

CS290A/Phys250, Spring 2007:

Quantum Information and Quantum Computation

Wim van Dam

Harold Frank Hall, Room 5109

vandam@cs

http://www.cs.ucsb.edu/~vandam/teaching/s07_CS290/

Midterm

Wednesday, May 16, 10–10:50 am, PHELPS 1401

Material: Everything covered in Week 1–6, including Simon’s algorithm. The quantum Fourier transform is not included. In the textbook [KLM] this corresponds to Chapters 1–6, excluding parts that we did not discuss such as Bloch ball, the Kraus representation, et c. When in doubt, consult the slides. The slides of Weeks 1–6 are posted as one pdf and ppt file. Let me know if you manage to convert them to a B&W version. You are allowed to bring with you: personal notes, the textbook and print outs of material on CS290 site (slides, exercises, et c.)

Projects Thus Far

Greg Dyer: Implementations using quantum optics

Sean Ford: "Quantum finite automata"

Mark Howard: "Measurement Based Quantum Computation"

Tom Howard: "The Hidden Subgroup Problem"

Hyochul Kim: Implementations using optical lattices

Erik Lucero: "Quantum Computing: Where are we at and what can we do for real?"

Alex Roizinov: Quantum algorithms for playing chess

Simon Rubinstein-Salzedo: "Quantum algorithms in algebraic number theory"

Maria Matthews, Christopher Eberz, Puneeth Kalavase: ?

Back to the (Quantum) Fourier Transform'

Properties of Four_N

The definition: $\text{Four}_N : |x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \cdot xy / N} |y\rangle$

Hence: $\text{Four}_N = \frac{1}{\sqrt{N}} \sum_{x,y=0}^{N-1} e^{2\pi i \cdot xy / N} |y\rangle\langle x|$

and: $\langle y | \text{Four}_N | x \rangle = \frac{1}{\sqrt{N}} e^{2\pi i \cdot xy / N}$

The inverse: $\text{Four}_N^{-1} : |y\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} e^{-2\pi i \cdot yz / N} |z\rangle$

Know your phase summations...

Example

What happens if you apply Four_N twice to $|0\rangle$?

$$\begin{aligned} |0\rangle &\mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle \\ &\mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{z=0}^{N-1} e^{2\pi i y z / N} |z\rangle \\ &= \frac{1}{N} \sum_{z=0}^{N-1} \left(\sum_{y=0}^{N-1} e^{2\pi i y z / N} \right) |z\rangle \end{aligned}$$

The (summation) is 0 if $z \neq 0$ and N if $z=0$.
Hence the outcome state is $|0\rangle$.

Question: What happens if we apply Four_N twice to a basis state $|x\rangle$ with $0 < x < N$?

More About Fourier

Traditionally, Fourier transforms are used to detect periodic signals (depending on their frequencies).

In quantum computing we will use the QFT to determine the periodicity of a function F .

Already interesting by itself, this periodicity finding subroutine can be used to factorize numbers and calculate discrete logarithms over \mathbb{Z}/N .

See later Handouts for more technical details and a description of efficient circuits to implement Four_N .

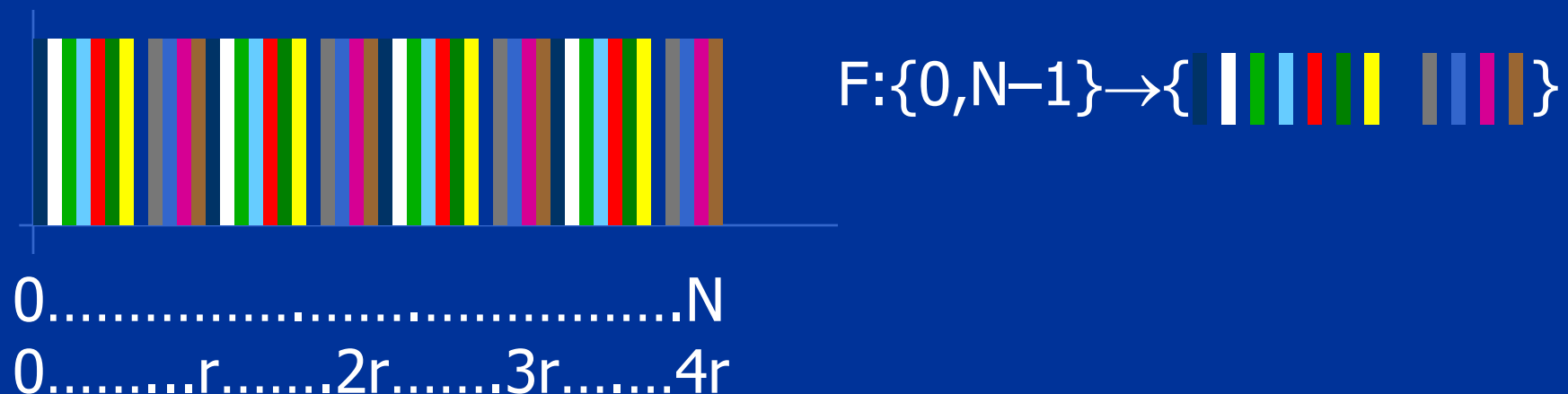
Periodicity Problem

Consider function $F:\{0,\dots,N-1\} \rightarrow S$

Assuming: F has period r
 F is bijective on its period

$$F(x) = F(y) \text{ if and only if } x = y \pmod r$$

Task: Determine r efficiently ($\text{poly}(\log N)$)



