Algorithms

Grover's search algorithm

Shor's factoring algorithm

Lecture 8

Grover's search algorithm

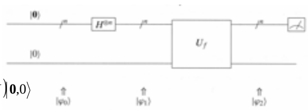
- Search element in an unordered array of size m in $\sqrt{m}$ time instead of $m/2$ time on average.
- In terms of functions, given a function $f: \{0,1\}^n \rightarrow \{0,1\}$, where there exists exactly one binary string $\mathbf{x_0}$, such that:

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{x_0} \\ 0, & \text{if } \mathbf{x} \neq \mathbf{x_0} \end{cases}$$

- Find $\mathbf{x_0}$. Classically, in the worst case, we have to evaluate all 2^n binary strings. Grover's algorithm demands only $\sqrt{2^n} = 2^{n/2}$ evaluations.

First try

- Put |**x**> into a superposition of all possible strings and then evaluate U_f



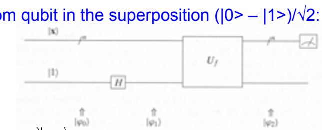
- In terms of matrices $U_f\left(H^{\otimes n} \otimes I\right)|\mathbf{0},0\rangle$
- The states are

$$|\varphi_0\rangle = |\mathbf{0},0\rangle, \quad |\varphi_1\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x}\rangle}{\sqrt{2^n}}\right]|0\rangle, \quad |\varphi_2\rangle = \frac{\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x}, f(\mathbf{x})\rangle}{\sqrt{2^n}}$$

- Measuring the top qubits will, with equal probability, give one of the 2^n binary strings. Measuring the bottom qubit will give |0> with probability $2^{n}-1/2^n$, and |1> with probability $1/2^n$. If one is lucky enough to measure |1>, the top qubits will have the correct answer, because of the entanglement. However, probably not so lucky.

First trick: phase inversion

- Change the phase of the desired state.
- Take U_f and place the bottom qubit in the superposition $(|0> – |1>)/\sqrt{2}$:



- In terms of matrices: $U_f\left(I_n \otimes H\right)|\mathbf{x},1\rangle$
- The states are:

$$|\varphi_0\rangle = |\mathbf{x},1\rangle,$$

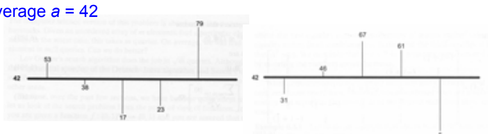
$$|\varphi_1\rangle = |\mathbf{x}\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] = \left[\frac{|\mathbf{x},0\rangle - |\mathbf{x},1\rangle}{\sqrt{2}}\right],$$

$$|\varphi_2\rangle = |\mathbf{x}\rangle\left[\frac{|f(\mathbf{x})\oplus 0\rangle - |f(\mathbf{x})\oplus 1\rangle}{\sqrt{2}}\right] = |\mathbf{x}\rangle\left[\frac{|f(\mathbf{x})\rangle - |\overline{f(\mathbf{x})}\rangle}{\sqrt{2}}\right] = \begin{cases} -1|\mathbf{x}\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], & \text{if } \mathbf{x} = \mathbf{x_0} \\ +1|\mathbf{x}\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], & \text{if } \mathbf{x} \neq \mathbf{x_0} \end{cases}$$

Example: $[½,½,½,½]^T$ and f chooses string "10", then after phase inversion: $[½,½,-½,½]^T$. Measuring |**x**> does not give any information $|½|^2= |-½|^2=¼$.

Second trick: inversion about the mean or inversion about the average

- Boosting the separation of the phases.
- Explain with an example:
  - 53, 38, 17, 23, and 79
  - Average a = 42



  - Sum of the lengths of lines above the average is the same as the sum of lines below.
  - Invert each element around the average: $v' = a + (a-v)$; example [53, 38, 17, 23, 79] → [31, 46, 67, 61, 5]
  - In terms of matrices: $V'=(-I +2A)V$, with $A[i,j]=1/n$.

