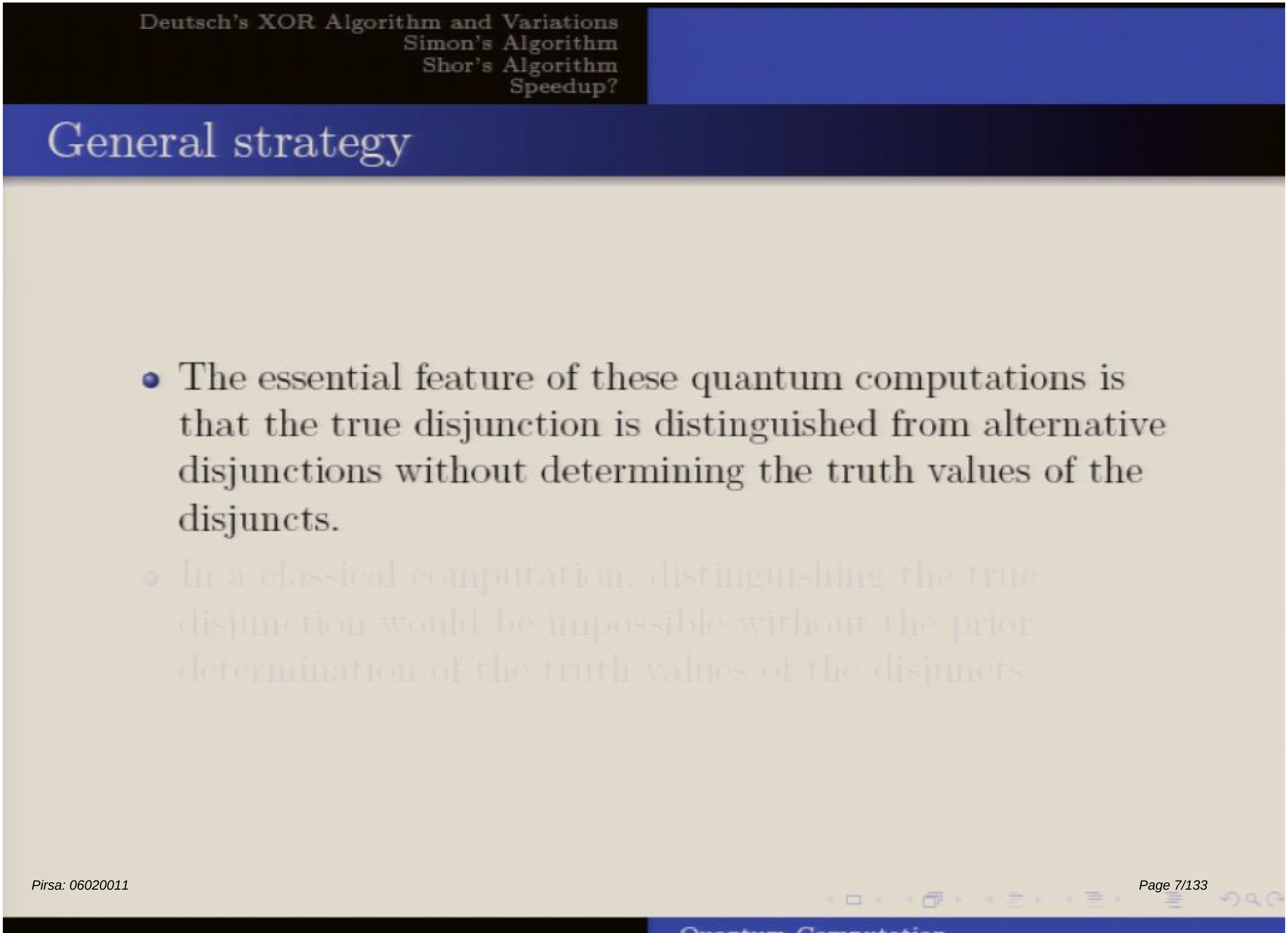
- All three algorithms involve the determination of a global property of a function, i.e., a disjunctive property.
- The disjunction is represented as a subspace in an appropriate Hilbert space, and alternative possible disjunctions turn out to be represented as orthogonal subspaces, except for intersections or overlaps.
- The true disjunction is determined as the subspace containing the state vector via a measurement.
- The algorithm generally has to be run several times because the state might be found in the overlap region.

General strategy

- All three algorithms involve the determination of a global property of a function, i.e., a disjunctive property.
- The disjunction is represented as a subspace in an appropriate Hilbert space, and alternative possible disjunctions turn out to be represented as orthogonal subspaces, except for intersections or overlaps.
- The true disjunction is determined as the subspace containing the state vector via a measurement.
- The algorithm generally has to be run several times because the state might be found in the overlap region.

General strategy

- All three algorithms involve the determination of a global property of a function, i.e., a disjunctive property.
- The disjunction is represented as a subspace in an appropriate Hilbert space, and alternative possible disjunctions turn out to be represented as orthogonal subspaces, except for intersections or overlaps.
- The true disjunction is determined as the subspace containing the state vector via a measurement.
- The algorithm generally has to be run several times because the state might be found in the overlap region.



General strategy

- The essential feature of these quantum computations is that the true disjunction is distinguished from alternative disjunctions without determining the truth values of the disjuncts.
- In a classical computation, distinguishing the true disjunction would be impossible without the prior determination of the truth values of the disjuncts.

General strategy

- The essential feature of these quantum computations is that the true disjunction is distinguished from alternative disjunctions without determining the truth values of the disjuncts.
- In a classical computation, distinguishing the true disjunction would be impossible without the prior determination of the truth values of the disjuncts.

Deutsch's XOR algorithm

- $B = \{0, 1\}$ a Boolean algebra (or the additive group of integers mod 2).
- Given a 'black box' or oracle that computes a function $f : B \to B$.
  Required to determine whether the function is 'constant' (takes the same value for both inputs) or 'balanced' (takes a different value for each input).
- Classically, the only way to do this would be to consult the oracle twice, for the input values 0 and 1, and compare the outputs.

Deutsch's XOR algorithm

- B = $\{0, 1\}$ a Boolean algebra (or the additive group of integers mod 2).
- Given a 'black box' or oracle that computes a function f : B $\rightarrow$ B.
  Required to determine whether the function is 'constant' (takes the same value for both inputs) or ''balanced' (takes a different value for each input).
- Classically, the only way to do this would be to consult the oracle twice, for the input values 0 and 1, and compare the outputs.

Deutsch's XOR algorithm

- $B = \{0, 1\}$ a Boolean algebra (or the additive group of integers mod 2).
- Given a 'black box' or oracle that computes a function $f : B \to B$.
  Required to determine whether the function is 'constant' (takes the same value for both inputs) or ''balanced' (takes a different value for each input).
- Classically, the only way to do this would be to consult the oracle twice, for the input values 0 and 1, and compare the outputs.

Deutsch's XOR algorithm: quantum computation

- Quantum computation: input and output registers are 1-qubit registers initialized to the state $|0\rangle|0\rangle$ in a standard basis.
- Apply Hadamard transformation to the input register — linear superposition of states corresponding to the two possible input values 0 and 1.
- Unitary transformation $U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ corresponding to the 'black box' correlates input values with corresponding output values.

Deutsch's XOR algorithm: quantum computation

- Quantum computation: input and output registers are 1-qubit registers initialized to the state $|0\rangle|0\rangle$ in a standard basis.
- Apply Hadamard transformation to the input register $\rightarrow$ linear superposition of states corresponding to the two possible input values 0 and 1.
- Unitary transformation $U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ corresponding to the 'black box' correlates input values with corresponding output values.

Deutsch's XOR algorithm: quantum computation

- Quantum computation: input and output registers are 1-qubit registers initialized to the state $|0\rangle|0\rangle$ in a standard basis.
- Apply Hadamard transformation to the input register $\rightarrow$ linear superposition of states corresponding to the two possible input values 0 and 1.
- Unitary transformation $U_f : |x\rangle|y\rangle \longrightarrow |x\rangle|y \oplus f(x)\rangle$ corresponding to the 'black box' correlates input values with corresponding output values.

Deutsch's XOR Algorithm: quantum computation

$$|0\rangle|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle \quad (1)$$

$$\xrightarrow{U_f} \frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle) \quad (2)$$

Final state

f constant: the final composite state of both registers is one of the two orthogonal states:

$$|c_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|0\rangle) \quad (3)$$

$$|c_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle|1\rangle + |1\rangle|1\rangle) \quad (4)$$

f balanced: the final composite state is one of the two orthogonal states:

$$|b_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle) \quad (5)$$

$$|b_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle|1\rangle + |1\rangle|0\rangle) \quad (6)$$

Possible final states lie in planes

- States $|c_1\rangle, |c_2\rangle$ and $|b_1\rangle, |b_2\rangle$ span two planes P_c, P_b in $\mathcal{H}^2 \otimes \mathcal{H}^2$, represented by the projection operators:

$$P_c = P_{|c_1\rangle} + P_{|c_2\rangle} \tag{7}$$
$$P_b = P_{|b_1\rangle} + P_{|b_2\rangle} \tag{8}$$

- Planes are orthogonal, except for an intersection, so their projection operators commute. The intersection is the line (ray) spanned by the vector:

$$\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle) = \frac{1}{\sqrt{2}}(|c_1\rangle+|c_2\rangle) = \frac{1}{\sqrt{2}}(|b_1\rangle+|b_2\rangle) \tag{9}$$

Planes in prime basis

- In 'prime' basis spanned by the states $|0'\rangle = \mathrm{H}|0\rangle, |1'\rangle = \mathrm{H}|1\rangle$ intersection is the state $|0'\rangle|0'\rangle$, 'constant' plane is spanned by $|0'\rangle|0'\rangle, |0'\rangle|1'\rangle$, and 'balanced' plane is spanned by $|0'\rangle|0'\rangle, |1'\rangle|1'\rangle$.
- Note that:

$$|0'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|c_1\rangle - |c_2\rangle) \qquad (10)$$

$$|1'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|b_1\rangle - |b_2\rangle) \qquad (11)$$

Possible final states lie in planes

- States $|c_1\rangle, |c_2\rangle$ and $|b_1\rangle, |b_2\rangle$ span two planes P_c, P_b in $\mathcal{H}^2 \otimes \mathcal{H}^2$, represented by the projection operators:

$$P_c = P_{|c_1\rangle} + P_{|c_2\rangle} \quad (7)$$

$$P_b = P_{|b_1\rangle} + P_{|b_2\rangle} \quad (8)$$

- Planes are orthogonal, except for an intersection, so their projection operators commute. The intersection is the line (ray) spanned by the vector:

$$\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle) = \frac{1}{\sqrt{2}}(|c_1\rangle+|c_2\rangle) = \frac{1}{\sqrt{2}}(|b_1\rangle+|b_2\rangle) \quad (9)$$

Planes in prime basis

- In 'prime' basis spanned by the states $|0'\rangle = \mathrm{H}|0\rangle, |1'\rangle = \mathrm{H}|1\rangle$ intersection is the state $|0'\rangle|0'\rangle$, 'constant' plane is spanned by $|0'\rangle|0'\rangle, |0'\rangle|1'\rangle$, and 'balanced' plane is spanned by $|0'\rangle|0'\rangle, |1'\rangle|1'\rangle$.
- Note that:

$$|0'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|c_1\rangle - |c_2\rangle) \tag{10}$$

$$|1'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|b_1\rangle - |b_2\rangle) \tag{11}$$

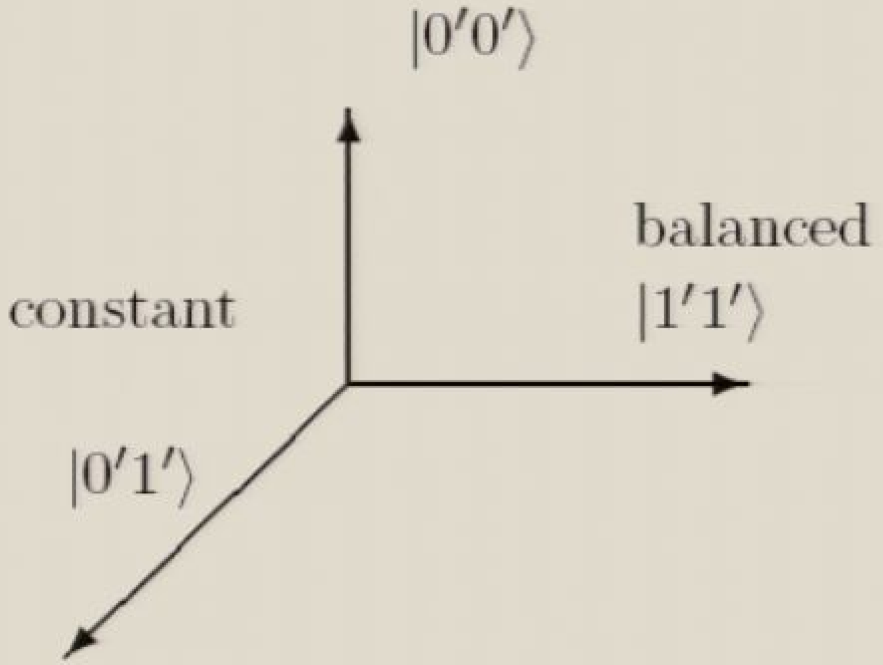
Planes in prime basis

- In 'prime' basis spanned by the states $|0'\rangle = \mathrm{H}|0\rangle, |1'\rangle = \mathrm{H}|1\rangle$ intersection is the state $|0'\rangle|0'\rangle$, 'constant' plane is spanned by $|0'\rangle|0'\rangle, |0'\rangle|1'\rangle$, and 'balanced' plane is spanned by $|0'\rangle|0'\rangle, |1'\rangle|1'\rangle$.
- Note that:

$$|0'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|c_1\rangle - |c_2\rangle) \quad (10)$$

$$|1'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|b_1\rangle - |b_2\rangle) \quad (11)$$