Periodicity Algorithm (1)

Start with a uniform superposition of x values:

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

Calculate the periodic function F for these values:

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x, F(x)\rangle \approx \frac{1}{\sqrt{N}} \sum_{y=0}^{r-1} \left(\sum_{t=0}^{N_r-1} |tr + y\rangle \right) \otimes |F(y)\rangle$$

“Measure” the rightmost register;
assume outcome “ $F(c)$ ” with $0 \leq c < r \dots$

Periodicity Algorithm (2)

... this yields the superposition for the left register:

$$\sqrt{\frac{r}{N}} \sum_{t=0}^{N/r-1} |tr + c\rangle$$

Apply the Fourier transform over Z/N , giving:

$$\frac{\sqrt{r}}{N} \sum_{t=0}^{N/r-1} \sum_{j=0}^{N-1} \zeta_N^{j(tr+c)} |j\rangle = \frac{\sqrt{r}}{N} \sum_{j=0}^{N-1} \zeta_N^{jc} \left(\sum_{t=0}^{N/r-1} \zeta_N^{jtr} \right) |j\rangle$$

with $\zeta_N = \exp(2\pi i/N)$.

If j multiple of N/r , then
constructive interference

If j *not* a multiple of N/r , then
destructive interference

Periodicity Algorithm (3)

Calculating the j-dependent interference:

$$\sum_{t=0}^{N/r-1} \zeta_N^{jtr} \approx \frac{N}{r} \quad \text{if } jr \text{ is a multiple of } N$$

$$\sum_{t=0}^{N/r-1} \zeta_N^{jtr} \approx \frac{\zeta_N^{jN} - 1}{\zeta_N^{jr} - 1} = 0 \quad \text{if } jr \text{ is not a multiple of } N$$

Hence we have the output state:

$$\frac{\sqrt{r}}{N} \sum_{j=0}^{N-1} \zeta_N^{jc} \left(\sum_{t=0}^{N/r-1} \zeta_N^{jtr} \right) |j\rangle \approx \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \zeta_N^{ckN/r} |k \cdot N/r\rangle$$

Periodicity Algorithm (4)

With very high probability we will measure a multiple of N/r , where r is the period of the function.

By repeating the procedure several times, we obtain enough information to determine N/r and hence r .

(This is not entirely trivial and requires the usage of the “continued fractions method”, but it can be done.)

Being able to find the (hidden) period of a function allows us to solve factoring, discrete logarithms and other (presumed) hard problems.

Use of Periodicity Finding?

The quantum algorithm for periodicity finding works for a “black box function” F as long as it has the right properties (F is periodic, and unique within its period).

You can prove that any classical algorithm requires $\Theta(\text{poly}(r))$ time steps to solve the same problem.

We want to use this quantum subroutine to solve **natural problems** that are defined without reference to a black box function. That is: we want to look at explicit functions F .

Bad Example: The function $F(x) = x \text{ MOD } r$ has the right characteristics, but is easy classically.

A Hard Periodic Function

Take a (large) integer N , and an element $x \in \{0, 1, \dots, N-1\}$ with $\gcd(N, x) = 1$ (such that x has an inverse mod N).

The function $F: t \mapsto x^t \bmod N$ will be 'properly periodic'.

As $F(0) = 1, F(1) = x, \dots$; $F(r) = F(0) = 1$ shows that $x^r = 1 \bmod N$.

Example: Try $N = 15, x = 2$.

With the quantum algorithm for period finding, we can efficiently solve the problem: "Given N and x , determine the smallest $r > 1$ such that $x^r = 1 \bmod N$ ".

Classically, this appears to be a hard problem.

Side Comments

For the quantum algorithm to work, we have to efficiently implement the function $F:t \mapsto x^t \bmod N$.

This can be done by the “repeated squaring trick”:
We can calculate $x \mapsto x^2 \mapsto x^4 \mapsto x^8 \bmod N \dots$ fast;
hence we can calculate $x^t \bmod N$ in time $\text{poly}(\log t, \log N)$.

Initially, we do not know the period r of $F:\mathbf{N} \rightarrow \{0, \dots, N-1\}$,
so we have to ‘guess’ how many $F(0), F(1), F(2), \dots$
we want to evaluate in the superposition.

You can show that $F(0), \dots, F(\approx N)$ is sufficient.

(Period finding is a robust quantum algorithm: small mistakes in the function F do not matter.)

Factorizing by Period Finding

How to find a non-trivial factor of an integer N ?

Sketch of the algorithm using Period Finding mod N :

- Pick random $x < N$ with $\gcd(x, N) = 1$
- Determine smallest r such that: $x^r = 1 \pmod N$
- If r is even (*), note that $(x^{r/2} - 1)(x^{r/2} + 1) = 0 \pmod N$

In that case it is possible that $x^{r/2} - 1$ or $x^{r/2} + 1$ is a non-trivial factor of N (*).

(*) All this succeeds with high enough probability.

Repeat if necessary.

Again, check by yourself what happens for $N = 15$.