Shor's Algorithm

Special Topics in Computer Science:
Quantum Computing

CSC591/ECE592 – Fall 2019

- This presentation is a qualitative outline for implementing Shor's algorithm
- There is an extensive amount of additional mathematics that cannot be covered in one lecture
- To fully understand Shor's algorithm a much more detailed study of this topic needs to be undertaken to prove each step for a complete presentation
- A detailed set of references provided at the end of this presentation that expands in detail the complexity of the calculations needed to prove Shor's algorithm

Outline

- Preliminaries
  - Review of some basic mathematics relevant to Shor's algorithm
  - Introduction to Period Finding
- Simon's Algorithm
- Shor's Algorithm
- References

Math Review: Factoring and Modular Arithmetic

Factoring Numbers

- "Factors" are the numbers you multiply to get another number.
- Example: Let N=60 = 2^2 x 3 x 5 or $P_1^{e_1} P_2^{e_2} \ldots\ldots P_k^{e_k}$
- Select P and Q to be odd integer primes numbers
- N = PQ is a very large number and n is the length of N in bits
- Digital computers requires times $O(2^n)$ to factor these numbers
- State of the art factoring corresponds to 750 bits or ~ 200 decimal digits
- Impossible for digital computers today or anytime in the foreseeable future to be able to factor such numbers if P and Q are selected larger than 750 bits

Modular Arithmetic

- $a \equiv b \pmod{N}$ (a is congruent to (mod N)
- $b=qN+a$ (q is some quotient times N; a is the remainder)
- Examples
  - $24 \pmod{21} = 3$
  - $35 \pmod{21} = 14$
  - $20 \pmod{21} = -1$

Modular Arithmetic

- Addition

  24 (mod 21) + 35 (mod 21) = 3 +14 = 17 (mod 21)

- Multiplication

  24 (mod 21) x 30 (mod 21) = 3 x 9 = 27 = 6 (mod 21)

Arithmetic mod N

- Greatest common divisor

- Example gcd(15, 21) → gcd (3x5, 3x7) → 3
- Euclid's algorithm

21 = 1 x 15 + 6

15 = 2 x 6 + 3

6 = 2 x 3 + 0

GCD is the last non-zero remainder (in this case 3)

Notation: c|a → one integer c divides another integer a evenly (0 remainder)

Factoring

- Factoring consists of writing a number as a product of several factors usually smaller or simpler objects of the same kind
- Example – factor 21
  - Need to solve the equation $x^2 \equiv 1 \pmod{21}$ whose square root is such that get 1 for a given (mod N)
  - Choose trivial option x =1 get $x^2 \equiv 1 \pmod{21}$
  - Choose trivial option x =-1 get $x^2 \equiv 20 \pmod{21}$
    - 20^2 gives 400 (mod 21) = 1 (mod 21)
    - In general whatever N is N-1 mod squared is 1 (mod N)
- The question is are there other integers that satisfy

  $x^2 \equiv 1 \pmod{21}$ such that get 1

Factoring (cont'd)

- Make a guess x = 8
  - $8^2 = 64$ and 64 (mod 21) = 1 (mod 21)
- Consider another option
  - $8^2 - 1^2 = 0$ (mod 21)
  - $8^2 - 1^2 = (8 + 1)(8 - 1)$
  - Neither (8 + 1) or (8 - 1) can be divided into 21
  - Factorize 21 as 3 x 7
  - Now one part of (8 + 1)(8 - 1) divides each factor
  - Recover the prime factors or 21 by
    - gcd (21, 8+1) = 3
    - gcd (21, 8-1) = 7

Factoring (cont'd)

- What about x = -8 ?
  - X = -8 ≡ 13 (mod 21)
  - 13 2 = 169 ≡ 1 (mod 21)
- The numbers +8, -8 and 13 are the non-trivial square roots of 1(mod 21)

- Ultimate goal is to calculate non-trivial square roots of (mod N)
- **Need to finds values for x such that**

  $x \not\equiv +1 \pmod N$ or $x \not\equiv -1 \pmod N$

  but

  $x^2 \equiv 1 \pmod N$ such that N divides product (x+1)(x-1) but not individually

  Recover prime factors product 2 primes from gcd (N, x+1) and gcd (N, x-1)

Factoring (cont'd)

- Ultimate goal is to discover these non-trivial factors
- Procedure
  - For any given N pick a random number and calculate powers of that number
  - Calculate x^r until get $x^r \equiv 1 \pmod N$
  - If r is even then $(x^{r/2})^2 \equiv 1 \pmod N$

    and

    $x^{r/2}\, x \not\equiv +1 \pmod N$ or $x \not\equiv -1 \pmod N$
  - These r values will then give non-trivial square root values of 1(mod N)
  - This will allow calculation of the factors of N

Periodicity

- For example let N=21 and pick an example value x=2
- This give the following table

2^0	1mod(21)
2^1	2mod(21)
2^2	4mod(21)
2^3	8mod(21)
2^4	16mod(21)
2^5	11mod(21)
2^6	1mod(21)

- $2^6=2^3x2^3$ and note that $(2^3)^2=8^2=1$mod(21)
- Discovered the value 8 as a non-trivial square root of (mod 21)

r	$f(r) = x^r \pmod N$
2^0	1mod(21)
2^1	2mod(21)
2^2	4mod(21)
2^3	8mod(21)
2^4	16mod(21)
2^5	11mod(21)
2^6	1mod(21)
2^7	2mod(21)
2^8	4mod(21)
2^9	8mod(21)
2^{10}	16mod(21)
2^{11}	11mod(21)
2^{12}	1mod(21)
2^{13}	2mod(21)
2^{14}	4mod(21)
2^{15}	8mod(21)

Mathematical Lemma for Factoring

- Lemma

If N is an odd composite with at least two prime distinct factors and x is a uniformly random value between 0 and N-1, then the gcd (x, N) =1 has a probability of at least .5 that the order r of x (mod N) is even and $x^{r/2}$ is a non-trivial square root of 1(mod N)

