An introduction to quantum information and computation

A.y. 2023/24 — Leonardo Mazza — leonardo.mazza@universite-paris-saclay.fr

Lecture 4: The quantum Fourier transform.

1) Fourier transform.

Let us recall briefly what is the Fourier transform (in the discrete setting).

Take a vector $\vec{v} \in \mathbb{R}^N$ with $N = 2^n$, so that it can be mapped to a n-bit space.

The Fourier transform is a linear mapping to a novel vector:

$$FT: \vec{v} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{pmatrix} \longmapsto \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix} \quad \text{with} \quad w_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i \frac{2\pi}{N} j \cdot k} v_k$$

The linear operation can be represented as a matrix:

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{N-1} \end{pmatrix} = \frac{1}{\sqrt{N}} \underbrace{\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i/N} & e^{4\pi i/N} & \cdots & e^{2\pi i \frac{N-1}{N}} \\ 1 & e^{4\pi i/N} & e^{8\pi i/N} & \cdots & \\ 1 & e^{6\pi i/N} & e^{12\pi i/N} & \cdots & \\ \vdots & \vdots & \vdots & & \\ 1 & e^{2\pi i \frac{(N-1)}{N}} & & & \end{pmatrix}}_{FT_N} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{N-1} \end{pmatrix}$$

In the case $n = 1$, $N = 2$, we have:

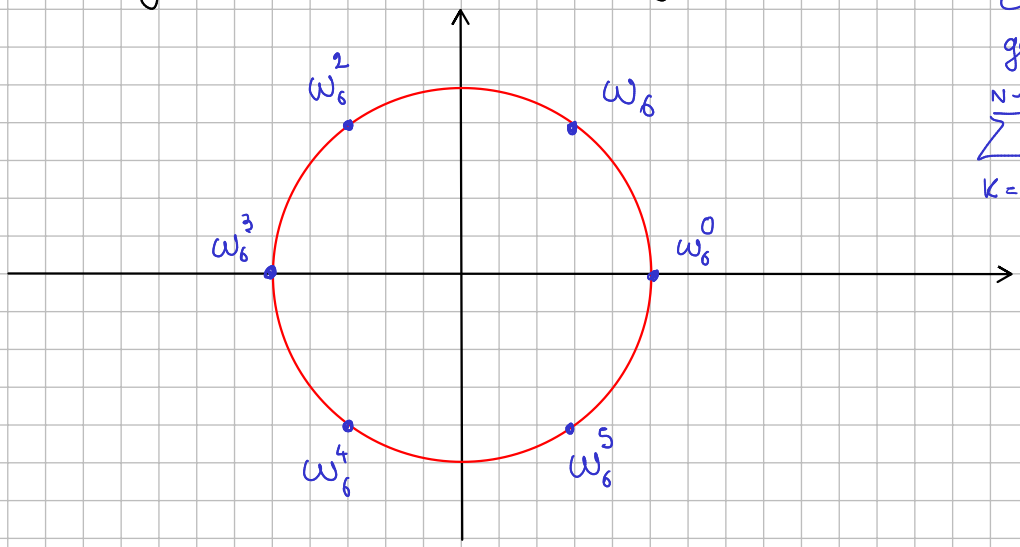
$$FT_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \rightsquigarrow \text{This is the Hadamard transformation } \hat{H}|0\rangle = +|+\rangle, \quad \hat{H}|1\rangle = +|-\rangle$$

The transformation FT_N is a unitary matrix thanks to the fact that

$$\omega_N = e^{2\pi i/N}$$ is a root of the identity:

- $(\omega_N)^N = 1$
- $\sum_{K=0}^{N-1} (\omega_N)^K = 0$
- $\sum_{K=0}^{N-1} (\omega_N^0)^K = N$

because they lie on the unitary circle:



Examples of the general formula:

$$\sum_{K=0}^{N-1} \omega_N^{mK} = N\delta_{m,0}$$

Given the vector $\vec{v}$, what is the cost of computing $\vec{w}$? There is a cost in constructing FT_N, which can be done once and stored in the laptop. Naively, the cost of computing $\vec{w}$ given $\vec{v}$ scales as N^2 because it requires N^2 multiplication. $\rightarrow \sim 2^{2n}$

At the moment, there is a classical algorithm, called Fast Fourier transform, that scales as $N \cdot \log N$. This is the best algorithm known for a classical computer.