Inversion about the mean or average (cont'd)

- In general: n qubits, 2^n possible states, where a state is 2^n vector. Then 2^n-by-2^n matrix

$$A = \begin{bmatrix} \frac{1}{2^n} & \frac{1}{2^n} & \cdots & \frac{1}{2^n} \\ \frac{1}{2^n} & \frac{1}{2^n} & \cdots & \frac{1}{2^n} \\ \vdots & \vdots & \ddots & \\ \frac{1}{2^n} & \frac{1}{2^n} & \cdots & \frac{1}{2^n} \end{bmatrix}$$

- Multiply any state by a will give state where each amplitude will be the average of all amplitudes.
- The 2^n-by-2^n matrix

$$-I + 2A = \begin{bmatrix} -1+\frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \frac{2}{2^n} & -1+\frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \\ \frac{2}{2^n} & \frac{2}{2^n} & \cdots & -1+\frac{2}{2^n} \end{bmatrix}$$

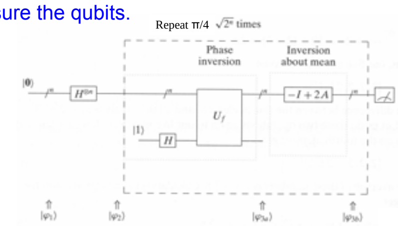
- Multiply a state by $-I+2A$ will invert amplitudes about the mean.

Phase inversion **and** inversion about the mean

- Combination is a powerful operation that separates the amplitude of the desired state from those of all other states.
- Example that demonstrates the combined techniques:
  - Vector $[10,10,10,10]^T$
  - Phase inversion to the fourth element: $[10,10,10,–10]^T$
  - Inversion about the mean (=6; -v+2a=2 or 22): $[2,2,2,22,2]^T$
  - Another phase inversion: $[2,2,2,–22,2]^T$
  - Inversion about the mean (=–2.8, -v+2a=–7.6 or 16.4): $[–7.6,–7.6, –7.6,16.4,–7.6]^T$
  - Another time? No, $\pi/4\sqrt{n}$ times, otherwise the numbers will be “overcooked”.

Grover’s algorithm

1) Start with a state |**0**>
2) Apply $H^{\otimes n}$
3) Repeat $\pi/4\sqrt{2^n}$ times:
   a) Apply the phase inversion operators: $U_f(I \otimes H)$
   b) Apply the inversion about the mean operation: $-I + 2A$
4) Measure the qubits.



Example Grover’s algorithm

- Let *f* be a function that picks out the string “101”.
- The states: $|\varphi_1\rangle = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$,

$$|\varphi_2\rangle = \left[\frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\right]^T,$$

$$|\varphi_{3a}\rangle = \left[\frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ -\frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\ \ \frac{1}{\sqrt{8}}\right]^T$$

- The average is: $a = \frac{7 * \frac{1}{\sqrt{8}} - \frac{1}{\sqrt{8}}}{8} = \frac{\frac{6}{\sqrt{8}}}{8} = \frac{3}{4\sqrt{8}}$
- Calculating the inversion about the mean:

$$-v + 2a = -\frac{1}{\sqrt{8}} + \left(2 \times \frac{3}{4\sqrt{8}}\right) = \frac{1}{2\sqrt{8}}$$

and

$$-v + 2a = \frac{1}{\sqrt{8}} + \left(2 \times \frac{3}{4\sqrt{8}}\right) = \frac{5}{2\sqrt{8}}$$

$$|\varphi_{3b}\rangle = \left[\frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{5\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\right]^T$$

Example Grover’s algorithm (cont’d)

- Another phase inversion:

$$|\varphi_{3a}\rangle = \left[\frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ -\frac{1}{5\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\ \ \frac{1}{2\sqrt{8}}\right]^T$$

- The average is: $a = \frac{7 * \frac{1}{2\sqrt{8}} - \frac{5}{2\sqrt{8}}}{8} = \frac{1}{8\sqrt{8}}$
- Calculating the inversion about the mean:

$$-v + 2a = -\frac{1}{2\sqrt{8}} + \left(2 \times \frac{1}{8\sqrt{8}}\right) = -\frac{1}{4\sqrt{8}}$$

and

$$-v + 2a = \frac{5}{2\sqrt{8}} + \left(2 \times \frac{1}{8\sqrt{8}}\right) = \frac{11}{4\sqrt{8}}$$

$$|\varphi_{3b}\rangle = \left[\frac{-1}{4\sqrt{8}}\ \ \frac{-1}{4\sqrt{8}}\ \ \frac{-1}{4\sqrt{8}}\ \ \frac{-1}{4\sqrt{8}}\ \ \frac{-1}{4\sqrt{8}}\ \ \frac{11}{4\sqrt{8}}\ \ \frac{-1}{4\sqrt{8}}\ \ \frac{-1}{4\sqrt{8}}\right]^T$$

- $11/4\sqrt{8}=0.97$ and $-1/4\sqrt{8}=-0.088$, and squaring these numbers gives us the probability of measuring the corresponding states. Most likely we will measure:

$$|\varphi_4\rangle = [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0]^T$$

OK!