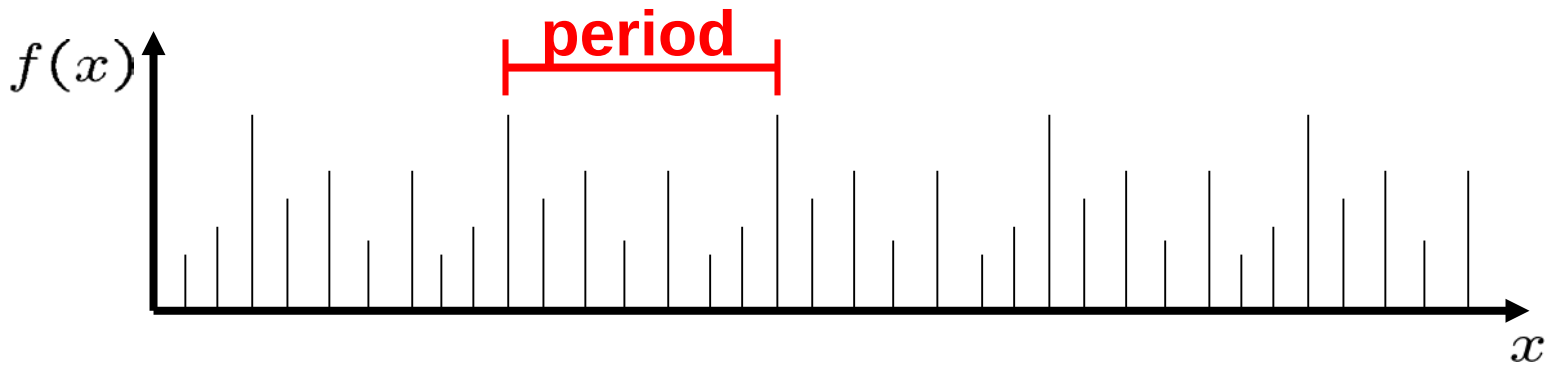
- Want to determine a periodic function from a repeating sequence of factoring

Period Finding
Main Building Block of Factoring Algorithm

Start with a function from 0,1,…,N-1 to some n bit numbers to set S
Input; f:{0,1,…N-1} → S such that for all x, f(x) = f(x+r)

$$f : x \in \{0, 1, 2, \ldots, N-1\} \rightarrow \{0,1\}^n$$

f(x) = f(y)= f(x+r) with r≠ 0



Number of repetitions is N/r = N/7 with **no repetitions within period**

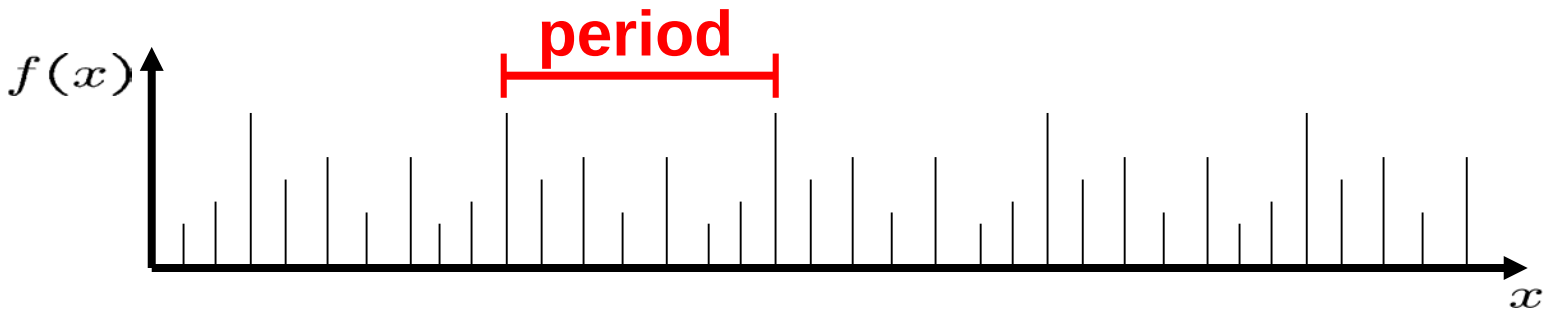
Assume that f is 1:1 within the period

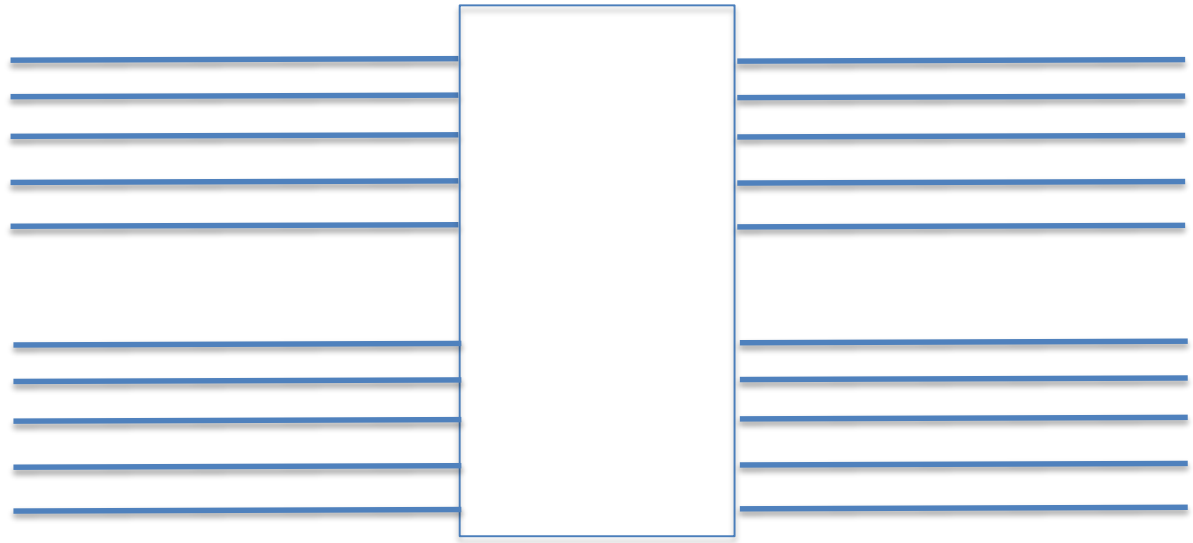
Examine simple case first: r divides N (N >> r^2)

Classical Option for Period Finding

- Selelct M as a 1000 digit number
- M ~ 10^{1000}
- r ~ $\sqrt{M}$
- r is on the order of a 500 digit number
- $\sqrt{r}$ inputs are sufficient to see a same value
- $\sqrt{r}$ is ~ 250 digit number ~ 10^{250}
- Not practical to calculate classically

