Quantum computation: lecture 8

Plan for today:

- end of order finding algorithm:
  - convergents
  - circuit for U_f, $f(x) = a^x \pmod N$
- Shor's factoring algorithm

Convergents (recall: a, N large integers (n digits each), $M = 2^m \geq N^2$)

Remember at which point we arrived last time:

- Our aim is to identify the smallest value of $r \geq 1$ s.t. $a^r \pmod N = 1$
- With probability $\geq \frac{2}{5}$, the quantum algorithm described in the previous lecture outputs a state $y_0 \in I = \bigcup_{k=0}^{r-1} I_k$ with $I_k = \left[k\frac{M}{r} - \frac{1}{2}, k\frac{M}{r} + \frac{1}{2} \right]$

What can we deduce in the case where $y_0 \in I$?

- first, that $\left|\frac{y_0}{M} - \frac{k}{r}\right| \leq \frac{1}{2M}$ for some $k \geq 1$
- but remember also that we have chosen $M \geq N^2 > r^2$, so this implies $\left|\frac{y_0}{M} - \frac{k}{r}\right| < \frac{1}{2r^2}$
- in order to understand what to do with this, we need the notion of <u>convergents</u>.

Convergents and continued fractions

Pick a real number $x = \frac{73}{31}$: ≈ 2.355

approx. value:

- $\frac{73}{31} = 2 + \frac{11}{31}$ — 2
- $\frac{73}{31} = 2 + \frac{1}{\frac{31}{11}} = 2 + \frac{1}{2 + \frac{9}{11}}$ — $2 + \frac{1}{2} = 2.5$
- $\frac{73}{31} = 2 + \frac{1}{2 + \frac{1}{\frac{11}{9}}} = 2 + \frac{1}{2 + \frac{1}{1 + \frac{2}{9}}}$ — $2 + \frac{1}{2 + \frac{1}{1}} = 2 + \frac{1}{3} = 2.\overline{3}$
- $\frac{73}{31} = 2 + \frac{1}{2 + \frac{1}{1 + \frac{1}{\frac{2}{2}}}} = 2 + \frac{1}{2 + \frac{1}{1 + \frac{1}{4 + \frac{1}{2}}}}$ STOP — $2 + \frac{1}{2 + \frac{1}{1 + \frac{1}{4}}} = 2 + \frac{5}{14} \approx 2.357$

Computing the convergents of a real number x is therefore a way to obtain successive approximations of x; moreover, the convergence is <u>fast</u>: one digit is essentially gained at each step.

(Note that if the initial x is irrational, then this procedure does not stop.)

Legendre's lemma: Let $x \in \mathbb{R}$: (and assume gcd(k,r)=1)

if $|x - \frac{k}{r}| < \frac{1}{2r^2}$, then $\frac{k}{r}$ is a convergent of x.

So the algorithm for finding r is as follows:

- if $y_0 \in I$, then $|\frac{y_0}{M} - \frac{k}{r}| < \frac{1}{2r^2}$ for some k
- by legendre's lemma, $\frac{k}{r}$ is a convergent of $\frac{y_0}{M}$
- compute all the convergents of $x = \frac{y_0}{M}$, along with their denominators (say d_i): if one d_i is such that $a^{d_i} \pmod N = 1$, then $r = d_i$

Finally, please observe that the above algorithm will probably fail to find the value of r if it turns out that $y_0 \notin I$. In this case, simply relaunch the whole thing: since $\text{prob}(y_0 \in I) \geq \frac{2}{5}$, success is around the corner.