$\sim 2^n \cdot n$

Remarks.

Unitarity of the FT. We show that $FT_N \cdot FT_N^{\dagger} = \mathbb{1}$

- $\sum_k (FT_N)_{ik} \left(FT_N^{\dagger}\right)_{kj} = \sum_k (FT_N)_{ik} \left(FT_N\right)^*_{jk} = \sum_k \frac{1}{\sqrt{N}} e^{2\pi i \frac{ik}{N}} \frac{1}{\sqrt{N}} e^{-\frac{2\pi}{N} i jk}$

$$= \frac{1}{N} \sum_k e^{i\frac{2\pi}{N} k(i-j)} = \begin{cases} 1 & \text{when } i = j \\ 0 & \text{when } i \neq j \end{cases} = \delta_{ij} \qquad \underline{\text{OK}}$$

Physical interpretation: $w_j = \sqrt{\frac{1}{N}} \sum_{k=0}^{N-1} e^{i\frac{2\pi}{N} k \cdot j} v_k$

- when $v_k = \delta_{k,\bar{k}} \rightarrow w_j = \sqrt{\frac{1}{N}} e^{i\frac{2\pi}{N} \bar{k} j} \rightarrow$ Plane wave with wavevector $\frac{2\pi}{N} \bar{k}$

- when $v_k = \frac{1}{\sqrt{N}} \rightarrow w_j = \frac{1}{N} \sum_{k=0}^{N-1} e^{i\frac{2\pi}{N} k \cdot j} = \delta_{j,0}$

2) The quantum Fourier transform.

Let us promote this to the quantum context. Since $N = 2^n$, the vector $\vec{v}$ can thus be interpreted as $|\psi\rangle$, a vector of the Hilbert space of N qubits.

Example for $n = 2$.

$$\left.\begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} \begin{matrix} \to 00 \\ \to 01 \\ \to 10 \\ \to 11 \end{matrix}\right\} \to |v\rangle = v_0|00\rangle + v_1|01\rangle + v_2|10\rangle + v_3|11\rangle.$$

The FT is a Quantum gate (unitary), $\hat{U}_{FT_N}$ and the vector $\vec{w}$ becomes: $|w\rangle$.

Hence, there is a simple analogy of $\vec{w} = FT_N \vec{v}$ which is

$$|w\rangle = \hat{U}_{FT_N} |v\rangle$$

and in the computational basis $\{|00\ldots0\rangle, |00\ldots1\rangle \ldots \quad |11\ldots1\rangle\}$ $\hat{U}_{FT_N}$ has a matrix representation that is

$$\hat{U}_{FT_N} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i/N} & e^{4\pi i/N} & \cdots & e^{2\pi i\frac{N-1}{N}} \\ 1 & e^{4\pi i/N} & e^{8\pi i/N} & \cdots & \\ 1 & e^{6\pi i/N} & e^{12\pi i/N} & \cdots & \\ \vdots & \vdots & \vdots & & \\ 1 & e^{2\pi i\frac{(N-1)}{N}} & & & \end{pmatrix}$$

Let me use another name for the basis:

$$\begin{cases} |0\rangle \equiv |000\ldots0\rangle \\ |1\rangle \equiv |000\ldots01\rangle \\ |2\rangle \equiv |000\ldots010\rangle \\ \vdots \\ |N-1\rangle \equiv |111\ldots1\rangle \end{cases}$$

In the 2-qubit case:

$$\begin{cases} |0\rangle \equiv |00\rangle \\ |1\rangle \equiv |01\rangle \\ |2\rangle \equiv |10\rangle \\ |3\rangle \equiv |11\rangle \end{cases}$$