Planes in prime basis



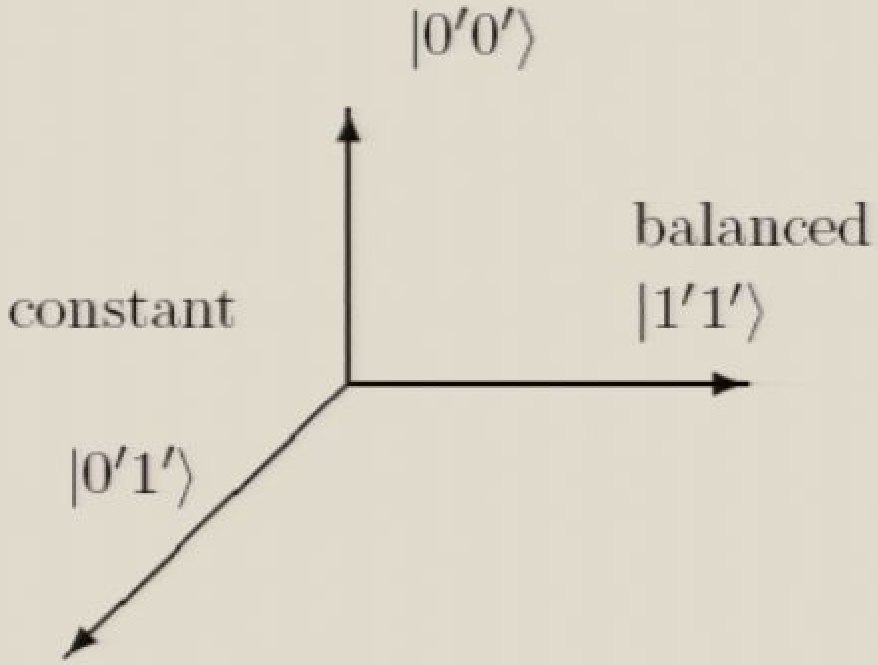
Planes in prime basis

- In 'prime' basis spanned by the states $|0'\rangle = \mathrm{H}|0\rangle, |1'\rangle = \mathrm{H}|1\rangle$ intersection is the state $|0'\rangle|0'\rangle$, 'constant' plane is spanned by $|0'\rangle|0'\rangle, |0'\rangle|1'\rangle$, and 'balanced' plane is spanned by $|0'\rangle|0'\rangle, |1'\rangle|1'\rangle$.
- Note that:

$$|0'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|\mathrm{c}_1\rangle - |\mathrm{c}_2\rangle) \quad (10)$$

$$|1'\rangle|1'\rangle = \frac{1}{\sqrt{2}}(|\mathrm{b}_1\rangle - |\mathrm{b}_2\rangle) \quad (11)$$

Planes in prime basis



Usual formulation of algorithm

- Usual formulation of the algorithm: to decide whether the function f is constant or balanced measure the output register in prime basis.
- If outcome is $0'$ (obtained with probability $1/2$, whether the state ends up in the constant plane or the balanced plane), the computation is inconclusive, yielding no information about the function f.
- If outcome is $1'$, measure the input register. If the outcome of the measurement on the input register is $0'$, the function is constant; if it is $1'$, the function is balanced.

Geometric formulation

- Alternatively, measure the observable with eigenstates $|0'0'\rangle$, $|0'1'\rangle$, $|1'0'\rangle$, $|1'1'\rangle$. Final state is in 3-dimensional subspace orthogonal to the vector $|1'0'\rangle$, either in the constant plane or the balanced plane.
- If state is in constant plane, we will either obtain the outcome $0'0'$ with probability $1/2$ (since the final state is at an angle $\pi/4$ to $|0'0'\rangle$), in which case the computation is inconclusive, or the outcome $0'1'$ with probability $1/2$.
- If state is in balanced plane, we will again obtain the outcome $0'0'$ with probability $1/2$, in which case the computation is inconclusive, or the outcome $1'1'$ with probability $1/2$.

Solution in one run

- With probability 1/2, we can distinguish in one run of the algorithm between the two quantum disjunctions 'constant' and 'balanced' represented by the planes:

$$P_c = P_{|0'0'\rangle} \vee P_{|0'1'\rangle} \quad (12)$$

$$P_b = P_{|0'0'\rangle} \vee P_{|1'1'\rangle} \quad (13)$$

  without finding out the truth values of the disjuncts in the computation (i.e., whether in the 'constant' case the function maps 0 to 0 and 1 to 0 or whether the function maps 0 to 1 and 1 to 1, and similarly in the 'balanced' case).
- We could also apply a Hadamard transformation to the final states of both registers and measure in the computational basis, since $|0'0'\rangle \xrightarrow{H} |00\rangle$, etc.

Modest speedup, probability 1/2 of failing

- Deutsch's XOR algorithm was the first quantum algorithm with a demonstrated speed-up over any classical algorithm performing the same computational task.
- Algorithm has an even probability of failing, so improvement in efficiency over a classical computation is only achieved if the algorithm succeeds, and even then is rather modest: one run of the quantum algorithm versus two runs of a classical algorithm.
- A variation by Cleve (1998) avoids this feature.

Usual formulation

- Consider state of input register:

$$\sum_{y}\sum_{x}\frac{(-1)^{x\cdot y+f(x)}}{\sqrt{2^n}}|y\rangle = \sum_{x}\frac{(-1)^{f(x)}}{\sqrt{2^n}}|0\ldots 0\rangle + \ldots \tag{17}$$

- Coefficient of state $|0\ldots 0\rangle$ in the linear superposition is $\sum_x \frac{(-1)^{f(x)}}{\sqrt{2^n}}$.

f constant: coefficient is ± 1, so the coefficients of the other terms must all be 0.

f is balanced: $f(x) = 0$ for half the values of x and $f(x) = 1$ for the other half, so the positive and negative contributions to the coefficient of $|0\ldots 0\rangle$ cancel to 0.

Geometric Picture

- The final Hadamard transformation transforms the constant state:

$$\pm\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \xrightarrow{\mathrm{H}} \pm|00\rangle \qquad (18)$$

  and the six balanced states to states in the 3-dimensional subspace orthogonal to $|00\rangle$.
- So to decide whether the function is constant or balanced we need only measure the input register and check whether it is in the state $|00\rangle$.

Simon's algorithm

- Problem: find the period r of a periodic function $f : B^n \rightarrow B^n$, i.e., a Boolean function for which

$$f(x_i) = f(x_j) \text{ if and only if } x_j = x_i \oplus r \text{ for all } x_i, x_j \in B^n \tag{19}$$

- Since $x \oplus r \oplus r = x$, the function is 2-to-1.

Simon's algorithm

- Problem: find the period r of a periodic function $\mathrm{f} : \mathrm{B}^{\mathrm{n}} \rightarrow \mathrm{B}^{\mathrm{n}}$, i.e., a Boolean function for which

$$\mathrm{f}(\mathrm{x_i}) = \mathrm{f}(\mathrm{x_j}) \text{ if and only if } \mathrm{x_j} = \mathrm{x_i} \oplus \mathrm{r} \text{ for all } \mathrm{x_i}, \mathrm{x_j} \in \mathrm{B}^{\mathrm{n}} \tag{19}$$

- Since $\mathrm{x} \oplus \mathrm{r} \oplus \mathrm{r} = \mathrm{x}$, the function is 2-to-1.

Simon's Algorithm

- Start with the input and output registers in the state $|0\dots0\rangle|0\rangle$ in the computational basis:

$$|0\dots0\rangle|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle|0\rangle \quad (20)$$

$$\xrightarrow{U_f} \frac{1}{\sqrt{2^n}}\sum_{x}|x\rangle|f(x)\rangle \quad (21)$$

$$= \frac{1}{\sqrt{2^{n-1}}}\sum_{x_i}\frac{|x_i\rangle+|x_i\oplus r\rangle}{\sqrt{2}}|f(x_i)\rangle \quad (22)$$

where U_f is the unitary transformation implementing the Boolean function as:

$$U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y\oplus f(x)\rangle \quad (23)$$

Usual formulation

- Consider what happens if we measure the output register and keep the state of the input register, which will have the form:
$$\frac{|x_i\rangle + |x_i \oplus r\rangle}{\sqrt{2}} \tag{24}$$

- This state contains the information r, but summed with an unwanted randomly chosen offset x_i that depends on the measurement outcome. A direct measurement of the state label would yield any $x \in B^n$ equiprobably, providing no information about r.

Usual formulation

- Consider what happens if we measure the output register and keep the state of the input register, which will have the form:

$$\frac{|x_i\rangle + |x_i \oplus r\rangle}{\sqrt{2}} \tag{24}$$

- This state contains the information r, but summed with an unwanted randomly chosen offset x_i that depends on the measurement outcome. A direct measurement of the state label would yield any $x \in B^n$ equiprobably, providing no information about r.

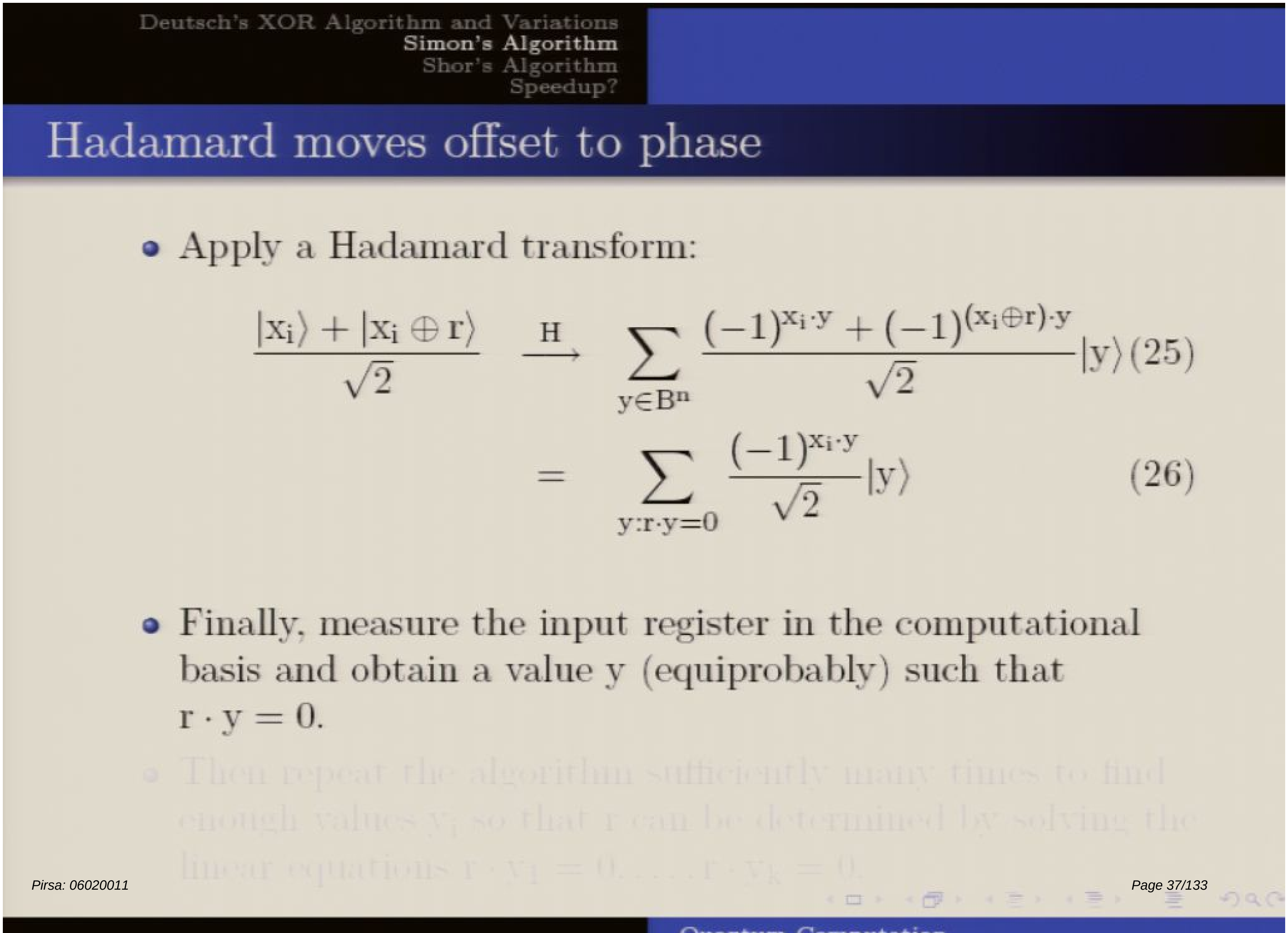
Hadamard moves offset to phase

- Apply a Hadamard transform:

$$\frac{|x_i\rangle + |x_i \oplus r\rangle}{\sqrt{2}} \xrightarrow{H} \sum_{y \in B^n} \frac{(-1)^{x_i \cdot y} + (-1)^{(x_i \oplus r)\cdot y}}{\sqrt{2}} |y\rangle \quad (25)$$

$$= \sum_{y : r \cdot y = 0} \frac{(-1)^{x_i \cdot y}}{\sqrt{2}} |y\rangle \quad (26)$$

- Finally: measure the input register in the computational basis and obtain a value y (equiprobably) such that $r \cdot y = 0$.
- Then repeat the algorithm sufficiently many times to find enough values y_i so that r can be determined by solving the linear equations $r \cdot y_1 = 0, \ldots, r \cdot y_k = 0$.