Generalizations Grover’s algorithm

- Search an unordered array of size *m* in *m* time → $\sqrt{m}$ time: quadratic speedup.
- What if there is more than one hit? Assume t objects: Grover’s algorithm still works, but one must go through the loop $\pi/4\sqrt{(2^n/t)}$ times.
- Many other types of generalizations and assorted changes.

Shor’s factoring algorithm

- Factoring integers important: security
- “Hard” on classical computers
- Peter Shor: in polynomial time on quantum computers
- Based on the fact that the factoring problem can be reduced to finding the period of a certain function (see Simon’s algorithm)
- In practice *N* will be a large number
- Assume *N* is not a prime number. However, there exists a deterministic, polynomial algorithm that determine if *N* is prime.

Modular exponentiation

- Modular arithmetic: for a positive integer N and any integer a, we write a mod N for the remainder (or residue) of the quotient a/N, e.g. 99 mod 15 = 9.
- $a \equiv a'$ mod N, if and only if (a mod N) = (a' mod N) or equivalent, if N is a divisor of $a-a'$, i.e. $N|(a-a')$.
- Start of the algorithm: choose randomly an interger a that is less than N, but does not have a nontrivial factor in common with N. This can be tested by Euclid's algorithm to calculate GCD(a,N):
  - GCD ≠ 1: found a factor of N and done;
  - GCD = 1: a is called co-prime to N and we can use it.
- We need to find the powers of a mod N, that is, a^0 mod N, a^1 mod N, a^2 mod N, a^3 mod N, …
- In other words, we need to find the values of the function

$$f_{a,N}(x) = a^x \bmod N$$

Examples $f_{a,N}(x) = a^x \bmod N$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	…
$f_{2,15}(x)$	1	2	4	8	1	2	4	8	1	2	4	8	1	…

x	0	1	2	3	4	5	6	7	8	9	10	11	12	…
$f_{4,15}(x)$	1	4	1	4	1	4	1	4	1	4	1	4	1	…

x	0	1	2	3	4	5	6	7	8	9	10	11	12	…
$f_{13,15}(x)$	1	13	4	7	1	13	4	7	1	13	4	7	1	…

In book: N = 371

Not the values, but the period

- Not the values of $f_{a,N}(x) = a^x \bmod N$, but the period of this function, i.e., we need to find the smallest $r > 0$ such that $f_{a,N}(r) = a^r \bmod N = 1$
- Theorem in number theory that for any co-prime $a \le N$, the function $f_{a,N}$ will output a 1 for some $r < N$. After it hits 1, e.g.,

$$\text{if} \quad f_{a,N}(r) = 1$$
$$\text{then} \quad f_{a,N}(r+1) = f_{a,N}(1)$$
$$\text{and in general} \quad f_{a,N}(r+s) = f_{a,N}(s)$$

Examples

x	0	1	2	3	4	5	6	7	8	9	10	11	12	…
$f_{2,15}(x)$	1	2	4	8	1	2	4	8	1	2	4	8	1	…

period 4

x	0	1	2	3	4	5	6	7	8	9	10	11	12	…
$f_{4,15}(x)$	1	4	1	4	1	4	1	4	1	4	1	4	1	…

period 2

x	0	1	2	3	4	5	6	7	8	9	10	11	12	…
$f_{13,15}(x)$	1	13	4	7	1	13	4	7	1	13	4	7	1	…

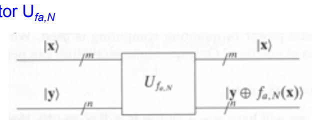
period 4

Quantum part of the algorithm

- For small numbers it is easy to determine the periods of these functions. But what if N is hundred digits long? This will be beyond the ability of any conventional computer: we need to calculate $f_{a,N}$ for ***all*** needed x: superposition.
- First we have to show that there is a quantum circuit that can implement $f_{a,N}$ (later).
- The output of this function will always be less than N, so we need $n = \log_2 N$ output qubits.
- We will need to evaluate $f_{a,N}$ for at least N^2 values of x, so we will need at least $m = \log_2 N^2 = 2 \log_2 N = 2n$ input qubits.