Quantum Computing Option for Period Finding





superposition

|x>

|0>

|x>

|f(x)>

$$\frac{1}{\sqrt{M}}\sum_{x=0}^{M-1}|x>|f(x)>$$

$$\frac{1}{\sqrt{M}}\sum_{x=0}^{M-1}|x>|f(x)> \rightarrow \sum_{x=0}^{M-1}\alpha_x|x>$$

- Measure output
- See some random value f(r)
- Determine if there exists arithmetic progression as a repetition

Determine the value of α_x

- From the previous slide

$$\frac{1}{\sqrt{M}}\sum_{x=0}^{M-1}|x>|f(x)> \rightarrow \sum_{x=0}^{M-1}\alpha_x\,|\mathrm{x}>$$

- The α_x coefficients are only non-zero at the periodic repetition points in the summation
- Introduce Fourier sampling
  - Shift the α_x so that the 1st non-zero value is matched to x=0
  - This repetition interval is now matched to the period

Procedure

- A quantum computer can discover this period in polynomial time
- Pick a random value M ($M > 2r^2$) → ($M > 2N^2$) and construct a superposition

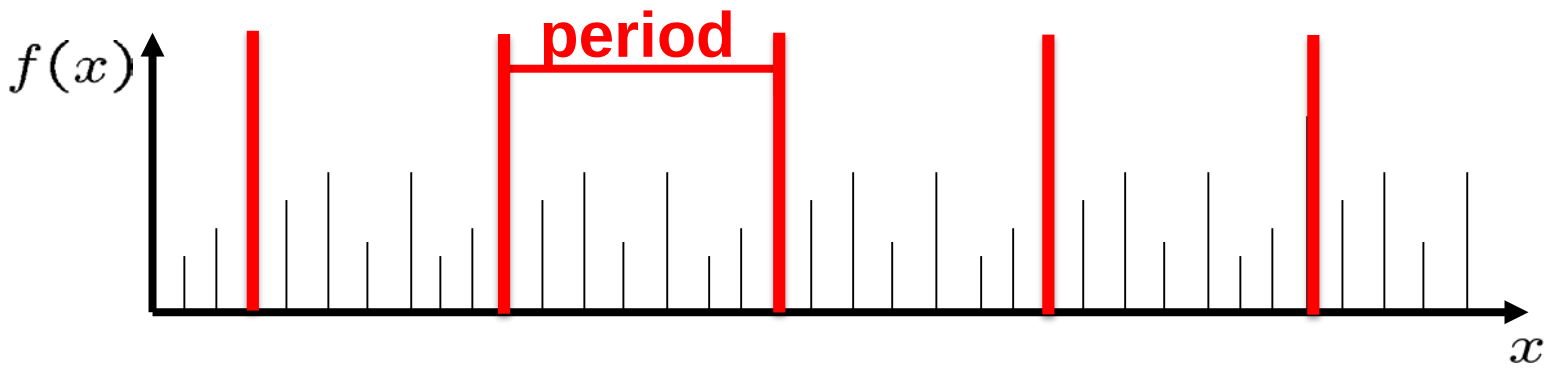
$$\frac{1}{\sqrt{M}} \sum_{r}^{M-1}$$

- Next do Fourier sampling and classically reconstruct the period r
- Check that r is even and
  - If r is even then $(x^{r/2})^2 \equiv 1 (\mathrm{mod}\ N)$

    and

    $x^{r/2}\ x \not\equiv +1 (\mathrm{mod}\ N)$ or $x \not\equiv -1 (\mathrm{mod}\ N)$
  - These r values will then give non-trivial square root values of 1(mod N)
  - This will allow calculation of the factors of N

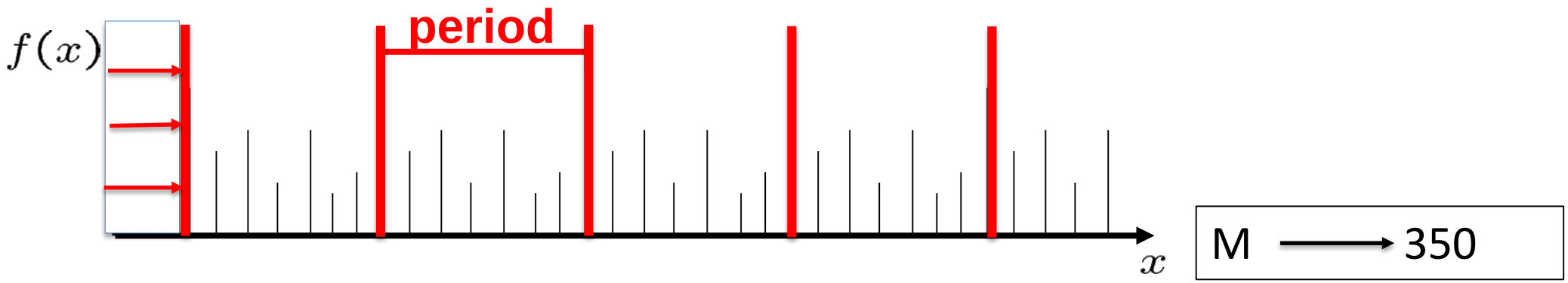
Fourier Sampling



- In the graph above the repetition is r = 7 and number of trials is M = 35
  - Shift the $\boldsymbol{\alpha_x}$ so that the 1st non-zero value is matched to x=0
  - This repetition interval is now matched to the period with amplitude $\sqrt{\frac{r}{M}}$

$$\sum_{x=0}^{M-1} \alpha_x \, |\mathrm{x}> \rightarrow \sqrt{\frac{r}{M}} \sum_{j=0}^{M-1} |jr>$$

Fourier Sampling



- In the graph above the repetition is r = 7 and number of trials is M = 350
  - Shift the $\boldsymbol{\alpha_x}$ so that the 1st non-zero value is matched to x=0
  - This repetition interval is now matched to the period with amplitude $\sqrt{\frac{r}{M}}$

$$\sum_{x=0}^{M-1} \alpha_x \, |\mathrm{x}> \rightarrow \sqrt{\frac{r}{M}} \sum_{j=0}^{\frac{M}{r}-1} |jr>$$