Hadamard moves offset to phase

- Apply a Hadamard transform:

$$\frac{|x_i\rangle + |x_i \oplus r\rangle}{\sqrt{2}} \xrightarrow{H} \sum_{y \in B^n} \frac{(-1)^{x_i \cdot y} + (-1)^{(x_i \oplus r)\cdot y}}{\sqrt{2}} |y\rangle \quad (25)$$

$$= \sum_{y : r \cdot y = 0} \frac{(-1)^{x_i \cdot y}}{\sqrt{2}} |y\rangle \quad (26)$$

- Finally, measure the input register in the computational basis and obtain a value y (equiprobably) such that $r \cdot y = 0$.
- Then repeat the algorithm sufficiently many times to find enough values y_i so that r can be determined by solving the linear equations $r \cdot y_1 = 0, \ldots, r \cdot y_k = 0$.

Hadamard moves offset to phase

- Apply a Hadamard transform:

$$\frac{|x_i\rangle + |x_i \oplus r\rangle}{\sqrt{2}} \xrightarrow{H} \sum_{y \in B^n} \frac{(-1)^{x_i \cdot y} + (-1)^{(x_i \oplus r)\cdot y}}{\sqrt{2}} |y\rangle \quad (25)$$

$$= \sum_{y : r \cdot y = 0} \frac{(-1)^{x_i \cdot y}}{\sqrt{2}} |y\rangle \quad (26)$$

- Finally, measure the input register in the computational basis and obtain a value y (equiprobably) such that $r \cdot y = 0$.
- Then repeat the algorithm sufficiently many times to find enough values y_i so that r can be determined by solving the linear equations $r \cdot y_1 = 0, \ldots, r \cdot y_k = 0$.

Simon's Algorithm—Geometric Picture

- To see what is going on geometrically, consider case $n = 2$.
- Possible values of the period r are: 01, 10, 11, and corresponding states of the input and output registers after the unitary transformation U_f are:

$$r = 01: \ (|00\rangle + |01\rangle)|f(00)\rangle + (|10\rangle + |11\rangle)|f(10)\rangle$$
$$r = 10: \ (|00\rangle + |10\rangle)|f(00)\rangle + (|01\rangle + |11\rangle)|f(01)\rangle$$
$$r = 11: \ (|00\rangle + |11\rangle)|f(00)\rangle + (|01\rangle + |10\rangle)|f(01)\rangle$$

Simon's Algorithm—Geometric Picture

- To see what is going on geometrically, consider case $n = 2$.
- Possible values of the period r are: 01, 10, 11, and corresponding states of the input and output registers after the unitary transformation U_f are:

$$r = 01: \ (|00\rangle + |01\rangle)|f(00)\rangle + (|10\rangle + |11\rangle)|f(10)\rangle$$
$$r = 10: \ (|00\rangle + |10\rangle)|f(00)\rangle + (|01\rangle + |11\rangle)|f(01)\rangle$$
$$r = 11: \ (|00\rangle + |11\rangle)|f(00)\rangle + (|01\rangle + |10\rangle)|f(01)\rangle$$

$n = 2$ case reduces to Deutsch's XOR algorithm

- This case reduces to the same geometric construction as in Deutsch's XOR algorithm.

  $r = 10$: input register states are $|c_1\rangle = |00\rangle + |10\rangle$ or $|c_2\rangle = |01\rangle + |11\rangle$, depending on the outcome of the measurement of the output register.

  $r = 11$: input register states are $|b_1\rangle = |00\rangle + |11\rangle$ or $|b_2\rangle = |01\rangle + |10\rangle$, depending on the outcome of the measurement of the output register.

- So the three possible periods are associated with three planes in $\mathcal{H}^2 \otimes \mathcal{H}^2$, which correspond to the constant and balanced planes in Deutsch's XOR algorithm, and a third plane, all three planes intersecting in the line spanned by the vector $|00\rangle$.

n = 2 case reduces to Deutsch's XOR algorithm

- This case reduces to the same geometric construction as in Deutsch's XOR algorithm.

  $r = 10$: input register states are $|c_1\rangle = |00\rangle + |10\rangle$ or $|c_2\rangle = |01\rangle + |11\rangle$, depending on the outcome of the measurement of the output register.

  $r = 11$: input register states are $|b_1\rangle = |00\rangle + |11\rangle$ or $|b_2\rangle = |01\rangle + |10\rangle$, depending on the outcome of the measurement of the output register.

- So the three possible periods are associated with three planes in $\mathcal{H}^2 \otimes \mathcal{H}^2$, which correspond to the constant and balanced planes in Deutsch's XOR algorithm, and a third plane, all three planes intersecting in the line spanned by the vector $|00\rangle$.

n = 2 case reduces to Deutsch's XOR algorithm

- This case reduces to the same geometric construction as in Deutsch's XOR algorithm.

  $r = 10$: input register states are $|c_1\rangle = |00\rangle + |10\rangle$ or $|c_2\rangle = |01\rangle + |11\rangle$, depending on the outcome of the measurement of the output register.

  $r = 11$: input register states are $|b_1\rangle = |00\rangle + |11\rangle$ or $|b_2\rangle = |01\rangle + |10\rangle$, depending on the outcome of the measurement of the output register.

- So the three possible periods are associated with three planes in $\mathcal{H}^2 \otimes \mathcal{H}^2$, which correspond to the constant and balanced planes in Deutsch's XOR algorithm, and a third plane, all three planes intersecting in the line spanned by the vector $|00\rangle$.

Prime basis

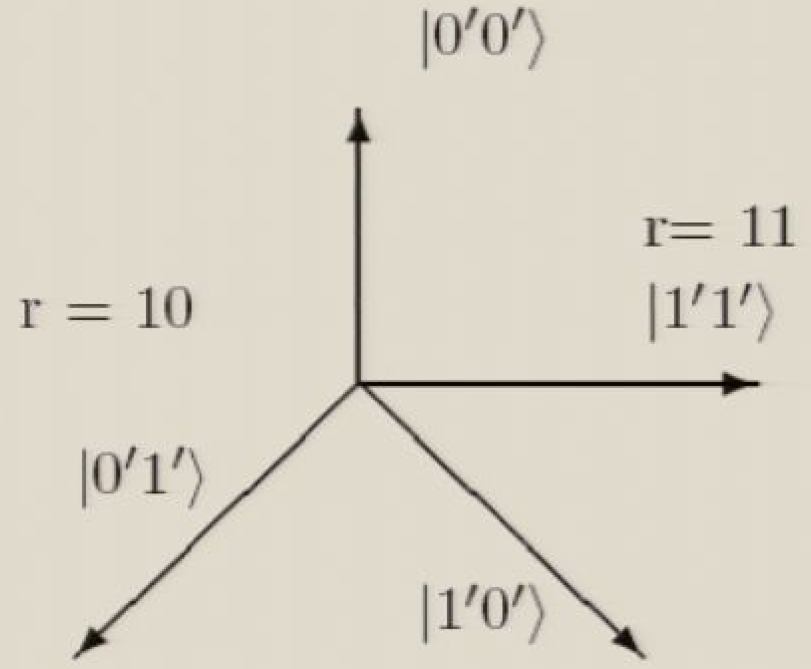
In the prime basis obtained by applying the Hadamard transformation, planes are as follows:

r = 01 : plane spanned by $|0'0'\rangle, |1'0'\rangle$

r = 10 : plane spanned by $|0'0'\rangle, |0'1'\rangle$ (corresponds to 'constant' plane)

r = 11 : plane spanned by $|0'0'\rangle, |1'1'\rangle$ (corresponds to 'balanced' plane)

Planes in prime basis



Finding the period

- We could simply measure the input register in the prime basis to find the period.
- Alternatively, we could apply a Hadamard transformation (which amounts to dropping the primes in the above representation of the r-planes) and measure in the computational basis.

Finding the period

- We could simply measure the input register in the prime basis to find the period.
- Alternatively, we could apply a Hadamard transformation (which amounts to dropping the primes in the above representation of the r-planes) and measure in the computational basis.

Finding the period

- The three planes are orthogonal, except for their intersection in the line spanned by the vector $|00\rangle$. The three possible periods can therefore be distinguished by measuring the observable with eigenstates $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, except when the state of the register is projected by the measurement ('collapses') onto the intersection state $|00\rangle$ (which occurs with probability $1/2$).
- So the algorithm will generally have to be repeated until we find an outcome that is not 00.

Finding the period

- The three planes are orthogonal, except for their intersection in the line spanned by the vector $|00\rangle$. The three possible periods can therefore be distinguished by measuring the observable with eigenstates $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, except when the state of the register is projected by the measurement ('collapses') onto the intersection state $|00\rangle$ (which occurs with probability $1/2$).
- So the algorithm will generally have to be repeated until we find an outcome that is not 00.

General case: n = 3

- We can see what happens in the general case if we consider the case n = 3.
- There are now seven possible periods: 001, 010, 011, 100, 101, 110, 111.

General case: n = 3

- We can see what happens in the general case if we consider the case $n = 3$.
- There are now seven possible periods: 001, 010, 011, 100, 101, 110, 111.

Period r = 001

- Period r = 001: the state of the two registers after the unitary transformation U_f is:

$$\begin{aligned}&(|000\rangle + |001\rangle)|f(000)\rangle + (|010\rangle + |011\rangle)|f(010)\rangle \\ &\quad +(|100\rangle + |101\rangle)|f(100)\rangle + (|110\rangle + |111\rangle)|f(110)\rangle\end{aligned} \tag{27}$$

- Measure the output register — the input register is left in one of four states, depending on the outcome of the measurement:

$$\begin{aligned}|000\rangle + |001\rangle &= |0'0'0'\rangle + |0'1'0'\rangle + |1'0'0'\rangle + |1'1'0'\rangle \\ |010\rangle + |011\rangle &= |0'0'0'\rangle - |0'1'0'\rangle + |1'0'0'\rangle - |1'1'0'\rangle \\ |100\rangle + |101\rangle &= |0'0'0'\rangle + |0'1'0'\rangle - |1'0'0'\rangle - |1'1'0'\rangle \\ |110\rangle + |111\rangle &= |0'0'0'\rangle - |0'1'0'\rangle - |1'0'0'\rangle + |1'1'0'\rangle\end{aligned}$$

State ends up in 4-dimensional subspace

- Applying a Hadamard transformation amounts to dropping the primes.
- So if the period is $r = 001$, the state of the input register ends up in the 4-dimensional subspace of $\mathcal{H}^2 \otimes \mathcal{H}^2 \otimes \mathcal{H}^2$ spanned by the vectors: $|000\rangle, |010\rangle, |100\rangle, |110\rangle$.

Period r = 001

- Period r = 001: the state of the two registers after the unitary transformation U_f is:

$$\begin{aligned}(|000\rangle + |001\rangle)|f(000)\rangle + (|010\rangle + |011\rangle)|f(010)\rangle \\ +(|100\rangle + |101\rangle)|f(100)\rangle + (|110\rangle + |111\rangle)|f(110)\rangle\end{aligned} \quad (27)$$

- Measure the output register $\rightarrow$ the input register is left in one of four states, depending on the outcome of the measurement:

$$\begin{aligned}|000\rangle + |001\rangle &= |0'0'0'\rangle + |0'1'0'\rangle + |1'0'0'\rangle + |1'1'0'\rangle \\ |010\rangle + |011\rangle &= |0'0'0'\rangle - |0'1'0'\rangle + |1'0'0'\rangle - |1'1'0'\rangle \\ |100\rangle + |101\rangle &= |0'0'0'\rangle + |0'1'0'\rangle - |1'0'0'\rangle - |1'1'0'\rangle \\ |110\rangle + |111\rangle &= |0'0'0'\rangle - |0'1'0'\rangle - |1'0'0'\rangle + |1'1'0'\rangle\end{aligned}$$

State ends up in 4-dimensional subspace

- Applying a Hadamard transformation amounts to dropping the primes.
- So if the period is $r = 001$, the state of the input register ends up in the 4-dimensional subspace of $\mathcal{H}^2 \otimes \mathcal{H}^2 \otimes \mathcal{H}^2$ spanned by the vectors: $|000\rangle, |010\rangle, |100\rangle, |110\rangle$.