The representation is the standard base-2 representation:

$$|0\rangle \leftrightarrow |000\ldots0\rangle$$
$$|1\rangle \leftrightarrow |000\ldots01\rangle$$
$$|2\rangle \leftrightarrow |000\ldots10\rangle$$
$$\vdots$$
$$|j\rangle \leftrightarrow |j_1 j_2 j_3 \ldots j_n\rangle \quad \rightsquigarrow j = 2^{n-1} \sum_{k=1}^{n} \frac{1}{2^{k-1}} j_k$$

For $n=2, N=4$ $\rightarrow$

$$|11\rangle \rightsquigarrow j = 2\left(1 + \frac{1}{2}\right) = 2+1 = 3$$
$$|10\rangle \rightsquigarrow j = 2\,(1+0) = 2$$
$$|01\rangle \rightsquigarrow j = 2\,(0+\tfrac{1}{2}) = 1$$
$$|00\rangle \rightsquigarrow j = 2\,(0+0) = 0$$

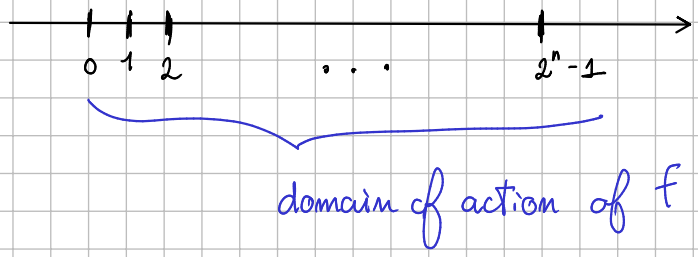
Remark: $\hat{U}_{FT}$ is unitary. $\rightarrow$ it can be interpreted as a quantum gate.

3) Periodicity determination

The answer can be encoded in n qubits

Let us consider a function $f : \{0,1\}^{\otimes m} \longrightarrow \{0, 1 \dots 2^n - 1\}$

The n-bits states can be ordered according to the mapping defined above. Since we have a notion of order, we can discuss periodic function.

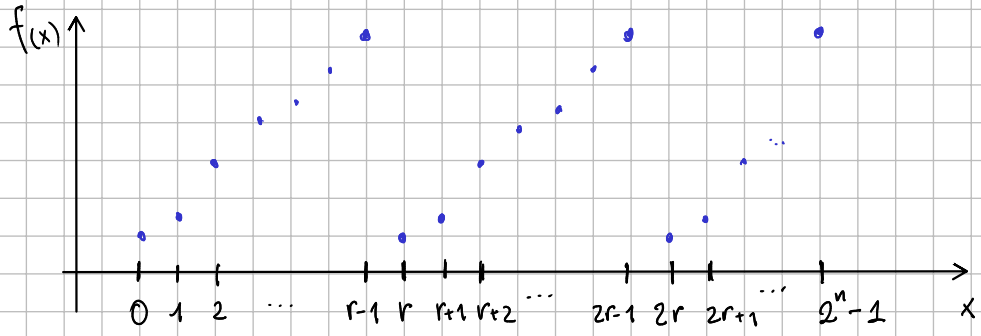


A periodic function with period r has the property

$$f(\underbrace{x+r}_{\text{mod } N}) = f(x)$$

The period r should divide exactly 2^n (other cases can be considered, but we will not discuss them here), hence $r = 2^m$ with $m \leq n$

We will also assume that in the period $[0, r-1]$, the function never takes twice the same value:



A function f with period r with the properties that we consider.

The goal is to find r, the period of this function.

Recap of notation:

$$\begin{cases} N = 2^n & : \text{\# of points} \\ r = 2^m & : \text{period of the function} \\ A = \frac{N}{r} = 2^{n-m} & : \text{\# of periods repeated on the N points.} \end{cases}$$

The best classical algorithm finds r with a given probability $1-\varepsilon$ for fixed N in a number of interrogations of f that scales as $\sqrt{N} \sim 2^{n/2}$. Exponential in n, # of bits.