Circuit for U_f, $f(x) = a^x \pmod N$

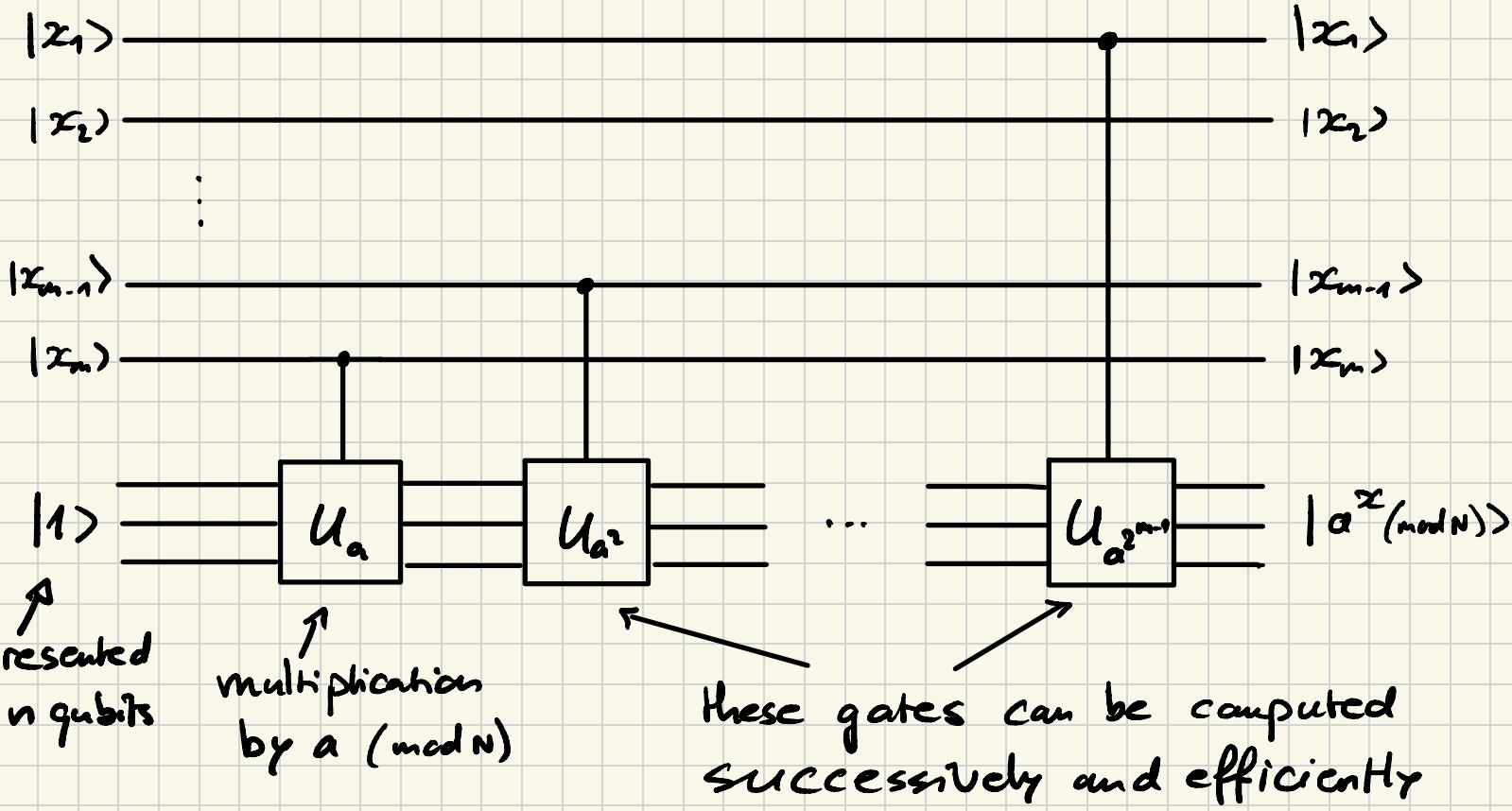
Recall that x is represented on m bits, with $m = \log_2 M$ and $M \geq N^2$.