Measurements from the Fourier Sampling to Get Period

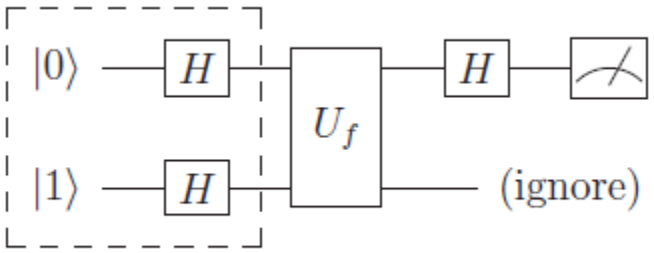
- Output from the Fourier sampling will display random multiples of $\frac{M}{r}$
- Run the Fourier sampling to get outputs
- Given that the M = 350 and r = 7 the outputs will have random values such as 100, 150, 200, etc.
- The outputs will be multiples of $\frac{M}{r}$
- Calculate the greatest common divisor

  gcd (100, 150) is 50
- Calculate r by dividing $\frac{M}{\gcd(100,150)}$=7
- This is the procedure for period finding

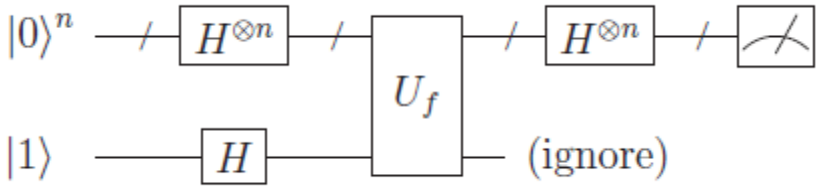
Simon’s Algorithm

Review Previously Studied Quantum Algorithm Circuits

Deutsch's Algorithm Circuit



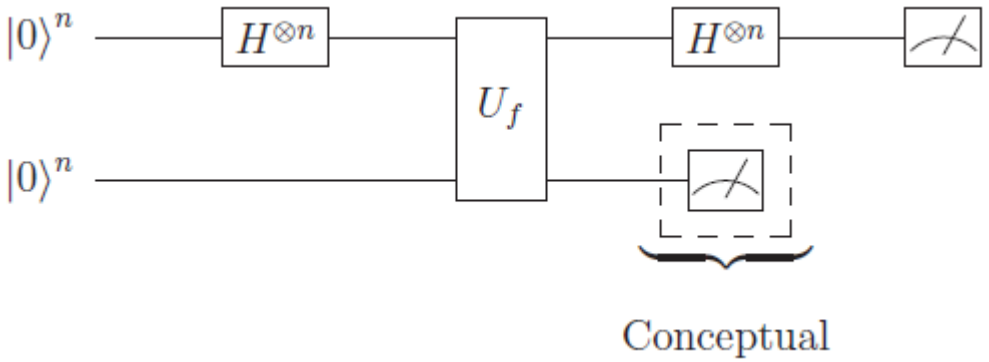
Bernstein-Vazirani Algorithm Circuit



NOTE:

- Each of these algorithms is applied to get a deterministic solution by initializing a |1> in the *B* register
- This give a *phase kick-back output and an* answer in a single evaluation

Quantum Circuit for Simon's Algorithm

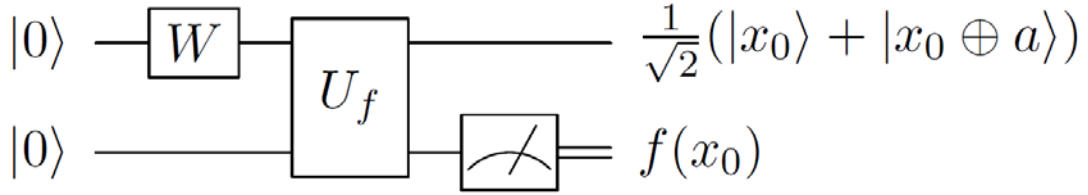


NOTE:

- Simon's algorithm is initialized in the B register with a $|0\rangle^n$
- There is no *phase kick-back output for this circuit*
- *The output of this circuit gives superposition of 2 states*

Recall From Professor Byrd's Lecture

- Complete entangled output is $\left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x>^n |f(x)>^n$
- Because the output is 2:1 (have both x and x $\oplus a$) only need to sum the B register output over half the values



$$W\left(\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)\right) = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2^n}}\sum_y ((-1)^{x_0 \cdot y} + (-1)^{(x_0 \oplus a)\cdot y})\,|y\rangle\right)$$

$$= \frac{1}{\sqrt{2^{n+1}}}\sum_y (-1)^{x_0 \cdot y}(1 + (-1)^{a \cdot y})\,|y\rangle$$

$$= \frac{2}{\sqrt{2^{n+1}}}\sum_{y \cdot a \text{ even}} (-1)^{x_0 \cdot y}\,|y\rangle$$

From Professor Byrd's Lecture
Walsh Hadamard

$$W|0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

$$\begin{aligned}
W|r\rangle &= (H \otimes \cdots \otimes H)(|r_{n-1}\rangle \otimes \cdots \otimes |r_0\rangle) \\
&= \frac{1}{\sqrt{2^n}}(|0\rangle + (-1)^{r_{n-1}}|1\rangle) \otimes \cdots \otimes (|0\rangle + (-1)^{r_0}|1\rangle) \\
&= \frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} (-1)^{s_{n-1}r_{n-1}} |s_{n-1}\rangle \otimes \cdots \otimes (-1)^{s_0 r_0} |s_0\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} (-1)^{s \cdot r} |s\rangle
\end{aligned}$$

From Professor Byrd's Lecture

Simon's Algorithm

Problem: Given a 2-to-1 function f, such that $f(x) = f(x \oplus a)$, find the hidden string a.

Create superposition $|x\rangle|f(x)\rangle$

Measure the right part, which projects the left state into $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus a\rangle)$.

Apply Walsh-Hadamard. (Details next slide.)

Measurement yields a random y such that $y \cdot a = 0 \pmod 2$.

Computation is repeated until n independent equations – about $2n$ times.

Solve for a in $O(n^2)$ steps.

Requires $O(n)$ calls to U_f, followed by $O(n^2)$ steps to solve for a.

Classical approach requires $O(2^{n/2})$ calls to f.

From Professor Byrd's Lecture

Measurement yields random y such that $y \cdot a = 0 \mod 2$, so the unknown bits of a_i of a must satisfy this equation:

$$y_0 \cdot a_0 \oplus \cdots \oplus y_{n-1} \cdot a_{n-1} = 0$$

Computation is repeated until n linearly independent equations have been found. Each time, the resulting equation has at least a 50% probability of being linearly independent of the previous equations. After repeating $2n$ times, there is a 50% chance that n linearly independent equations have been found. These equations can be solved to find a in $O(n^2)$ steps.