Let us now consider the oracle associated to F:

- $\hat{\mathcal{O}}_f |0\rangle|j\rangle = |f(j)\rangle|j\rangle$

  $|0\rangle$: n qubits, OUTPUT; $|j\rangle$: n qubits, INPUT

In total $\mathcal{O}_F$ acts on $2n$ qubits.

We initialise the qubits in $|000\ldots0\rangle$, we apply a Hadamard on each input qubit, so that the initial state is

$$|\psi\rangle = |0\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_j |j\rangle = |0\rangle \otimes |\psi_{k=0}\rangle$$

$|0\rangle$: n qubits; $|j\rangle$: n qubits; $|\psi_{k=0}\rangle$: Uniform state.

We now apply the ORACLE $\hat{\mathcal{O}}_f$ and hence:

$$|\psi'\rangle = \hat{\mathcal{O}}_f |\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_j |f(j)\rangle \otimes |j\rangle$$

Now we measure the output qubits and we obtain the value: y_0.
There is $x_0 \in [0, r-1]$ such that $f(x_0) = y_0$, but also $f(x_0 + pr) = y_0$, with $p \in \mathbb{N}$
Hence, the state $|\psi'\rangle$ is projected onto:

$$|\psi''\rangle = \frac{1}{\sqrt{A}} \sum_{p=0}^{A-1} |y_0\rangle \otimes |x_0 + pr\rangle$$

where $A = \frac{2^n}{r}$ is the number of periods of the function.

Remember that $r = 2^m$ and hence: $A = 2^{n-m}$.

$$= |y_0\rangle \otimes \underbrace{\left(\frac{1}{\sqrt{A}} \sum_{p=0}^{A-1} |x_0 + pr\rangle \right)}_{|\chi''\rangle}$$

It is clear that $|\psi''\rangle$ contains information on r, the question is how to extract it.

1) The first idea would be to make a measurement of $|\psi''\rangle$.
If I get $|x_0 + pr\rangle$, what do I learn? Nothing.
I need to repeat the measurement and get statistics.
But this is a problem. Everytime I perform the FIRST measurement I get a different $|y_0\rangle$, say $|y_1\rangle \rightarrow$ hence a different $|x_1 + p'r\rangle$. The dependence on the result of the previous measurement is called the postselection problem.
Very large number of measurements is necessary.

2) To be a bit more clever, we apply $\hat{U}_{FT_{2^n}}$ on $|\psi''\rangle$.

$$|\psi'''\rangle = \hat{U}_{FT_N} |\psi''\rangle = \frac{1}{\sqrt{A}} \sum_{p=0}^{A-1} \hat{U}_{FT_N} |x_0 + pr\rangle = \frac{1}{\sqrt{A}} \sum_{p=0}^{A-1} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i}{N} j(x_0+pr)} |j\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{2\pi i j x_0}{N}} \left(\frac{1}{\sqrt{A}} \sum_{p=0}^{A-1} e^{\frac{2\pi i \, j \cdot p}{(N/r)}} \right) |j\rangle$$

$$\frac{1}{\sqrt{A}} \sum_{p=0}^{A-1} e^{\frac{2\pi i \, j \cdot p}{A}} = \begin{cases} A & \text{for } j = 0, A, 2A, 3A \ldots \\ 0 & \text{otherwise.} \end{cases}$$

$$= \sqrt{\frac{A}{N}} \sum_{q=0}^{r-1} e^{\frac{2\pi i \, qA}{N} x_0} |q \cdot A\rangle = \frac{1}{\sqrt{r}} \sum_{q=0}^{r-1} e^{\frac{2\pi i \, qA}{N} x_0} |qA\rangle$$

Hence, $|\psi'''\rangle$ has overlap only on states $|j\rangle$ such that:

$$|j=0\rangle, \; |j=A\rangle, \; |j=2A\rangle \; \ldots \; |j=(r-1)A\rangle$$

with $A = N/r$.