Period r = 001

- Period r = 001: the state of the two registers after the unitary transformation U_f is:

$$\begin{aligned}(|000\rangle + |001\rangle)|f(000)\rangle + (|010\rangle + |011\rangle)|f(010)\rangle \\ +(|100\rangle + |101\rangle)|f(100)\rangle + (|110\rangle + |111\rangle)|f(110)\rangle\end{aligned} \tag{27}$$

- Measure the output register → the input register is left in one of four states, depending on the outcome of the measurement:

$$\begin{aligned}|000\rangle + |001\rangle &= |0'0'0'\rangle + |0'1'0'\rangle + |1'0'0'\rangle + |1'1'0'\rangle \\ |010\rangle + |011\rangle &= |0'0'0'\rangle - |0'1'0'\rangle + |1'0'0'\rangle - |1'1'0'\rangle \\ |100\rangle + |101\rangle &= |0'0'0'\rangle + |0'1'0'\rangle - |1'0'0'\rangle - |1'1'0'\rangle \\ |110\rangle + |111\rangle &= |0'0'0'\rangle - |0'1'0'\rangle - |1'0'0'\rangle + |1'1'0'\rangle\end{aligned}$$

State ends up in 4-dimensional subspace

- Applying a Hadamard transformation amounts to dropping the primes.
- So if the period is r = 001, the state of the input register ends up in the 4-dimensional subspace of $\mathcal{H}^2 \otimes \mathcal{H}^2 \otimes \mathcal{H}^2$ spanned by the vectors: $|000\rangle, |010\rangle, |100\rangle, |110\rangle$.

State ends up in 4-dimensional subspace

- Applying a Hadamard transformation amounts to dropping the primes.
- So if the period is r = 001, the state of the input register ends up in the 4-dimensional subspace of $\mathcal{H}^2 \otimes \mathcal{H}^2 \otimes \mathcal{H}^2$ spanned by the vectors: $|000\rangle, |010\rangle, |100\rangle, |110\rangle$.

Subspaces for different possible periods

A similar analysis applies to the other six possible periods. Corresponding subspaces are spanned by the following vectors:

$r = 001$: $|000\rangle, |010\rangle, |100\rangle, |110\rangle$

$r = 010$: $|000\rangle, |001\rangle, |100\rangle, |101\rangle$

$r = 011$: $|000\rangle, |011\rangle, |100\rangle, |111\rangle$

$r = 100$: $|000\rangle, |001\rangle, |010\rangle, |011\rangle$

$r = 101$: $|000\rangle, |010\rangle, |101\rangle, |111\rangle$

$r = 110$: $|000\rangle, |001\rangle, |110\rangle, |111\rangle$

$r = 111$: $|000\rangle, |011\rangle, |101\rangle, |110\rangle$

Geometric Picture

- Subspaces are orthogonal except for intersections in 2-dimensional planes. Note that the subspaces correspond to quantum disjunctions. So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.
- The period can be found by measuring in the computational basis. Repetitions of the measurement will eventually yield sufficiently many distinct values to determine in which subspace out of the seven possibilities the final state lies.

Geometric Picture

- Subspaces are orthogonal except for intersections in 2-dimensional planes. Note that the subspaces correspond to quantum disjunctions. So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.
- The period can be found by measuring in the computational basis. Repetitions of the measurement will eventually yield sufficiently many distinct values to determine in which subspace out of the seven possibilities the final state lies.

Geometric Picture

- In case $n = 3$, it is clear by examining the above list that two values distinct from 000 suffice to determine the subspace, and these are just the values y_i for which $r \cdot y_i = 0$.
- Note that the subspaces correspond to quantum disjunctions.
- So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.

Geometric Picture

- In case $n = 3$, it is clear by examining the above list that two values distinct from 000 suffice to determine the subspace, and these are just the values y_i for which $r \cdot y_i = 0$.
- Note that the subspaces correspond to quantum disjunctions.
- So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.

Geometric Picture

- In case $n = 3$, it is clear by examining the above list that two values distinct from 000 suffice to determine the subspace, and these are just the values y_i for which $r \cdot y_i = 0$.
- Note that the subspaces correspond to quantum disjunctions.
- So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.

Prime factorization

- Shor's factorization algorithm exploits the fact that the two prime factors p, q of a positive integer $N = pq$ can be found by determining the period of a function $f(x) = a^x \bmod N$, for any $a < N$ which is coprime to N, i.e., has no common factors with N other than 1.
- The period r of $f(x)$ depends on a and N. Once we know the period, we can factor N if r is even and $a^{r/2} \neq -1 \bmod N$, which will be the case with probability greater than $1/2$ if a is chosen randomly. (If not, we choose another value of a.)

Geometric Picture

- In case $n = 3$, it is clear by examining the above list that two values distinct from 000 suffice to determine the subspace, and these are just the values y_i for which $r \cdot y_i = 0$.
- Note that the subspaces correspond to quantum disjunctions.
- So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.

Prime factorization

- Shor's factorization algorithm exploits the fact that the two prime factors p, q of a positive integer $N = pq$ can be found by determining the period of a function $f(x) = a^x \bmod N$, for any $a < N$ which is coprime to N, i.e., has no common factors with N other than 1.
- The period r of f(x) depends on a and N. Once we know the period, we can factor N if r is even and $a^{r/2} \neq -1 \bmod N$, which will be the case with probability greater than 1/2 if a is chosen randomly. (If not, we choose another value of a.)

Prime factorization

- Shor's factorization algorithm exploits the fact that the two prime factors p, q of a positive integer $N = pq$ can be found by determining the period of a function $f(x) = a^x \bmod N$, for any $a < N$ which is coprime to N, i.e., has no common factors with N other than 1.
- The period r of $f(x)$ depends on a and N. Once we know the period, we can factor N if r is even and $a^{r/2} \neq -1 \bmod N$, which will be the case with probability greater than $1/2$ if a is chosen randomly. (If not, we choose another value of a.)

Geometric Picture

- In case $n = 3$, it is clear by examining the above list that two values distinct from 000 suffice to determine the subspace, and these are just the values y_i for which $r \cdot y_i = 0$.
- Note that the subspaces correspond to quantum disjunctions.
- So determining the period of the function by Simon's algorithm amounts to determining which disjunction out of the seven alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.

Prime factorization

- Shor's factorization algorithm exploits the fact that the two prime factors p, q of a positive integer $N = pq$ can be found by determining the period of a function $f(x) = a^x \bmod N$, for any $a < N$ which is coprime to N, i.e., has no common factors with N other than 1.
- The period r of $f(x)$ depends on a and N. Once we know the period, we can factor N if r is even and $a^{r/2} \neq -1 \bmod N$, which will be the case with probability greater than $1/2$ if a is chosen randomly. (If not, we choose another value of a.)

Prime factorization

- The factors of N are the greatest common factors of $a^{r/2} \pm 1$ and N, which can be found in polynomial time by the Euclidean algorithm.
- So the problem of factorizing a composite integer N that is the product of two primes reduces to the problem of finding the period of a certain periodic function $f : Z_s \to Z_N$, where Z_n is the additive group of integers mod n (rather than B^n, the n-fold Cartesian product of a Boolean algebra B, as in Simon's algorithm).
- Note that $f(x + r) = f(x)$ if $x + r \leq s$. The function f is periodic if r divides s exactly; otherwise it is almost periodic.

Prime factorization

- The factors of N are the greatest common factors of $a^{r/2} \pm 1$ and N, which can be found in polynomial time by the Euclidean algorithm.
- So the problem of factorizing a composite integer N that is the product of two primes reduces to the problem of finding the period of a certain periodic function $f : Z_s \rightarrow Z_N$, where Z_n is the additive group of integers mod n (rather than B^n, the n-fold Cartesian product of a Boolean algebra B, as in Simon's algorithm).
- Note that $f(x + r) = f(x)$ if $x + r \leq s$. The function f is periodic if r divides s exactly; otherwise it is almost periodic.

Prime factorization

- The factors of N are the greatest common factors of $a^{r/2} \pm 1$ and N, which can be found in polynomial time by the Euclidean algorithm.
- So the problem of factorizing a composite integer N that is the product of two primes reduces to the problem of finding the period of a certain periodic function $f : Z_s \rightarrow Z_N$, where Z_n is the additive group of integers mod n (rather than B^n, the n-fold Cartesian product of a Boolean algebra B, as in Simon's algorithm).
- Note that $f(x+r) = f(x)$ if $x + r \leq s$. The function f is periodic if r divides s exactly, otherwise it is almost periodic.

Shor's algorithm: usual formulation

- Begin by initializing the input register (s qubits) to the state $|0\rangle \in \mathcal{H}^{\mathrm{s}}$ and the output register (N qubits) to the state $|0\rangle \in \mathcal{H}^{\mathrm{N}}$.
- Apply an s-fold Hadamard transformation to the input register, followed by the unitary transformation $\mathrm{U_f}$ which implements the function $\mathrm{f(x) = a^x \bmod N}$:

$$|0\rangle|0\rangle \quad \xrightarrow{\mathrm{H}} \quad \frac{1}{\sqrt{\mathrm{s}}}\sum_{\mathrm{x}=0}^{\mathrm{s}-1}|\mathrm{x}\rangle|0\rangle \tag{28}$$

$$\xrightarrow{\mathrm{U_f}} \quad \frac{1}{\sqrt{\mathrm{s}}}\sum_{\mathrm{x}=0}^{\mathrm{s}-1}|\mathrm{x}\rangle|0\rangle \tag{29}$$

$$= \frac{1}{\sqrt{\mathrm{s}}}\sum_{\mathrm{x}=0}^{\mathrm{s}-1}|\mathrm{x}\rangle|\mathrm{x} + \mathrm{a^x} \bmod \mathrm{N}\rangle \tag{30}$$

Shor's algorithm: usual formulation

- Begin by initializing the input register (s qubits) to the state $|0\rangle \in \mathcal{H}^{\mathrm{s}}$ and the output register (N qubits) to the state $|0\rangle \in \mathcal{H}^{\mathrm{N}}$.
- Apply an s-fold Hadamard transformation to the input register, followed by the unitary transformation $\mathrm{U_f}$ which implements the function $\mathrm{f(x) = a^x \bmod N}$:

$$|0\rangle|0\rangle \xrightarrow{\mathrm{H}} \frac{1}{\sqrt{\mathrm{s}}}\sum_{\mathrm{x=0}}^{\mathrm{s-1}} |\mathrm{x}\rangle|0\rangle \qquad (28)$$

$$\xrightarrow{\mathrm{U_f}} \frac{1}{\sqrt{\mathrm{s}}}\sum_{\mathrm{x=0}}^{\mathrm{s-1}} |\mathrm{x}\rangle|0\rangle \qquad (29)$$

$$= \frac{1}{\sqrt{\mathrm{s}}}\sum_{\mathrm{x=0}}^{\mathrm{s-1}} |\mathrm{x}\rangle|\mathrm{x} + \mathrm{a^x} \bmod \mathrm{N}\rangle \qquad (30)$$

Shor's algorithm: usual formulation

- Measure the output register in the computational basis and obtain a state of the following form for the input register:

$$\frac{1}{\sqrt{s/r}} \sum_{j=0}^{s/r-1} |x_i + jr\rangle \tag{31}$$

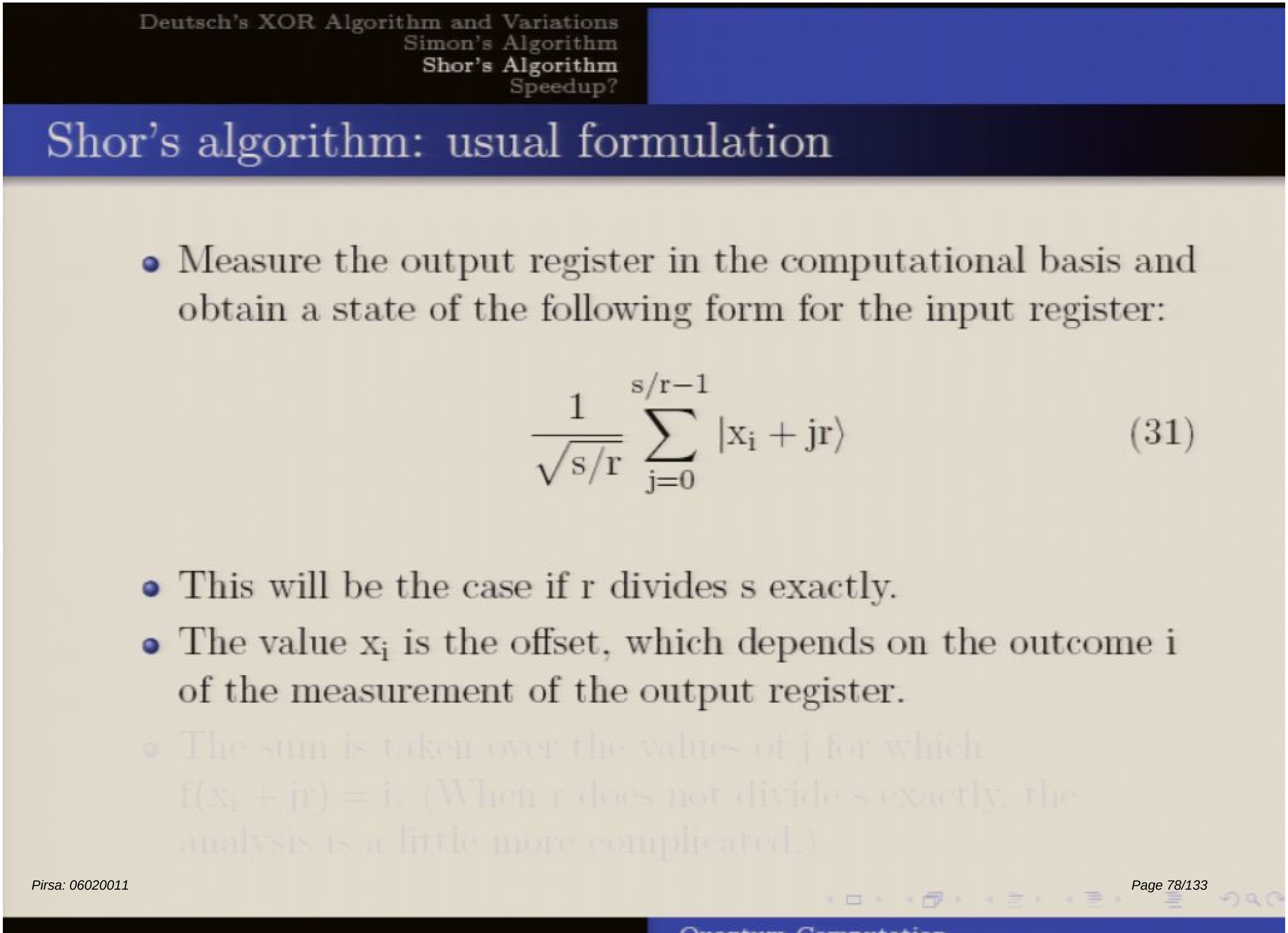
- This will be the case if r divides s exactly.
- The value x_i is the offset, which depends on the outcome i of the measurement of the output register.
- The sum is taken over the values of j for which $f(x_i + jr) = i$. (When r does not divide s exactly, the analysis is a little more complicated.)

Shor's algorithm: usual formulation

- Measure the output register in the computational basis and obtain a state of the following form for the input register:

$$\frac{1}{\sqrt{s/r}}\sum_{j=0}^{s/r-1}|x_i + jr\rangle \tag{31}$$

- This will be the case if r divides s exactly.
- The value x_i is the offset, which depends on the outcome i of the measurement of the output register.
- The sum is taken over the values of j for which $f(x_i + jr) = i$. (When r does not divide s exactly, the analysis is a little more complicated.)



Shor's algorithm: usual formulation

- Measure the output register in the computational basis and obtain a state of the following form for the input register:

$$\frac{1}{\sqrt{s/r}} \sum_{j=0}^{s/r-1} |x_i + jr\rangle \qquad (31)$$

- This will be the case if r divides s exactly.
- The value x_i is the offset, which depends on the outcome i of the measurement of the output register.
- The sum is taken over the values of j for which $f(x_i + jr) = i$. (When r does not divide s exactly, the analysis is a little more complicated.)

Shor's algorithm: usual formulation

- Measure the output register in the computational basis and obtain a state of the following form for the input register:

$$\frac{1}{\sqrt{s/r}} \sum_{j=0}^{s/r-1} |x_i + jr\rangle \tag{31}$$

- This will be the case if r divides s exactly.
- The value x_i is the offset, which depends on the outcome i of the measurement of the output register.
- The sum is taken over the values of j for which $f(x_i + jr) = i$. (When r does not divide s exactly, the analysis is a little more complicated.)

Discrete Fourier transform

- Since the state label contains the random offset, a direct measurement of the label yields no information about the period.
- Apply a discrete Fourier transform for the integers mod s to the input register, i.e., a unitary transformation:

$$|x\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{s}} \sum_{y=0}^{s-1} e^{2\pi i \frac{xy}{s}} |y\rangle, \text{ for } x \in Z_s \quad (32)$$

Discrete Fourier transform

- Since the state label contains the random offset, a direct measurement of the label yields no information about the period.
- Apply a discrete Fourier transform for the integers mod s to the input register, i.e., a unitary transformation:

$$|x\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{s}} \sum_{y=0}^{s-1} e^{2\pi i \frac{xy}{s}} |y\rangle, \text{ for } x \in Z_s \tag{32}$$

Offset shifted into phase factor

- This yields the transition:

$$\frac{1}{\sqrt{\frac{s}{r}}}\sum_{j=0}^{\frac{s}{r}-1}|x_i + jr\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{2\pi i \frac{x_i k}{r}}|ks/r\rangle \qquad (33)$$

- Shifts the offset into a phase factor and inverts the period as a multiple of s/r.
- A measurement of the input register in the computational basis yields $c = ks/r$.
- The algorithm is run a number of times until a value of k coprime to r is obtained. Cancelling c/s to lowest terms then yields k and r as k/r.

Offset shifted into phase factor

- This yields the transition:

$$\frac{1}{\sqrt{\frac{s}{r}}}\sum_{j=0}^{\frac{s}{r}-1}|x_i+jr\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{2\pi i\frac{x_i k}{r}}|ks/r\rangle \qquad (33)$$

- Shifts the offset into a phase factor and inverts the period as a multiple of s/r.
- A measurement of the input register in the computational basis yields $c = ks/r$.
- The algorithm is run a number of times until a value of k coprime to r is obtained. Cancelling c/s to lowest terms then yields k and r as k/r.

Offset shifted into phase factor

- This yields the transition:

$$\frac{1}{\sqrt{\frac{s}{r}}}\sum_{j=0}^{\frac{s}{r}-1}|x_i + jr\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{2\pi i \frac{x_i k}{r}}|ks/r\rangle \qquad (33)$$

- Shifts the offset into a phase factor and inverts the period as a multiple of s/r.
- A measurement of the input register in the computational basis yields $c = ks/r$.
- The algorithm is run a number of times until a value of k coprime to r is obtained. Cancelling c/s to lowest terms then yields k and r as k/r.

Offset shifted into phase factor

- This yields the transition:

$$\frac{1}{\sqrt{\frac{s}{r}}}\sum_{j=0}^{\frac{s}{r}-1}|x_i+jr\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{2\pi i\frac{x_i k}{r}}|ks/r\rangle \qquad (33)$$

- Shifts the offset into a phase factor and inverts the period as a multiple of s/r.
- A measurement of the input register in the computational basis yields $c = ks/r$.
- The algorithm is run a number of times until a value of k coprime to r is obtained. Cancelling c/s to lowest terms then yields k and r as k/r.

Randomized algorithm

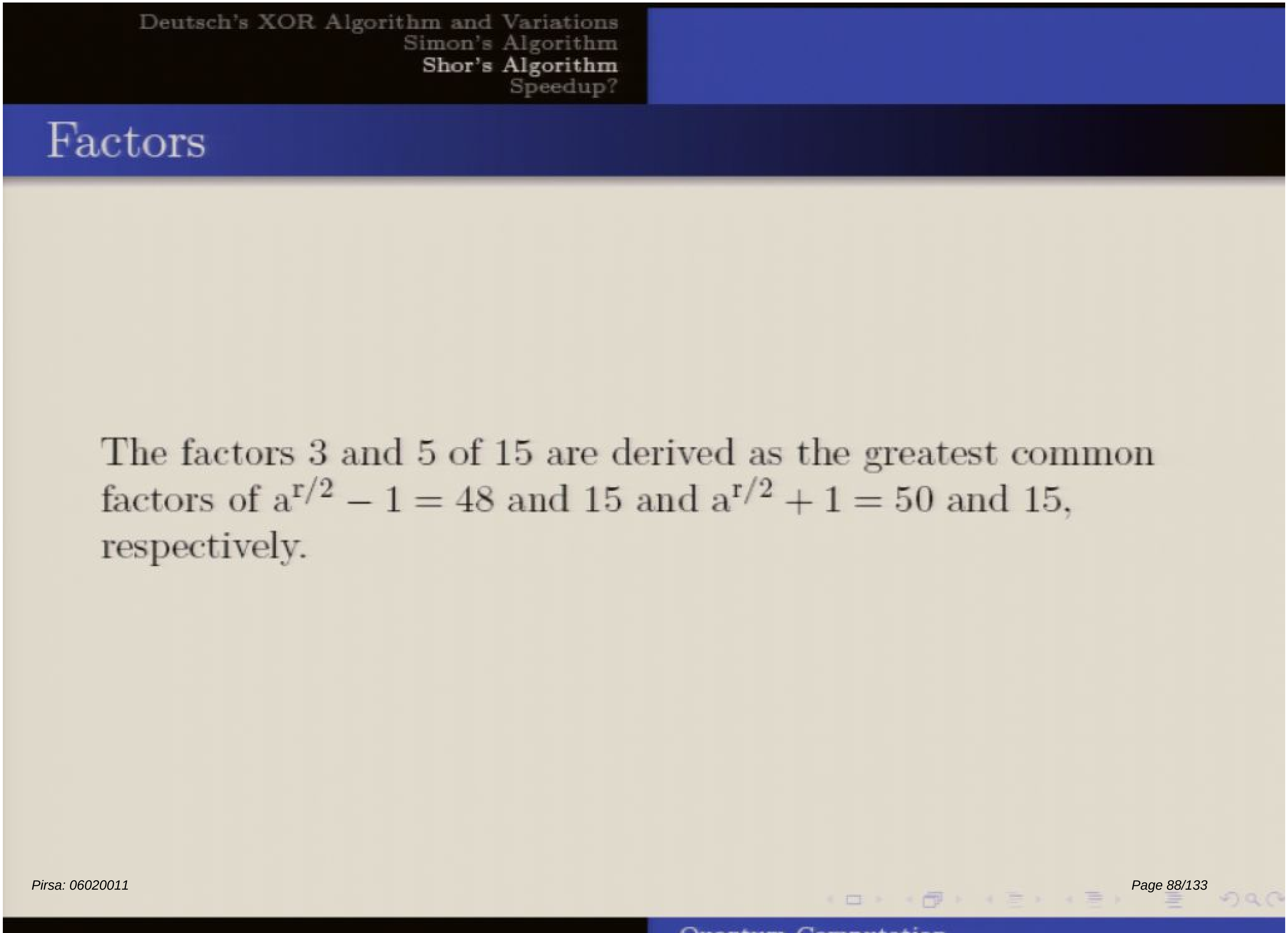
- We don't know the value of r in advance of applying the algorithm, so we don't recognize when a measurement outcome yields a value of k coprime to r.
- The idea is to run the algorithm, cancel c/s to lowest terms to obtain a candidate value for r and hence a candidate factor of N, which can then be tested by division into N.
- Even when we do obtain a value of k coprime to r, some values of a will yield a period for which the method fails to yield a factor of N, in which case we randomly choose a new value of a and run the algorithm with this value.

Geometrical picture

- Consider case N = 15, a = 7 and s = 64 (discussed in Barenco).
- The function $f(x) = a^x \bmod 15$ is:

$$\begin{aligned} 7^0 \bmod 15 &= 1 \\ 7^1 \bmod 15 &= 7 \\ 7^2 \bmod 15 &= 4 \\ 7^3 \bmod 15 &= 13 \\ 7^4 \bmod 15 &= 1 \\ &\vdots \end{aligned}$$

and the period is evidently r = 4.



Factors

The factors 3 and 5 of 15 are derived as the greatest common factors of $a^{r/2} - 1 = 48$ and 15 and $a^{r/2} + 1 = 50$ and 15, respectively.

Final state after unitary evolution

After the application of the unitary transformation $U_f = a^x \bmod N$, the state of the two registers is:

$$\begin{aligned} &\tfrac{1}{8}(|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + |3\rangle|13\rangle \\ &+ |4\rangle|1\rangle + |5\rangle|7\rangle + |6\rangle|4\rangle + |7\rangle|13\rangle \\ &\vdots \\ &+ |60\rangle|1\rangle + |61\rangle|7\rangle + |62\rangle|4\rangle + |63\rangle|13\rangle) \end{aligned} \tag{34}$$

Final state after unitary evolution

This state can be expressed as:

$$\begin{aligned}&\tfrac{1}{4}(|0\rangle + |4\rangle + |8\rangle + \ldots + |60\rangle)|1\rangle \\ &+\tfrac{1}{4}(|1\rangle + |5\rangle + |9\rangle + \ldots + |61\rangle)|7\rangle \\ &+\tfrac{1}{4}(|2\rangle + |6\rangle + |10\rangle + \ldots + |62\rangle)|4\rangle \\ &+\tfrac{1}{4}(|3\rangle + |7\rangle + |11\rangle + \ldots + |63\rangle)|13\rangle)\end{aligned} \tag{35}$$

Input states

- If we measure the output register, we obtain (equiprobably) one of four states for the input register, depending on the outcome of the measurement: 1, 7, 4, or 13:

$$\frac{1}{4}(|0\rangle + |4\rangle + |8\rangle + \ldots + |60\rangle) \tag{36}$$
$$\frac{1}{4}(|1\rangle + |5\rangle + |9\rangle + \ldots + |61\rangle) \tag{37}$$
$$\frac{1}{4}(|2\rangle + |6\rangle + |10\rangle + \ldots + |62\rangle) \tag{38}$$
$$\frac{1}{4}(|3\rangle + |7\rangle + |11\rangle + \ldots + |63\rangle) \tag{39}$$

- These are the states

$$\frac{1}{\sqrt{s/r}} \sum_{j=0}^{s/r-1} |x_i + jr\rangle \tag{40}$$

for values of the offset 0, 1, 2, 3.