Simon's Algorithm Procedure

- Set up a random superposition $\frac{1}{\sqrt{2}}|r> + \frac{1}{\sqrt{2}}|r \oplus s>$ where r is a random n-bit string
- Fourier sample to get a random y: $y \cdot s = 0 (mod\ 2)$
- Repeat steps 1 and 2 n-1 times to generate linear equations in s
- Solve for s

$$\begin{pmatrix} y_1^1 s_1 & \cdots & y_n^1 s_n & \equiv 0\ (mod\ 2) \\ y_1^2 s_1 & \cdots & y_n^2 s_n & \equiv 0\ (mod\ 2) \\ y_1^{(n-1)} s_1 & \cdots & y_n^{(n-1)} s_n & \equiv 0\ (mod\ 2) \end{pmatrix}$$

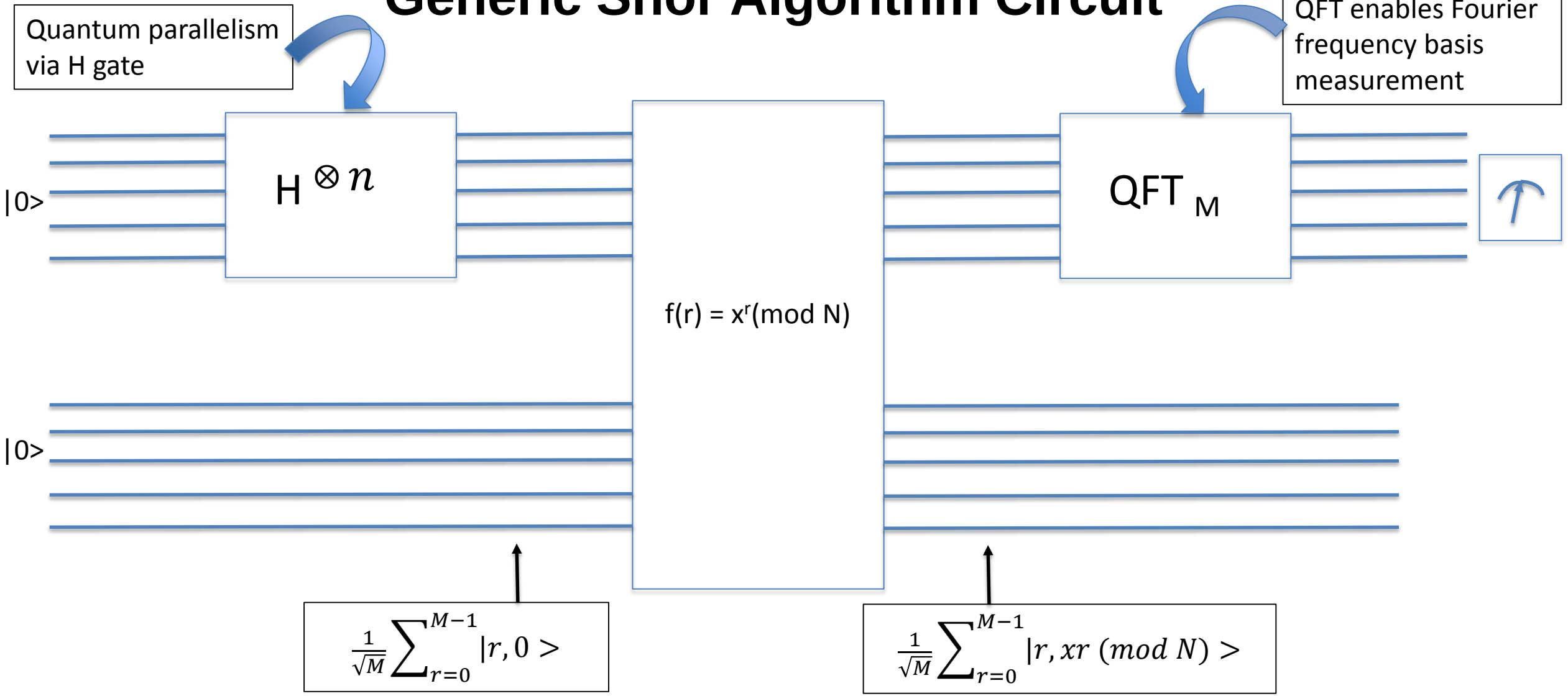
Shor’s Algorithm

Shor's Algorithm

- Statement of Shor's Periodicity Problem
- Let f: $\mathbb{Z}_M \longrightarrow \mathbb{Z}$ be injective* periodic
- A function defined on $\mathbb{Z}_M$ is called periodic injective if there exists an integer $\alpha \in \mathbb{Z}_M$ such that for all $x \neq y$ in $\mathbb{Z}_M$ there exists $f(x) = f(y)$ implies $y=x+ka$ for some integer k
- Relaxes mod 2 restriction and expands the values to include any decimal integer
- Determine the period a

* **Definition injective function** - An injective function or one-to-one function is a function that preserves distinctness: it never maps distinct elements of its domain to the same element of its codomain. In other words, every element of the function's codomain is the image of at most one element of its domain.

Generic Shor Algorithm Circuit



Gate Circuit for Shor's Algorithm

- Quantum parallelism is established in the A channel with an $H^{\otimes n}$ Hadamard gate
- The B channel is initialized with a $|0\rangle^r$ to effect a generalized Born Rule measurement
- Two analysis options at this point
  - Easy Case
  - a|N → period a divides evenly into $N=2^n$ (the original function's domain size)
  - Generalization of Simon's 2:1 procedure
  - Simon's measurement gave a superposition of 2 states
  - Shor's measurement will give a superposition of m states

Born Rule

- Born rule for a 2 qubit system (easily generalized)

$$|\Psi>^2 = \alpha|00> + \beta|01> + \gamma|10> + \delta|11> \geq \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$$

- Factor the A-kets
- $|\Psi>^2 = |0>_A(\alpha|0>_B + \beta|1>_B) + |1>_A(\gamma|0>_B + \delta|1>_B)$
- **Born Rule:** If a bipartite state is factored relative to the A-register then a measurement of A will cause the collapse of the B register (vice-versa)
- A →0 implies that B→ $\frac{(\alpha|0>+\beta|1>)}{\sqrt{|\alpha|^2+|\beta|2}}$ and if A→1 implies that B→ $\frac{(\gamma|0>+\delta|1>)}{\sqrt{|\gamma|^2+|\delta|^2}}$