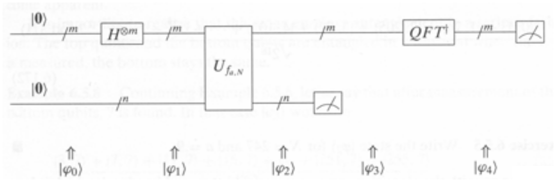
Quantum circuit for $U_{fa,N}$

- Operator $U_{fa,N}$



- where $|\mathbf{x},\mathbf{y}\rangle \mapsto |\mathbf{x},\mathbf{y} \oplus f_{a,N}(\mathbf{x})\rangle = |\mathbf{x},\mathbf{y} \oplus a^x \bmod N\rangle$
- How is it formed? Later…

Quantum circuit



$$(Measure \otimes I)(QFT^{\dagger} \otimes I)(I \otimes Measure)U_{f_{a,N}}(H^{\otimes m} \otimes I)|\mathbf{0}_m, \mathbf{0}_n\rangle$$

States $|\varphi_0\rangle$, $|\varphi_1\rangle$, and $|\varphi_2\rangle$

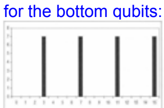
- We start at $|\varphi_0\rangle = |\mathbf{0}_m, \mathbf{0}_n\rangle$
- Then we place the input in an equally weighted superposition of all possible inputs
  $$|\varphi_1\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^m} |\mathbf{x}, \mathbf{0}_n\rangle}{\sqrt{2^m}}\right]$$
- Evaluation of f on all these possibilities gives us
  $$|\varphi_2\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^m} |\mathbf{x}, f_{a,N}(\mathbf{x})\rangle}{\sqrt{2^m}}\right] = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^m} |\mathbf{x}, a^{\mathbf{x}} \bmod N\rangle}{\sqrt{2^m}}\right]$$
- These outputs are periodic, e.g., for N = 15, we have n = 4 and m = 8. For a = 13 we have
  $$|\varphi_2\rangle = \frac{|0,1\rangle + |1,13\rangle + |2,4\rangle + |3,7\rangle + |4,1\rangle + \cdots |254,4\rangle + |255,7\rangle}{\sqrt{256}}$$

Measure $|\varphi_2\rangle$

- Measuring the bottom qubits gives us $a^{\bar{\mathbf{x}}} \bmod N$ for some $\bar{\mathbf{x}}$
- However, by periodicity we also have
  $$a^{\bar{\mathbf{x}}} \equiv a^{\bar{\mathbf{x}}+r} \bmod N \quad \text{and} \quad a^{\bar{\mathbf{x}}} \equiv a^{\bar{\mathbf{x}}+2r} \bmod N,$$
  $$\text{and in fact, for any } s \in Z: \; a^{\bar{\mathbf{x}}} \equiv a^{\bar{\mathbf{x}}+sr} \bmod N$$
- How many of the 2^m superpositions **x** have $a^{\bar{\mathbf{x}}} \bmod N$ as output?
- Answer: $\left\lfloor \frac{2^m}{r} \right\rfloor$

State $|\varphi_3\rangle$

- So
  $$|\varphi_3\rangle = \left[\frac{\sum_{a^{\mathbf{x}} \equiv a^{\bar{\mathbf{x}}} \bmod N} |\mathbf{x}, a^{\bar{\mathbf{x}}} \bmod N\rangle}{\sqrt{\left\lfloor \frac{2^m}{r} \right\rfloor}}\right]$$
- We might also write this as
  $$|\varphi_3\rangle = \left[\frac{\sum_{j=0}^{2^m/r-1} |t_0 + jr, a^{\bar{\mathbf{x}}} \bmod N\rangle}{\sqrt{\left\lfloor \frac{2^m}{r} \right\rfloor}}\right]$$
- Here t_0 is the first time that the measured value occurs. It is called the offset of the period.
- Example (cont'd), let us say that we measure 7 for the bottom qubits:
  $$|\varphi_3\rangle = \frac{|3,7\rangle + |7,7\rangle + |11,7\rangle + |15,7\rangle + \cdots |251,7\rangle + |255,7\rangle}{\sqrt{\left\lfloor \frac{256}{4} \right\rfloor}}$$



Vandermonde matrix

- Evaluating polynomials: $P(x) = a_0 + a_1x^1 + a_2x^2 + a_3x^3 + \cdots + a_{n-1}x^{n-1}$
- This polynomial can be represented by a column vector $[a_0, a_1, a_2, \cdots, a_{n-1}]^T$
- Suppose we want to evaluate this polynomial at numbers $x_0, x_1, x_2, \cdots, x_{n-1}$
- This can be achieved by

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^j & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^j & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^j & \cdots & x_2^{n-1} \\ \vdots & & & & \vdots & & \vdots \\ 1 & x_k & x_k^2 & \cdots & x_k^j & \cdots & x_k^{n-1} \\ \vdots & & & & \vdots & & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^j & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} P(x_0) \\ P(x_1) \\ P(x_2) \\ \vdots \\ P(x_k) \\ \vdots \\ P(x_{n-1}) \end{bmatrix}.$$