Can I extract the value of r with a single measurement?

The result of the first measurement was y_0, such that $f(x_0) = y_0$ with x_0 in the first period. The final state $|\psi'''\rangle$ depends on that result only via a phase factor $\longrightarrow \exp\left(2\pi i \frac{j x_0}{N}\right)$.

If we measure in the computational basis the state $|\psi'''\rangle$, we obtain

$$qA \quad \text{with probability } \frac{1}{r}$$

$$q \in \{0, 1, 2 \dots r-1\}$$

By repeating the measurement, we obtain the values qA with $q \in \{0, 1, 2 \dots r-1\} \longrightarrow$ we have reconstructed the value r.
This strategy suffers from the postselection problem.
What is the most efficient way to reconstruct the value of r?
Suppose you do one measurement, and get

$$c = \bar{q} A = \bar{q} \frac{N}{r}$$

Note that $c \in \mathbb{N}$, $N = 2^n \in \mathbb{N}$
$r = 2^m \in \mathbb{N}$, $\bar{q} \in \mathbb{N}$

Compute $\frac{c}{N} \in \mathbb{Q}$ (remember that N is known). We know that $\frac{c}{N} = \frac{\bar{q}}{r}$. If $\bar{q}$ and r are coprime, we know r. After one measurement. How do we know whether it worked? Just check!

Evaluate $f(0)$ and $f(r_{guess})$, where r_{guess} is the result obtained assuming that $\bar{q}$ is coprime to r.
If $\bar{q}$ and r are not coprime, you will estimate a $r_{guess} < r$, hence the test will show that $f(0) \neq f(r_{guess})$.

Theorem 1 *(Coprimality theorem) The number of integers less than r that are coprime to r grows as $O(r/\log\log r)$ with increasing r. Hence if $k_0 < r$ is chosen at random*

$$prob(k_0 \text{ coprime to } r) \approx O((r/\log\log r)/r) = O(1/\log\log r). \qquad \square$$

Thus if we repeat the whole process $O(\log\log r) < O(\log\log N)$ times we will obtain a coprime k_0 in at least one case with a constant level of probability. Here we have used the following fact from probability theory:

Lemma 1 *If a single trial has success probability p and we repeat the trial M times independently then for any constant $0 < 1-\epsilon < 1$:*

$$\textit{prob(at least one success in } M \textit{ trials)} > 1-\epsilon \textit{ if } M = \frac{-\log \epsilon}{p}$$

so to achieve any constant level $1-\epsilon$ of success probability, $O(1/p)$ trials suffice.

What is the computational cost of all this?

- Each time we need to evaluate f three times (one at the beginning, two at the end).
- We need to repeat the procedure $\log\log N \sim \log\log 2^n \sim \log n$ to have a high probability that we get r.
- We need to create the $\hat{U}_{FT_N}$ quantum gate. How difficult is that? We will see that it is a particularly lucky situation as it requires $(\log N)^2 \sim n^2$ operations.

In summary, we can obtain the result with a given high probability $1-\varepsilon$ in $\log n$ interrogations and with a circuit composed of n^2 operations.

This has to be contrasted with the classical case where the period determination requires $2^{n/2}$ operations.

EXPONENTIAL ADVANTAGE!

4) Factoring a number in prime numbers.

Given N with $n \sim \log N$ digits, find its prime factors.

We know that this mathematical problem is crucial in current cryptographic schemes. This problem is useful because:

i) it is not easy to find the prime factors of a number

ii) it is easy to check whether a number p_1 is a factor of N.
Indeed the calculation of N/p_1 is a problem whose difficulty scales linearly with $n \sim \log N$, the digits of the number.
I can tell if p_1 is a factor without knowing in advance the answer.

Notation convention for the following: we will not write $N/p_1 \in \mathbb{N}$ but rather $N \equiv 0 \mod p_1$.