Generalized Born Rule

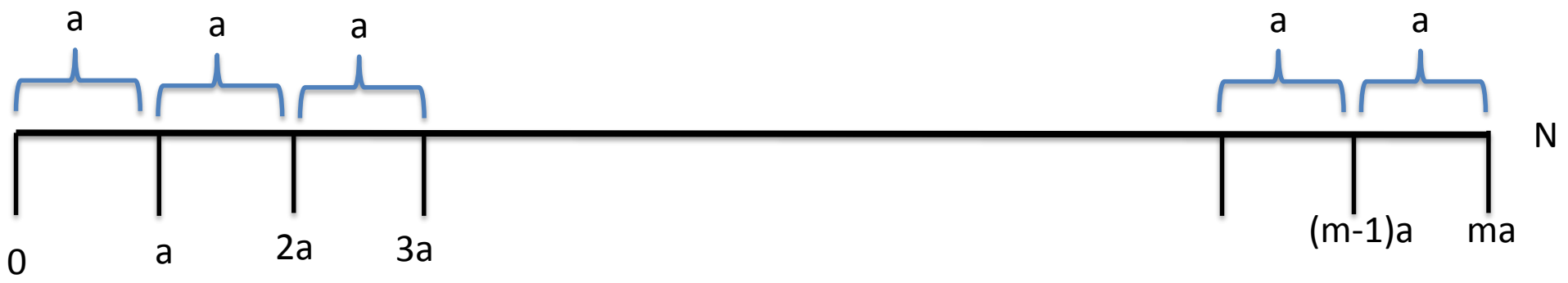
- Assume there exists an (i+j)th order state $|\varphi>^{(i+j)}$ in the product space A⊗ B $(\mathcal{H}_i \otimes \mathcal{H}_j)$ with the property that $|\varphi>^{(i+j)}$ can be written as follows

$$|\varphi>^{(i+j)} = |0>_A^i \psi_{0_B}^j + |1>_A^i \psi_{1_B}^j \ldots + |2^n - 1>_A^i |\psi_{2^n-1}>_B^j$$

- Generalized Born Rule
- A → $|k>^j$ implies that B → $\frac{|\psi_k>^i}{\sqrt{<|\psi_k|\psi_k>}}$ for k=0, 1,….2^n -1

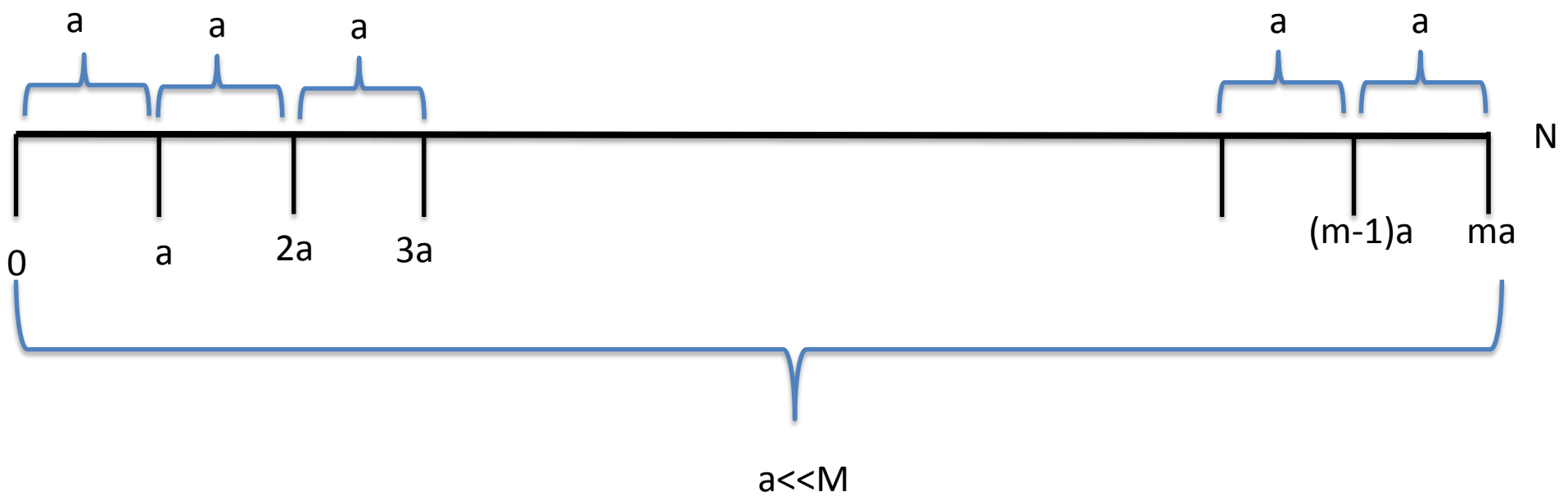
Shor's Algorithm Easy Case

- Assumption of a|N and that f has property of injective periodicity
- Domain can be completely partitioned into many disjoint cosets of size "a", each of which provides a 1-to-1 sub-domain for f



a|N → period a divides evenly into N=2^n (the original function's domain size)

- Need to also make sure that the periodicity cycles at least twice in the interval [0, M-1] (in practice want many cycles i.e. a << M)



Comparison of Shor's Algorithm with Simon's Algorithm

- Initialization for both algorithms
  - The circuit analysis is similar for both algorithms
  - Simon used mod 2 integers and Shor tackles ordinary integer values
  - In both algorithms the input to the oracle set for maximally mixed superposition in both algorithms
  - Output is a superposition of 2 states
- Post-oracle measurement
  - in the standard basis should give all possible values of with equal probabilities
  - Simon measure along the x-basis with a final Hadamard operator.
  - Shor's algorithm inserts a QFT for the post Oracle evaluation
  - QFT sets up a measurement of the function's frequency spectrum in Shor's algorithm versus a preferred basis measurement in Simon's (information about the frequency spectrum helps determine the period frequency)
  - (Easy Case) Output is a superposition of m states

Shor's Algorithm Procedure – Easy Case

- Select an integer T, that reflects an acceptable failure rate based on any known aspects of the period. (E.g., for a failure tolerance of .000001)
- Repeat the following loop at most T times.
  1. Apply Shor's circuit
  2. Measure output of QFT and get cm
     a) [NOTE that output is zero everywhere except for values that are separated by multiples of the period "a" as a result of the Born Rule collapse of the output]
     b) Measurements produce eigenvalues of m (m is the number of times the period divides evenly into the function's domain m=N/a is the number of times "a" divides $N=2^n$)
     c) If one can determine then it is possible to compute "a"
     d) In the general case cm does not give discrete periodic spectrum but rather a distribution of values. Must invoke probability theory to prove that the data can be reduced to a value of m

Shor's Algorithm Procedure – Easy Case

3. Compute *m'* = gcd(*N*, *cm*), and set *a'* ≡ *N/m'*
4. Test *a'*: If *f*(1 + *a'*) = *f*(1) then *a'*=*a*, (success and break from loop)
5. Otherwise continue to the next pass of the loop

- If the above loop ended naturally (i.e., not from the "break") after *T* full passes the trial computation failed. If not, then have success in finding a
- NOTE: This procedure has a constant time looping with the circuit being computed gives an algorithm $\mathbb{O}(log^4(M))$. This is what gives the algorithm polynomial complexity.