Input states

- If we measure the output register, we obtain (equiprobably) one of four states for the input register, depending on the outcome of the measurement: 1, 7, 4, or 13:

$$\tfrac{1}{4}(|0\rangle + |4\rangle + |8\rangle + \ldots + |60\rangle) \tag{36}$$
$$\tfrac{1}{4}(|1\rangle + |5\rangle + |9\rangle + \ldots + |61\rangle) \tag{37}$$
$$\tfrac{1}{4}(|2\rangle + |6\rangle + |10\rangle + \ldots + |62\rangle) \tag{38}$$
$$\tfrac{1}{4}(|3\rangle + |7\rangle + |11\rangle + \ldots + |63\rangle) \tag{39}$$

- These are the states

$$\frac{1}{\sqrt{s/r}} \sum_{j=0}^{s/r-1} |x_i + jr\rangle \tag{40}$$

for values of the offset 0, 1, 2, 3.

Input states after Fourier transform

- Application of the discrete Fourier transform yields:

$$\begin{aligned}
x_1 = 0 &: \ \tfrac{1}{2}(|0\rangle + |16\rangle + |32\rangle + |48\rangle) \\
x_7 = 1 &: \ \tfrac{1}{2}(|0\rangle + i|16\rangle - |32\rangle - i|48\rangle) \\
x_4 = 2 &: \ \tfrac{1}{2}(|0\rangle - |16\rangle + |32\rangle - |48\rangle) \\
x_{13} = 3 &: \ \tfrac{1}{2}(|0\rangle - i|16\rangle - |32\rangle + i|48\rangle)
\end{aligned}$$

  which are the states in

$$\frac{1}{\sqrt{\frac{s}{r}}} \sum_{j=0}^{\frac{s}{r}-1} |x_i + jr\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{2\pi i \frac{x_i k}{r}} |ks/r\rangle \qquad (41)$$

- So for the period $r = 4$, the state of the input register ends up in the 4-dimensional subspace spanned by the vectors $|0\rangle, |16\rangle, |32\rangle, |48\rangle$.

Input states after Fourier transform

- Application of the discrete Fourier transform yields:

$$\begin{aligned} x_1 = 0: &\ \tfrac{1}{2}(|0\rangle + |16\rangle + |32\rangle + |48\rangle) \\ x_7 = 1: &\ \tfrac{1}{2}(|0\rangle + i|16\rangle - |32\rangle - i|48\rangle) \\ x_4 = 2: &\ \tfrac{1}{2}(|0\rangle - |16\rangle + |32\rangle - |48\rangle) \\ x_{13} = 3: &\ \tfrac{1}{2}(|0\rangle - i|16\rangle - |32\rangle + i|48\rangle) \end{aligned}$$

which are the states in

$$\frac{1}{\sqrt{\frac{s}{r}}}\sum_{j=0}^{\frac{s}{r}-1} |x_i + jr\rangle \xrightarrow{U_{DFT_s}} \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} e^{2\pi i \frac{x_i k}{r}} |ks/r\rangle \qquad (41)$$

- So for the period $r = 4$, the state of the input register ends up in the 4-dimensional subspace spanned by the vectors $|0\rangle, |16\rangle, |32\rangle, |48\rangle$.

Geometric Picture

- Consider all possible even periods r for which $f(x) = a^x \bmod 15$, where a is coprime to 15.
- The other possible values of a are 2, 4, 8, 11, 13, 14 and the corresponding periods turn out to be 4, 2, 4, 2, 4, 2. So we need only consider $r = 2$.
- Note: every value of a except $a = 14$ yields the correct factors for 15. For $a = 14$, the method fails: $r = 2$, so $a^{\frac{r}{2}} = -1 \bmod 15$.

Geometric Picture

- Consider all possible even periods r for which $f(x) = a^x \bmod 15$, where a is coprime to 15.
- The other possible values of a are 2, 4, 8, 11, 13, 14 and the corresponding periods turn out to be 4, 2, 4, 2, 4, 2. So we need only consider $r = 2$.
- Note: every value of a except $a = 14$ yields the correct factors for 15. For $a = 14$, the method fails: $r = 2$, so $a^{\frac{r}{2}} = -1 \bmod 15$.

Geometric Picture

- Consider all possible even periods r for which $f(x) = a^x \bmod 15$, where a is coprime to 15.
- The other possible values of a are 2, 4, 8, 11, 13, 14 and the corresponding periods turn out to be 4, 2, 4, 2, 4, 2. So we need only consider $r = 2$.
- Note: every value of a except $a = 14$ yields the correct factors for 15. For $a = 14$, the method fails: $r = 2$, so $a^{\frac{r}{2}} = -1 \bmod 15$.

Geometric picture

- For r = 2, if we measure the output register, we will obtain (equiprobably) one of two states for the input register, depending on the outcome of the measurement (say, a or b):

$$|0\rangle + |2\rangle + |4\rangle + \ldots + |62\rangle \tag{42}$$

$$|1\rangle + |3\rangle + |5\rangle + \ldots + |63\rangle \tag{43}$$

- After the discrete Fourier transform, these states are transformed to:

$$x_a = 0 : \ |0\rangle + |32\rangle$$
$$x_b = 1 : \ |0\rangle - |32\rangle$$

Geometric picture

- For r = 2, if we measure the output register, we will obtain (equiprobably) one of two states for the input register, depending on the outcome of the measurement (say, a or b):

$$|0\rangle + |2\rangle + |4\rangle + \ldots + |62\rangle \tag{42}$$
$$|1\rangle + |3\rangle + |5\rangle + \ldots + |63\rangle \tag{43}$$

- After the discrete Fourier transform, these states are transformed to:

$$x_a = 0: \ |0\rangle + |32\rangle$$
$$x_b = 1: \ |0\rangle - |32\rangle$$

Geometric picture

- In this case, the 2-dimensional subspace $\mathcal{V}_{\mathrm{r}=2}$ spanned by $|0\rangle, |32\rangle$ for $\mathrm{r}=2$ is included in the 4-dimensional subspace $\mathcal{V}_{\mathrm{r}=4}$ for $\mathrm{r}=4$.
- A measurement can distinguish $\mathrm{r}=4$ from $\mathrm{r}=2$ reliably, i.e., whether the final state of the input register is in $\mathcal{V}_{\mathrm{r}=4}$ or $\mathcal{V}_{\mathrm{r}=2}$, only if the final state is in $\mathcal{V}_{\mathrm{r}=4} - \mathcal{V}_{\mathrm{r}=2}$, the part of $\mathcal{V}_{\mathrm{r}=4}$ orthogonal to $\mathcal{V}_{\mathrm{r}=2}$.
- What happens if the final state ends up in $\mathcal{V}_{\mathrm{r}=2}$?

Geometric picture

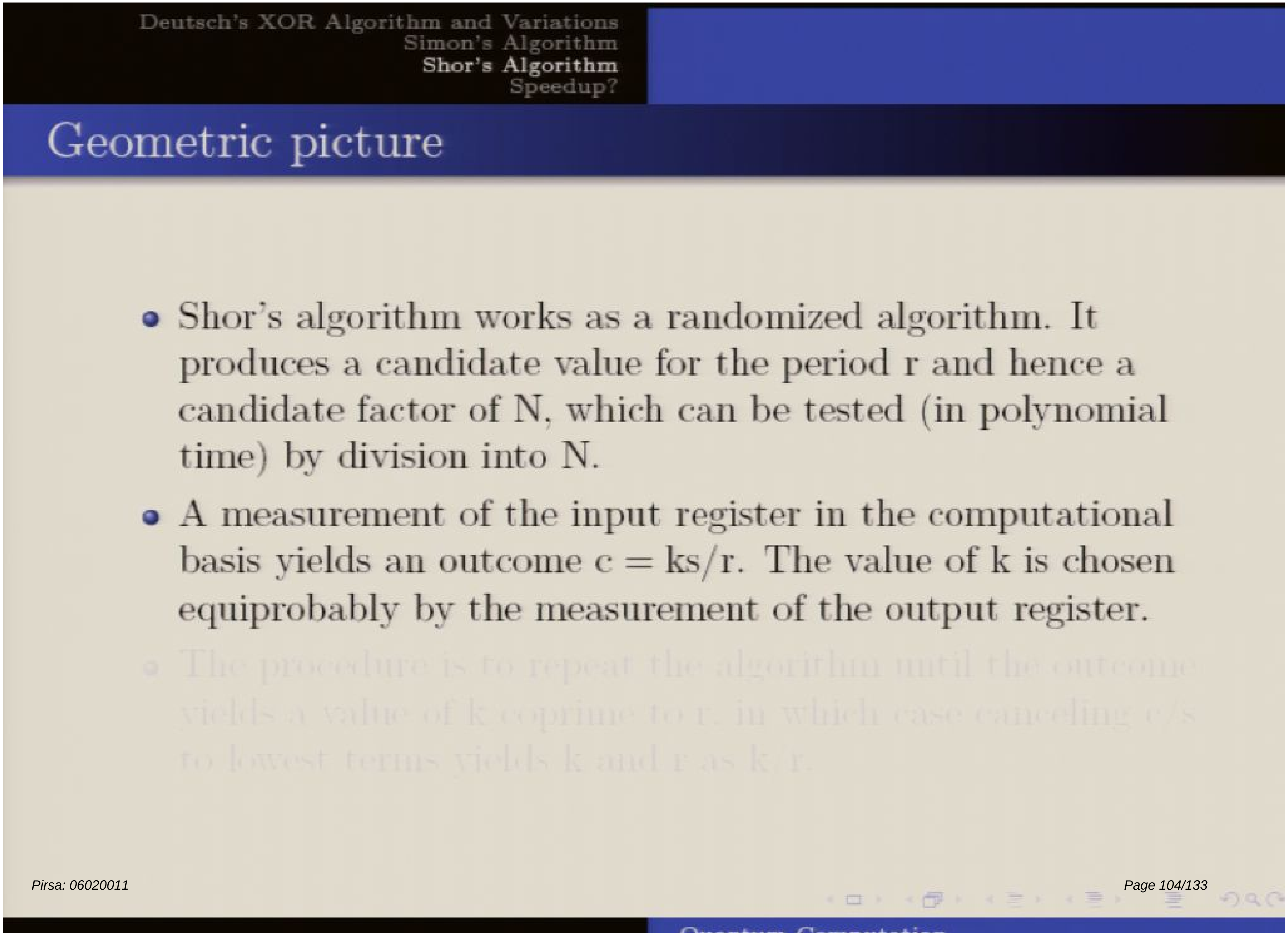
- In this case, the 2-dimensional subspace $\mathcal{V}_{\mathrm{r}=2}$ spanned by $|0\rangle, |32\rangle$ for $\mathrm{r} = 2$ is included in the 4-dimensional subspace $\mathcal{V}_{\mathrm{r}=4}$ for $\mathrm{r} = 4$.
- A measurement can distinguish $\mathrm{r} = 4$ from $\mathrm{r} = 2$ reliably, i.e., whether the final state of the input register is in $\mathcal{V}_{\mathrm{r}=4}$ or $\mathcal{V}_{\mathrm{r}=2}$, only if the final state is in $\mathcal{V}_{\mathrm{r}=4} - \mathcal{V}_{\mathrm{r}=2}$, the part of $\mathcal{V}_{\mathrm{r}=4}$ orthogonal to $\mathcal{V}_{\mathrm{r}=2}$.
- What happens if the final state ends up in $\mathcal{V}_{\mathrm{r}=2}$?

Geometric picture

- In this case, the 2-dimensional subspace $\mathcal{V}_{\mathrm{r}=2}$ spanned by $|0\rangle, |32\rangle$ for $\mathrm{r} = 2$ is included in the 4-dimensional subspace $\mathcal{V}_{\mathrm{r}=4}$ for $\mathrm{r} = 4$.
- A measurement can distinguish $\mathrm{r} = 4$ from $\mathrm{r} = 2$ reliably, i.e., whether the final state of the input register is in $\mathcal{V}_{\mathrm{r}=4}$ or $\mathcal{V}_{\mathrm{r}=2}$, only if the final state is in $\mathcal{V}_{\mathrm{r}=4} - \mathcal{V}_{\mathrm{r}=2}$, the part of $\mathcal{V}_{\mathrm{r}=4}$ orthogonal to $\mathcal{V}_{\mathrm{r}=2}$.
- What happens if the final state ends up in $\mathcal{V}_{\mathrm{r}=2}$?

Geometric picture

- Shor's algorithm works as a randomized algorithm. It produces a candidate value for the period r and hence a candidate factor of N, which can be tested (in polynomial time) by division into N.
- A measurement of the input register in the computational basis yields an outcome c = ks/r. The value of k is chosen equiprobably by the measurement of the output register.
- The procedure is to repeat the algorithm until the outcome yields a value of k coprime to r, in which case canceling c/s to lowest terms yields k and r as k/r.