Let us get back to i): what is the best classical algorithm for prime factoring? Just divide N by all numbers smaller than $\sqrt{N}$.

Complexity: $\sim \sqrt{N} \sim 2^{n/2} \sim e^{\frac{1}{2}\log N}$

Best known algorithm so far $\sim e^{(\log N)^{1/3}(\log\log N)^{2/3}}$

Still an exponential scaling in n, the digits, or if you want the bits that are necessary to encode the number.

5) Shor's quantum algorithm.

Shor's algorithm for the factorisation of N that is not prime relies on the quantum Fourier transform. Using some arguments from number theory, he links the problem of number factorisation to the search of the periodicity of a function.

The algorithm. INPUT: N n-bit number.

Steps of the algorithm.

The algorithm starts with a few simple checks that one can easily perform.

1) N is even? YES: 2 is a factor. NO: go to 2)

2) Check whether $N = m^k$ with $1 < m \leq \sqrt{N}$ and $2 \leq k \leq \log N$
Important and not obvious: this check can be done efficiently.
YES: m is a factor. NO: go to 3).

3) Generate a random number a with $1 < a < N$
Check whether $s = \gcd(N, a)$ is different from 1.
Yes? Great, you have a factor of N. (very unlikely)
No? then N and a are co-primes.

4) According to a theorem by Euler, when $a < N$ are co-prime there is a number $1 < r < N$ such that:

$$a^r \equiv 1 \bmod N.$$

r: is the order of a mod N if it is the smallest possible.

Note that this means:

$$a^r = \alpha N + 1 \Rightarrow a^{r+1} = a\alpha N + a$$

$$\Rightarrow a^{r+1} \bmod N \equiv a \bmod N$$

We obtain that the function $f(n) = a^n \bmod N$ has periodicity r, the order of $a \bmod N$

Use the algorithm detailed at the previous section to find r.

Is r odd? Bad luck, go back to 3) and generate another number a.

Is r even? Continue.

We wrote $a^r = 1 \bmod N \iff a^r - 1 \equiv 0 \bmod N$

$$\underbrace{(a^{r/2} - 1)}_{\alpha} \cdot \underbrace{(a^{r/2} + 1)}_{\beta} \equiv 0 \bmod N.$$

This means that $\alpha \cdot \beta = cN$.

- It is impossible that $\alpha \equiv 0 \bmod N$. This would imply that $\alpha \equiv 0 \bmod N \rightarrow (a^{r/2} - 1) \equiv 0 \bmod N$
  $r/2$ would be the order of $a \bmod N$ but it was $r > \frac{r}{2}$!
  Absurd.

- It is possible that $\beta \equiv 0 \bmod N \rightarrow \beta$ is a multiple of N and α, N are co-prime. Bad luck! Go back to 3) and generate a new a.

- Much likely, α contains some factors of N and β contains some factors of N.
  Use the Euclide algorithm to compute

$$\gcd(\alpha, N) = t_1 \qquad \gcd(\beta, N) = t_2$$

t_1 and t_2 are factors of N.

6) Remarks on number theory to make Shor's algorithm a bit clearer.

Theorem 2 *(Euler's theorem): If a and N are coprime then there is a least power $1 < r < N$ such that $a^r \equiv 1 \bmod N$. r is called the order of a mod N.*

Let us first of all check the theorem with some examples.

① $a = 2, N = 3$ $\quad a^2 = 2^2 = 4 \equiv 1 \bmod 3$ OK $\quad r = 2$

② $a = 2, N = 5$

$$a = 2 \equiv 2 \bmod 5$$
$$a^2 = 4 \equiv 4 \bmod 5 \equiv 2 \cdot a \bmod 5$$
$$a^3 = 8 \equiv 3 \bmod 5 = 2 \cdot 4 \bmod 5$$
$$a^4 = 16 = 1 \bmod 5 = 2 \cdot 3 \bmod 5$$
$$a^5 = 32 = 2 \bmod 5 = 2 \cdot 1 \bmod 5$$

4 is the order of 3 mod N.