Every row is a geometric series, matrix is called the **Vandermonde matrix**, denoted by $V(x_0, x_1, x_2, \cdots, x_{n-1})$

Vandermonde matrix (cont'd)

- Elements changed to "powers of one of the M^{th} roots of unity ω^1_M" (chapter 1)
- $M = 2^m$ is fixed, so ω_M is simply ω. We obtain the M-by-M Vandermonde matrix $\mathbf{V}(\omega^0, \omega^1, \omega^2, \ldots, \omega^{M-1})$
- To evaluate $P(x)$ at the powers of one of the M^{th} roots of unity

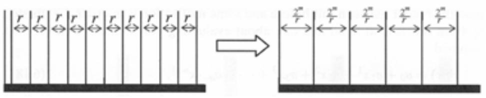
$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \cdots & \omega^j & \cdots & \omega^{M-1} \\ 1 & \omega^2 & \omega^{2\times 2} & \cdots & \omega^{2j} & \cdots & \omega^{2(M-1)} \\ \vdots & & & & \vdots & & \vdots \\ 1 & \omega^k & \omega^{k2} & \cdots & \omega^{kj} & \cdots & \omega^{k(M-1)} \\ \vdots & & & & \vdots & & \vdots \\ 1 & \omega^{M-1} & \omega^{(M-1)2} & \cdots & \omega^{(M-1)j} & \cdots & \omega^{(M-1)(M-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_k \\ \vdots \\ a_{M-1} \end{bmatrix} = \begin{bmatrix} P(\omega^0) \\ P(\omega^1) \\ P(\omega^2) \\ \vdots \\ P(\omega^k) \\ \vdots \\ P(\omega^{M-1}) \end{bmatrix}.$$

Discrete Fourier transform

- Definition of discrete Fourier transform (DFT)

$$DFT = \frac{1}{\sqrt{M}} \mathbf{V}(\omega^0, \omega^1, \omega^2, \ldots, \omega^{M-1})$$

- Formally, DFT is defined as $DFT = \frac{1}{\sqrt{M}} \omega^{jk}$
- Two tasks:
  - It modifies the period from r to $2^m/r$
  - It eliminates the offset.



Quantum Fourier transform

- Denoted by *QFT*.
- Same operation, but more suitable for quantum computers.
- This quantum version is very fast and made of “small” unitary operators that are easy to implement.

Measure the top qubits

- Assumption that r evenly divides into 2^m (not in Shor’s actual algorithm: finding period for any r). So we measure the top qubit and we will find some multiple of $2^m/r$. We will measure

$$x = \frac{\lambda 2^m}{r} \quad \text{for some whole number } \lambda$$

- We know 2^m and after measurement also x, so we get

$$\frac{x}{2^m} = \frac{\lambda 2^m}{r 2^m} = \frac{\lambda}{r}$$

- Reduce this number to an irreducible fraction and take the denominator to be the period r. If we don’t make the simplifying assumption, given above: perform this process several times.

From the Period to the Factors

- Assumption the period r is an even number; if not, choose another a.
- So $a^r \equiv 1 \bmod N$ and we may subtract 1 from both sides to get $a^r - 1 \equiv 0 \bmod N$, or equivalently $N \mid (a^r - 1)$.
- Or $N \mid (\sqrt{a^r} + 1)(\sqrt{a^r} - 1)$ or $N \mid (a^{r/2} + 1)(a^{r/2} - 1)$, remember r is even.
- So any factor of N is also a factor of either $(a^{r/2} + 1)$ or $(a^{r/2} - 1)$ or both.
- Either way, a factor for N can be found by looking at GCD($(a^{r/2} + 1)$, N) and GCD($(a^{r/2} - 1)$, N), which can be done by the classical Euclidean algorithm.
- One problem: be sure that $a^{r/2} \neq -1 \bmod N$. Solution: start over again.
- Example: period $f_{2,15}$ is 4. So GCD(5,15)=5 and GCD(3,15)=3.

Shor’s algorithm

- Putting all pieces together, see p217 of the book.
- Complexity of this algorithm? $O(n^2 \log n \log \log n)$, where n is the number of bits to represent the number N.
- The best classical algorithms demand

$$O(e^{c n^{1/3} \log^{2/3} n}) \quad \text{where } c \text{ is some constant}$$

- This is exponential in terms of n.
- Implementation of $U_{fa,N}$: see p217-218.

Final remark

“Even if a real implementation of large-scale quantum computers is years away, the design and study of quantum algorithms is something that is ongoing and is an exciting field of interest.”

Reading

- This lecture: Ch 6.4-6.5
- Next lecture: Ch 9