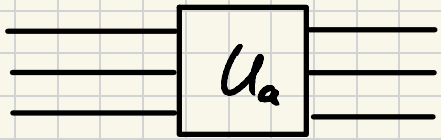
Also, recall that $x = \sum_{k=1}^{m} x_k 2^{m-k}$, so that

$$f(x) = \left(\prod_{k=1}^{m} a^{x_k 2^{m-k}} \right) \pmod N$$

$$= \prod_{k=1}^{m} \left(\underbrace{a^{x_k 2^{m-k}} \pmod N}_{\text{can be represented on } n \text{ bits}} \right) \pmod N$$

This suggests the following circuit for U_f:



Note about the gate 

(as well as about all the other gates $U_{a^2}, U_{a^4}, \ldots$)

The input (say $|y\rangle$) is represented on n qubits

If $N \neq 2^n$ (which is the case in general), then

$$U_a|y\rangle = \begin{cases} |ay \pmod N\rangle & \text{for } 0 \leq y \leq N-1 \\ |y\rangle & \text{for } N \leq y \leq 2^n - 1 \end{cases}$$

(= permutation $\Rightarrow$ unitary operation)

Important notes

- There is no need to know in advance the order r in order to build the circuit U_f ($\rightarrow$ no "cheating" here !)

- The above circuit performs efficiently the modular exponential with $O(n^3) = O((\log N)^3)$ gates ($\sim$ runtime)

Factoring problem

Given a (large) integer N, find a number $2 \leq d \leq N-1$ such that $d \mid N$ (read this as "d divides N"). By repeatedly solving this pb, one finds the prime factor decomposition of N.

Hardest instance of the problem: $N = P \cdot Q$ with P & Q large primes ($\Rightarrow$ very few a's)

Here is finally Shor's factoring algorithm:

1. Choose $2 \leq a \leq N-1$ unif. at random and compute $d = \gcd(a, N)$ (this requires $O((\log N)^3)$ runtime with Euclid's algo)
2. If $d > 1$ (which happens with low prob.), then d solves the factoring pb.

Assume therefore $d = 1$ in the following.

3. Compute the smallest value of $r \geq 1$ such that $a^r \pmod N = 1$.
This is done using the algorithm described in the previous lectures ($O((\log N)^3)$ runtime) (which is at the heart of Shor's algorithm).
4. If r is odd, declare failure and restart the algorithm in 1.

5. If r is even, then observe that

$$a^r - 1 = \underbrace{(a^{r/2} - 1)}_{:= d_-} \cdot \underbrace{(a^{r/2} + 1)}_{:= d_+}$$

Also by part 3, $a^r - 1 = kN$ for some $k \in \mathbb{Z}$

So $N \mid a^r - 1 = d_- \cdot d_+$

Then three different things can happen:

a) Either $N \mid d_- = a^{r/2} - 1$, but this is actually impossible, as r is by assumption the smallest value s.t. $N \mid a^r - 1$.

b) or $N \mid d_+ = a^{r/2} + 1$; in this case, declare failure and restart the algorithm in 1.

c) or N shares non-trivial prime factors with both d_- and d_+ => success!
(by computing $\gcd(N, d_-)$ or $\gcd(N, d_+)$)

Rabin & Miller showed in 1974 that the success probability of this algorithm is greater than or equal to $3/4$.
So by repeating the algo T times, one can obtain an arbitrarily small error prob.

Classically, for a & N with order m digits, finding the smallest value of $r \geq 1$ s.t.

$$a^r \pmod N = 1$$

requires order $\exp\left(\left(\frac{64m}{19}\right)^{1/3} \cdot (\log m)^{2/3}\right)$

runtime with the best known algorithm

$\Longrightarrow$ runtime superpolynomial in m.

In comparison, the quantum order finding algorithm described in the previous lectures finds the minimum value of $r \geq 1$ such that $a^r \pmod N = 1$ in $O(m^3) = O((\log N)^3)$ runtime, opening the door to a polynomial time resolution of the factoring problem (in theory!)