(Easy Case) Computation Final Measurement Probabilities

1. Identify a special set of *a* elements $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ of certain measurement likelihood
2. Observe that each of $y_c = cm$ will be measured with equal likelihood
3. Prove that a random selection from [0, a-1] will be co-prime–to-"a" 50% of the time
4. Observe that a $y_c = cm$ associated with *c* coprime-to-*a* will be measured with probability ½
5. Measure *y* = *cm* associated with *c* coprime-to-*a* with arbitrarily high confidence in *constant time complexity**.

*The Constant Time Complexity Theorem for Looping Algorithms. Assume that A is a probabilistic, looping algorithm having size N. If we can show that the probability of success for a single loop pass is bounded away from zero (P(S) ≥ p > 0) with p independent of the size N, then A is a constant time algorithm

Step 1:
Identify a Special Set of "*a*" elements $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ of Certain Measurement Likelihood

- The post A register is left in state

$$\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|x_0 + ja>^n \xrightarrow{\text{QFT (N)}} \frac{1}{\sqrt{mN}}\sum_{y=0}^{N-1}\omega^{x_0 y}\left(\sum_{j=0}^{m-1}\omega^{jay}\right)|y>^n$$

- Note $\omega \equiv \omega_N$ was the Nth root of unity
- M is the number of times "*a*" divides (evenly) into *N* (m=N/a) implies

$$1 = \omega^N = \omega^{ma} = (\omega^a)^m$$

- This implies $\omega^a = \omega_m$ is the primitive m^{th} root of unity
- Using ω_m in place of ω^a produces

$$\sum_{j=0}^{m-1}\omega^{jay} = \sum_{j=0}^{m-1}\omega_m^{jy}$$

Gate Circuit for Shor's Algorithm

- Collaspsed B register will have $\left(\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|x_0 + ja>^n\right)|f(x_0)>^r$
- The superposition in the A register is determined by the measurement "f(x_0) of the collapsed B register

$$|\psi_{x0}>^n \equiv \left(\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|x_0 + ja>^n\right)$$

Effect on the A register is seen in the next sequence of 5 steps

Step 1:

Identify a Special Set of "*a*" elements $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ of Certain Measurement Likelihood

- This implies that the vast quantity of terms in the *QFT* output (the double sum) will disappear: only 1-in-*m* survives
- y=0 (mod m) or y=cm for some c= 0, 1 2, a-1
- This will define a special set of size "a" that will be certain to contain the measured value of y

$$\mathbb{C} = \{cm\}_{c=0}^{a-1}$$

Step 2: Observe that Each cm Will be Measured with Equal Likelihood

- The easy case assumes N = am
- Know that $\sqrt{m/N} = \sqrt{1/a}$

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja>^n \xrightarrow{\text{QFT (N)}} \frac{1}{\sqrt{a}} \sum_{c=0}^{a-1} \omega^{x_0 cm} |cm>^n$$

- Therefore each cm is measured with probability $1/a$

Step 3: Prove that a Random Selection From [0, a-1] Will Be Co-prime–to-"a" 50% of the Time

- State (without proof) that the probability of a randomly selected $c \in [0, a-1]$ being co-prime to "*a*" is ½

$$P(c \circ a) = \frac{\#\ co\text{–}primes\ to\ a \in [0,a-1]}{a}$$

$$= \frac{\#\ odds < 2^r}{2^r} = \frac{1}{2} \qquad (a = 2^r)$$

Step 4: Observe that a $y_c = cm$ Associated With "c" Coprime-to-"a" Will Be Measured With Probability ½*

- From step 1 shown that the likelihood of measuring one of the special $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ was 100%
- Steps 2 and 3 demonstrated that
  - Each of the cm will be measured with equal likelihood
  - The probability of selecting a number that is coprime to *a* at random from the numbers between 0 and a − 1 is (in this special "easy case") 1/2, i.e., a constant, independent of "a"

*Note: This step 4 seems overly elaborate. However in the general case these probabilities cannot be imply calculated

Step 4: Observe that a $y_c = cm$ Associated With *c* Coprime-to-"*a*" Will Be Measured With Probability ½*

- From step 1 shown that the likelihood of measuring one of the special $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ was 100%
- Steps 2 and 3 demonstrated that
  - Each of the *cm* will be measured with equal likelihood
  - The probability of selecting a number that is coprime to *a* at random from the numbers between 0 and *a* − 1 is (in this special *easy case*) 1/2, i.e., a constant, independent of *a*

**Note: This step 4 seems overly elaborate. However in the general case these probabilities cannot be imply calculated*

Step 4: Observe that a $y_c = cm$ Associated With *c* Coprime-to-"a" Will Be Measured With Probability ½

- Following claims are shown (without proof)
- Introduce notation
  - o $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$
  - o $\mathcal{B} = \{y_b | y_b = bm \in \mathbb{C}\ and\ b\ is\ coprime\ to\ a\}$
- $\mathrm{P}(\mathbb{C}) \equiv \mathrm{P}(\text{measure some y} \in \mathbb{C})$
- $\mathrm{P}(\mathcal{B}) \equiv \mathrm{P}(\text{measure some y} \in \mathcal{B})$
- $\mathrm{P}(\mathcal{B}|\mathbb{C}) \equiv \mathrm{P}(\text{measure some y} \in \mathcal{B} \text{ given that the measured y is known to be} \in \mathbb{C})$
- $\mathrm{P}(\mathcal{B}) = \mathrm{P}(\mathcal{B}|\mathbb{C})P(\mathbb{C})$ = (1/2)x1 = ½
- Therefore this also implies $\mathrm{P}(\neg c\ not\ coprime\ to\ a) = 1 - P(c\ is\ coprime\ to\ a)$ which is also 1/2