Geometric picture

- Shor's algorithm works as a randomized algorithm. It produces a candidate value for the period r and hence a candidate factor of N, which can be tested (in polynomial time) by division into N.
- A measurement of the input register in the computational basis yields an outcome $c = ks/r$. The value of k is chosen equiprobably by the measurement of the output register.
- The procedure is to repeat the algorithm until the outcome yields a value of k coprime to r, in which case canceling c/s to lowest terms yields k and r as k, r.

Geometric picture

- Shor's algorithm works as a randomized algorithm. It produces a candidate value for the period r and hence a candidate factor of N, which can be tested (in polynomial time) by division into N.
- A measurement of the input register in the computational basis yields an outcome $c = ks/r$. The value of k is chosen equiprobably by the measurement of the output register.
- The procedure is to repeat the algorithm until the outcome yields a value of k coprime to r, in which case canceling c/s to lowest terms yields k and r as k/r.

Geometric picture

- Suppose we choose a value of a with period $r = 2$ and find the value $c = 32$.
- The only value of k coprime to r is $k = 1$. Then c/s cancelled to lowest terms is 1/2, which yields the correct period, and hence the correct factors of N.
- But $c = 32$ could also be obtained for $a = 7$, $r = 4$, and $k = 2$, which does not yield the correct period, and hence does not yield the correct factors of N.

Geometric picture

- Suppose we choose a value of a with period $r = 2$ and find the value $c = 32$.
- The only value of k coprime to r is $k = 1$. Then c/s cancelled to lowest terms is 1/2, which yields the correct period, and hence the correct factors of N.
- But $c = 32$ could also be obtained for $a = 7$, $r = 4$, and $k = 2$, which does not yield the correct period, and hence does not yield the correct factors of N.

Geometric picture

- Putting it geometrically: the value $k = 1$ for $r = 2$ corresponds to the same state, $|32\rangle$, as the value $k = 2$ for $r = 4$.
- Once we obtain the candidate period $r = 2$ (by cancelling $c/s = 32/64$ to lowest terms), we calculate the factors of N as the greatest common factors of $a \pm 1$ and N and test these by division into N.
- If $a = 7$, these calculated factors will be incorrect. If $a = 2$, say, the factors calculated in this way will be correct.

Geometric picture

- Putting it geometrically: the value $k = 1$ for $r = 2$ corresponds to the same state, $|32\rangle$, as the value $k = 2$ for $r = 4$.
- Once we obtain the candidate period $r = 2$ (by cancelling $c/s = 32/64$ to lowest terms), we calculate the factors of N as the greatest common factors of $a \pm 1$ and N and test these by division into N.
- If $a = 7$, these calculated factors will be incorrect. If $a = 2$, say, the factors calculated in this way will be correct.

Geometric picture

- Putting it geometrically: the value $k = 1$ for $r = 2$ corresponds to the same state, $|32\rangle$, as the value $k = 2$ for $r = 4$.
- Once we obtain the candidate period $r = 2$ (by cancelling $c/s = 32/64$ to lowest terms), we calculate the factors of N as the greatest common factors of $a \pm 1$ and N and test these by division into N.
- If $a = 7$, these calculated factors will be incorrect. If $a = 2$, say, the factors calculated in this way will be correct.

Geometric picture

With the added information provided by the outcome of a test division of a candidate factor into N, Shor's randomized algorithm again amounts to determining which disjunction among alternative disjunctions is true, i.e., which subspace contains the state, without determining the truth values of the disjuncts.

What feature is responsible for the exponential speedup?

What, precisely, is the feature of a quantum computer responsible for the phenomenal efficiency over a classical computer? In the case of Simon's algorithm, the speed-up is exponential over any classical algorithm; in the case of Shor's algorithm, the speed-up is exponential over any known classical algorithm.

Deutsch's view

- The first stage of a quantum algorithm involves the creation of a state in which every input value to the function is correlated with a corresponding output value. Deutsch cites this 'quantum parallelism' as the source of the speed-up in a quantum computation.
- The idea is that a quantum computation is something like a massively parallel classical computation, for all possible values of a function, with the parallel computations taking place in parallel universes.
- For a critique, see Steane (2003).

Deutsch's view

- The first stage of a quantum algorithm involves the creation of a state in which every input value to the function is correlated with a corresponding output value. Deutsch cites this 'quantum parallelism' as the source of the speed-up in a quantum computation.
- The idea is that a quantum computation is something like a massively parallel classical computation, for all possible values of a function, with the parallel computations taking place in parallel universes
- For a critique, see Steane (2003).

Deutsch's view

- The first stage of a quantum algorithm involves the creation of a state in which every input value to the function is correlated with a corresponding output value. Deutsch cites this ‘quantum parallelism’ as the source of the speed-up in a quantum computation.
- The idea is that a quantum computation is something like a massively parallel classical computation, for all possible values of a function, with the parallel computations taking place in parallel universes
- For a critique, see Steane (2003).

Steane's view (1998)

The period finding algorithm appears at first sight like a conjuring trick: it is not quite clear how the quantum computer managed to produce the period like a rabbit out of a hat. ...I would say that the most important features are contained in $[|\psi\rangle = \frac{1}{s}\sum_{x=0}^{s-1}|x\rangle|f(x)\rangle]$. ...The 'magic' happens when a measurement of the y register produces the special state $[\frac{1}{s/r}\sum_{j=0}^{s/r-1}|x_i + jr\rangle]$ in the x-register, and it is quantum entanglement which permits this. The final Fourier transform can be regarded as an interference between the various superposed states in the x-register (compare with the action of a diffraction grating).

Steane (1998)

Jozsa's view

Jozsa (1997) points out that the state space (phase space) of a composite classical system is the Cartesian product of the state spaces of its subsystems, while the state space of a composite quantum system is the tensor product of the state spaces of its subsystems.

Jozsa's view

- For n qubits, the quantum state space has 2^n dimensions. So the information required to represent a general state increases exponentially with n: even if we restrict the specification of the amplitudes to numbers of finite precision, a superposition will in general have $\mathcal{O}(2^n)$ components.
- For a classical composite system of n two-level subsystems, the number of possible states grows exponentially with n, but the information required to represent a general state is just n times the information required to represent a single two-level system, i.e., the information grows only linearly with n because the state of a composite system is just a product state.

Entanglement and speedup

- Jozsa and Linden (2002) have shown that a quantum algorithm operating on pure states can achieve an exponential speed-up over classical algorithms only if the quantum algorithm involves multi-partite entanglement that increases unboundedly with the input size.
- Vidal (2003) has shown that a classical computer can simulate the evolution of a pure state of n qubits with computational resources that grow linearly with n and exponentially in multi-partite entanglement.

Steane's view (2003), my view

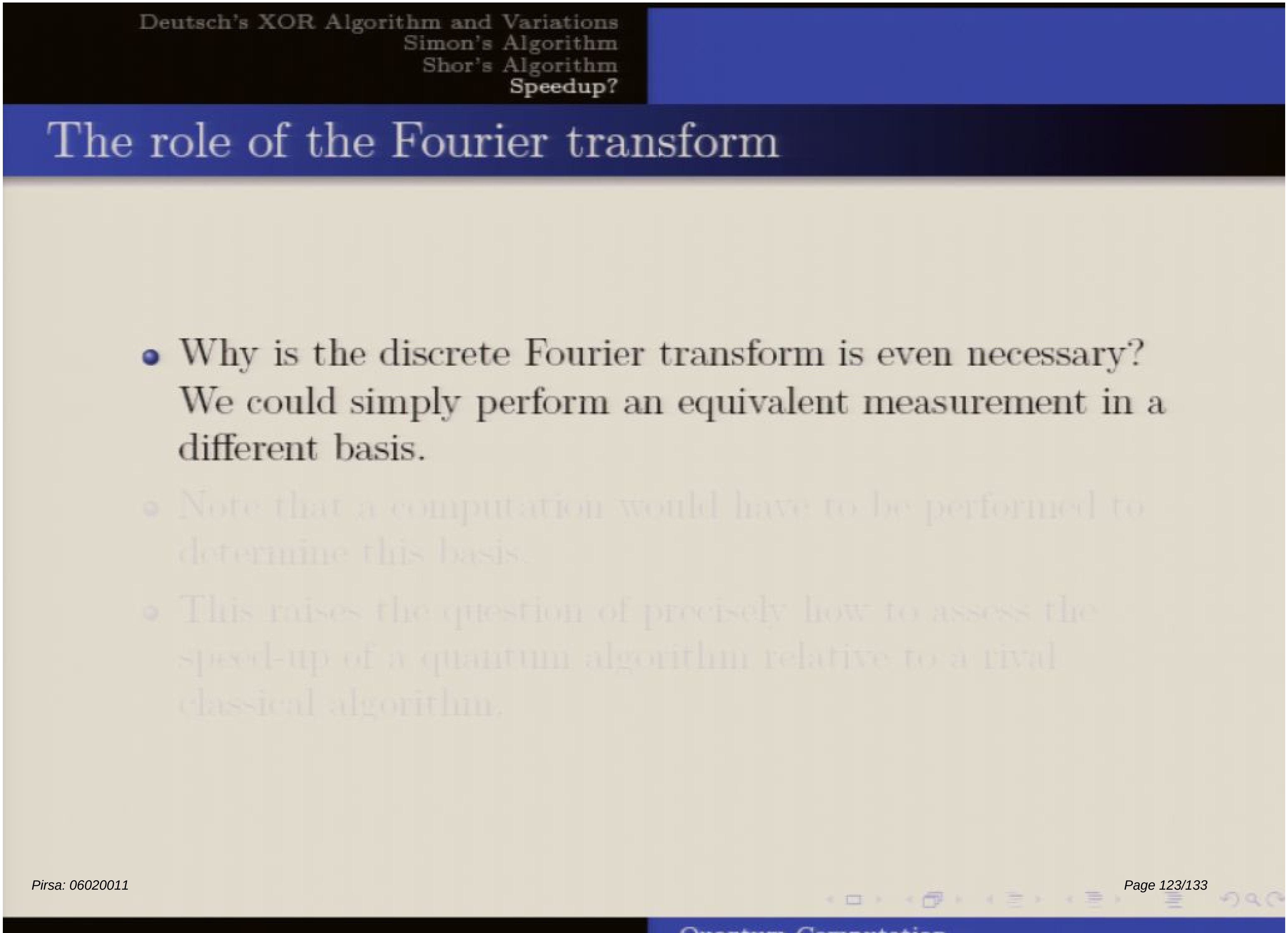
The essential feature of the quantum computations discussed above is the selection of a disjunction, representing a global property of a function, among alternative possible disjunctions without computing the truth values of the disjuncts, which is redundant information in a quantum computation but essential information classically.

The role of the Fourier transform

- It would be incorrect to attribute the efficiency of these quantum algorithms to the interference in the input register produced by the Fourier transform.
- The role of the Fourier transform is simply to allow a measurement in the computational basis to reveal which subspace representing the target disjunction contains the state.