Step 5: Measure y=cm Associated With c Coprime-to-a With Arbitrarily High Confidence In Constant Time Complexity

- Algorithm loops through the circuit
- Produces a number c, each pass, and c must be co-prime to a to be a success (and a c which is not coprime to a is classified as a failure)
- Constant Time Complexity (CTC) Theorem for looping algorithms - theorem states that the algorithm will "succeed," yield a co-prime to a in constant time T
- Probability of at least one "mc" having c co-prime to a is $1 - \left(\frac{1}{2}\right)^{T}$

Summary Argument Justifying Why Finding $y_c \in B$ Solves Shor's Problem

- Steps 1 through 5 led to the argument that a measurement of $y_c \in B$ can be determined with near certainty after some predetermined number of measurements T
- After a measurement using Shor's circuit the following information is known and unknown

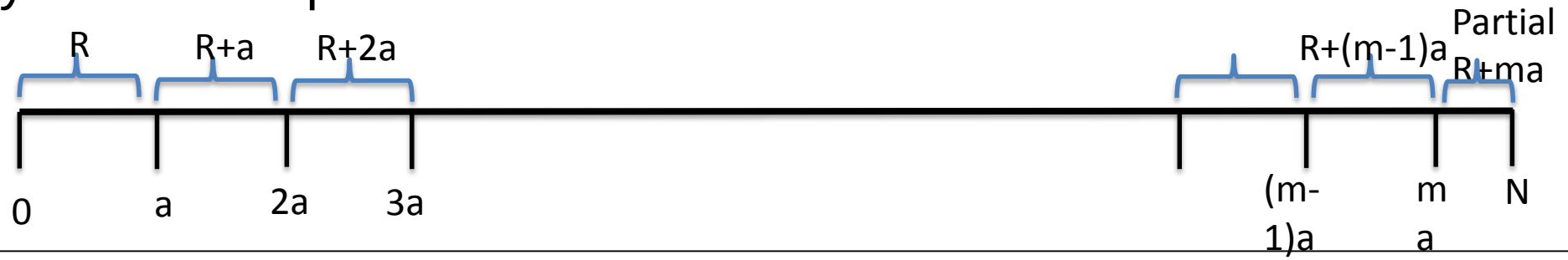
Known	Unknown
N	C
Cm	M
	a

Summary Argument Justifying Why Finding $y_c \in B$ Solves Shor's Problem (cont'd)

- Use GCD to produce m'=gcd (N, cm)
- If $\mathrm{cm} \in \mathcal{B}$ then c is co-prime to a and → m'=gcd (N, cm)=gcd (am, cm)=m
- Knowing m allows "a" to be calculated (a=N/m). [Note: this only happens if we measure $\mathrm{y} \in \mathcal{B}$ which is why finding such a *y* will give us the period and therefore solve Shor's problem
- Do not know a priori if $\mathrm{cm} \in \mathcal{B}$
- Try an m' that will calculate a'=N/m and test whether a'=a by asking whether f(x+a)=f(x) for any x and x=1
- "a" is the only number that will produce an equality here by the requirement of injective periodicity. If this is satisfied, problem is solved.
- If not, we try T-1 more times because, with .999999 probability, we'll succeed after T=20 times – no matter how big M (or N) is.

Shor's Algorithm – Second (General) Case

- No assumption of a|N but keep the property that f has injective periodicity
- take the integer quotient m=N/a there will be a remainder representing those "excess" integers that don't fit into a final, full period interval
- Cannot apply the roots of unity argument to the B register as was done to get that pure periodic function in the easy case
- Domain can still be partitioned into many disjoint cosets of size a, each of which provides a 1-to-1 sub-domain for f, but now there is a final coset which may not be complete:



Shor's Algorithm – Second (General) Case

- General case produces an output frequency spectrum where none of the values are ever zero
- Helpful that most values are small compared to a finite set that are large BUT these large values are not themselves multiples of the "m"

(General Case)
Computation Final Measurement Probabilities

1. Identify (without proof) a special set of "a" elements $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ of high measurement likelihood
2. Prove that the values in $\mathbb{C} = \{y_c = cm\}_{c=0}^{a-1}$ have high measurement likelihood
3. Associate $\{y_c\}_{c=0}^{a-1}$ with $\{c/a\}_{c=0}^{a-1}$
4. Describe an O($\log^3$ N) algorithm that will produce c/a from y_c [Use the mathematics of differential calculus and continued fractions to produce a sequence of (reduced) fractions, $\{n_k/d_k\}$ that approach (get closer and closer to) *x*. Want these other fractional approximations because among them we'll find one, n/d , which is identical to our sought-after c/a
5. Observe that y_c associated with c coprime to a will be measured in constant time

References

- *Shor, P.W. "Algorithms for quantum computation: discrete logarithms and factoring". Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press. doi:10.1109/sfcs.1994.365700. ISBN 0818665807*
- Lanyon, B. P.; Weinhold, T. J.; Langford, N. K.; Barbieri, M.; James, D. F. V.; Gilchrist, A. & White, A. G. (2007), "Experimental Demonstration of a Compiled Version of Shor's Algorithm with Quantum Entanglement" (PDF), *Physical Review Letters*, **99** (25): 250505, arXiv:0705.1398, Bibcode:2007PhRvL..99y0505L, doi:10.1103/PhysRevLett.99.250505, PMID 18233509
- *Shor, Peter W. (1997), "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM J. Comput., **26** (5): 1484–1509, arXiv:quant-ph/9508027v2, Bibcode:1999SIAMR..41..303S, doi:10.1137/S0036144598347011*. Revised version of the original paper by Peter Shor ("28 pages, LaTeX. This is an expanded version of a paper that appeared in the Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, Nov. 20--22, 1994. Minor revisions made January, 1996").
- Quantum Computing and Shor's Algorithm, Matthew Hayward's Quantum Algorithms Page, 2005-02-17, imsa.edu, LaTeX2HTML version of the original LaTeX document, also available as PDF or postscript document.
- *Lomonaco, Jr (2000). "Shor's Quantum Factoring Algorithm". arXiv:quant-ph/0010034*. This paper is a written version of a one-hour lecture given on Peter Shor's quantum factoring algorithm. 22 pages.

Questions