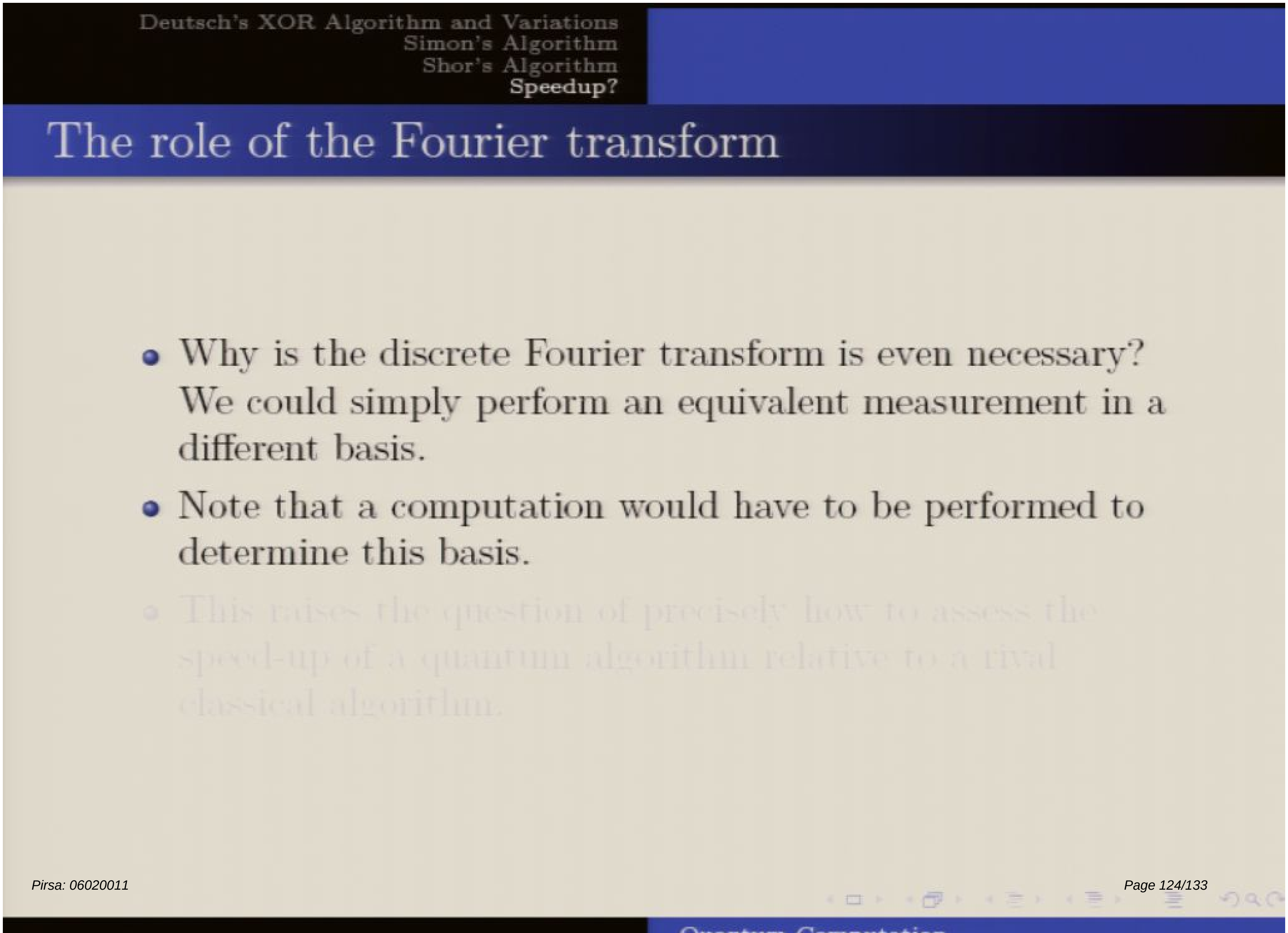
The role of the Fourier transform

- It would be incorrect to attribute the efficiency of these quantum algorithms to the interference in the input register produced by the Fourier transform.
- The role of the Fourier transform is simply to allow a measurement in the computational basis to reveal which subspace representing the target disjunction contains the state.



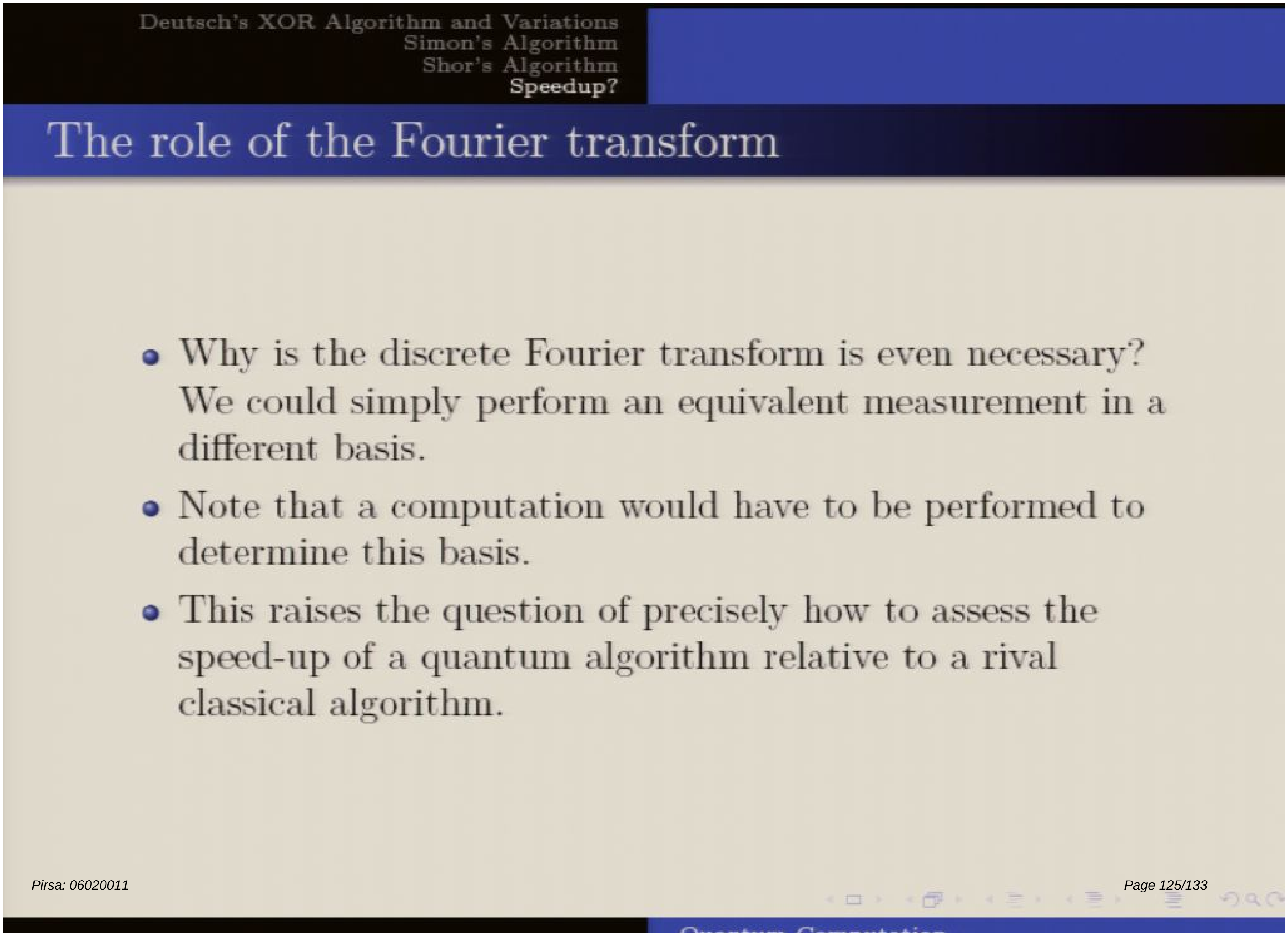
The role of the Fourier transform

- Why is the discrete Fourier transform is even necessary? We could simply perform an equivalent measurement in a different basis.
- Note that a computation would have to be performed to determine this basis.
- This raises the question of precisely how to assess the speed-up of a quantum algorithm relative to a rival classical algorithm.



The role of the Fourier transform

- Why is the discrete Fourier transform is even necessary? We could simply perform an equivalent measurement in a different basis.
- Note that a computation would have to be performed to determine this basis.
- This raises the question of precisely how to assess the speed-up of a quantum algorithm relative to a rival classical algorithm.



The role of the Fourier transform

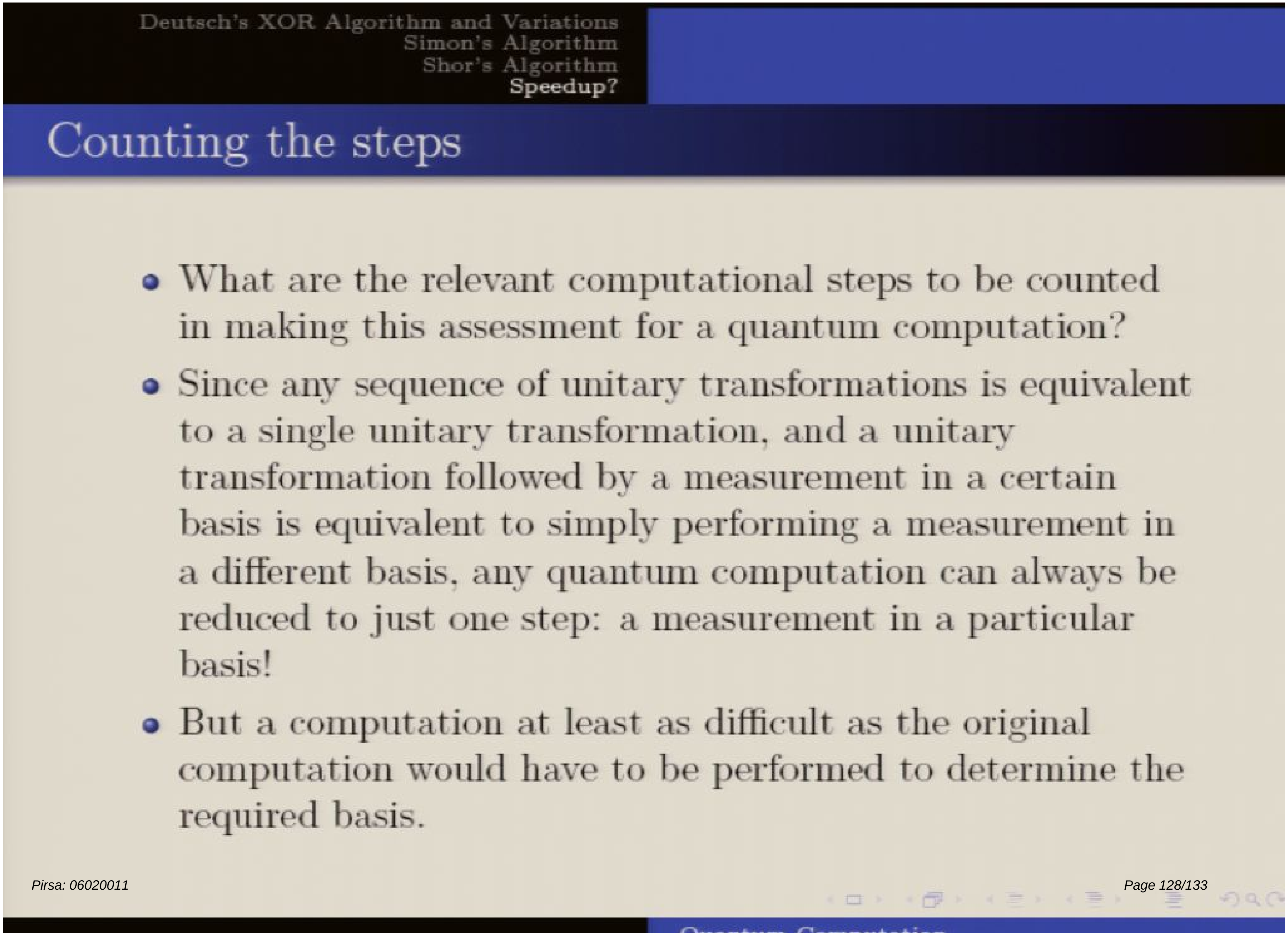
- Why is the discrete Fourier transform is even necessary? We could simply perform an equivalent measurement in a different basis.
- Note that a computation would have to be performed to determine this basis.
- This raises the question of precisely how to assess the speed-up of a quantum algorithm relative to a rival classical algorithm.

Counting the steps

- What are the relevant computational steps to be counted in making this assessment for a quantum computation?
- Since any sequence of unitary transformations is equivalent to a single unitary transformation, and a unitary transformation followed by a measurement in a certain basis is equivalent to simply performing a measurement in a different basis, any quantum computation can always be reduced to just one step: a measurement in a particular basis!
- But a computation at least as difficult as the original computation would have to be performed to determine the required basis.

Counting the steps

- What are the relevant computational steps to be counted in making this assessment for a quantum computation?
- Since any sequence of unitary transformations is equivalent to a single unitary transformation, and a unitary transformation followed by a measurement in a certain basis is equivalent to simply performing a measurement in a different basis, any quantum computation can always be reduced to just one step: a measurement in a particular basis!
- But a computation at least as difficult as the original computation would have to be performed to determine the required basis.



Counting the steps

- What are the relevant computational steps to be counted in making this assessment for a quantum computation?
- Since any sequence of unitary transformations is equivalent to a single unitary transformation, and a unitary transformation followed by a measurement in a certain basis is equivalent to simply performing a measurement in a different basis, any quantum computation can always be reduced to just one step: a measurement in a particular basis!
- But a computation at least as difficult as the original computation would have to be performed to determine the required basis.

Counting the steps

- Some convention is required about what steps to count in a quantum computation.
- Accepted convention is to require the unitary transformations in a quantum computation to be constructed from elementary quantum gates that form a universal set and to count each such gate as one step.
- All measurements are required to be performed in the computational basis, and these are counted as additional steps.

Counting the steps

- Some convention is required about what steps to count in a quantum computation.
- Accepted convention is to require the unitary transformations in a quantum computation to be constructed from elementary quantum gates that form a universal set and to count each such gate as one step.
- All measurements are required to be performed in the computational basis, and these are counted as additional steps.

Counting the steps

- Some convention is required about what steps to count in a quantum computation.
- Accepted convention is to require the unitary transformations in a quantum computation to be constructed from elementary quantum gates that form a universal set and to count each such gate as one step.
- All measurements are required to be performed in the computational basis, and these are counted as additional steps.

Counting the steps

The final discrete Fourier transform is indispensable in transforming the state so that the quantum algorithms can be completed by measurements in the computational basis, and it is an important feature of the algorithms that the Fourier transform can be implemented efficiently with elementary unitary gates.

Quantum computational speedup

To claim that a quantum algorithm is exponentially faster than a classical algorithm is to claim that the number of steps counted in this way for the quantum algorithm is a polynomial function of the size of the input (the number of qubits required to store the input), while the classical algorithm involves a number of steps that increases exponentially with the size of the input (the number of bits required to store the input).