Note that $a \equiv b \bmod N \Rightarrow a^2 \bmod N = ab \bmod N$

PROOF: $a = \alpha N + b \rightarrow a^2 = a\alpha N + ab$

③ $a = 3, N = 4$

$$a = 3 \equiv 3 \bmod 4$$
$$a^2 = 9 \equiv 1 \bmod 4$$
$$a^3 = 27 \equiv 3 \bmod 4$$
$$\vdots$$

2 is the order of 3 mod 4

④ What happens if a and N are not co-prime?

$a = 2, N = 6$

$$a = 2 \equiv 2 \bmod 6$$
$$a^2 = 4 \equiv 4 \bmod 6$$
$$a^3 = 8 \equiv 2 \bmod 6$$
$$\vdots$$

Cyclic structure but without 1.

⑤ Another situation $\quad a = 6 \quad N = 12 \quad$ (a divides N)

$$a = 6 \equiv 6 \bmod 12$$
$$a^2 = 36 = 0 \bmod 12$$
$$a^3 = 0 \bmod 12$$
$$\vdots$$

$f(n) \doteq a^n \bmod N$ takes values in $\{0, 1 \dots N-1\}$.

- Assume $f(\bar{n}) = 0$ for $\bar{n} > 0$. Then $a^{\bar{n}} \bmod N = 0$

  then $a^{\bar{n}+1} \bmod N = 0 \cdot a \bmod N = 0$

  $\Rightarrow f(n) = 0 \quad \forall n \geq \bar{n}$

  $a^{\bar{n}} = cN \qquad p_1^n p_2^n p_3^n \dots p_m^n = cN$

  $\Rightarrow a$ and N are not co-primes.

- If a and N are coprime $f(n) \neq 0 \quad \forall n$.

  Basically in this case we are studying the multiplicative group of order N with elements $\{1, \dots N\}$ and group operation

  $$a \otimes b \doteq ab \bmod N.$$

This object is well behaved (it is a group) when N is prime.

- When N is prime, the group is cyclic with order N.

  Hence: $a^N \bmod N = a \bmod N$ (Little Fermat's theorem)

- When N is odd and not prime it is composed of cyclic subgroups by repeated action of a

  $a \bmod N$
  $a^2 \bmod N$
  $\vdots$
  $a^n \bmod N$

  } it is a subgroup, hence you don't explore the full group $\{1 \dots N\}$

6 bis) On the Euclid algorithm.

At the end of the Shor's algorithm, we use many times that we need to compute the greatest common divisor of two numbers. How to do that? There is an efficient algorithm due to Euclid, based on the fact that

$$\text{for } a>b \qquad g.c.d.(a,b) = g.c.d.(a-b,b)$$

PROOF $\quad gcd(a,b) = c \Rightarrow a = \alpha c \;\&\; b = \beta c$ c is the largest possible divisor

$$\rightarrow a-b = (\alpha-\beta)\,c$$

Hence: c is a divisor of $a-b$. It is actually the greatest common divisor of $a-b$ and b because if $d>c$ is $gcd(b,a-b)$ then

$$b-a = \gamma' d \qquad b = \beta' d$$

$$\rightarrow a = (\beta'-\gamma')\,d \text{ and } dx \text{ is } gcd(a,b)$$

Absurd!

Euclid: $gcd(99,30) = gcd(69,30) = gcd(39,30)$

$= gcd(30,9) = gcd(21,9) = gcd(12,9)$

$= gcd(3,9) = gcd(6,3) = gcd(3,3).$

$$gcd(99,30) = 3.$$

6 (this) On the probabilistic nature of Shor's algorithm.

In the Shor's algorithm we have encountered two "bad luck" situations where we had to restart the algorithm:

- Compute r, and r is odd.
- r is even but $(a^{r/2}+1) \equiv 0 \mod N$

The algorithm by Shor does not work if this happens too often. There is a theorem that guarantees that there will not be problems:

> **Theorem 3** *Suppose N is odd and not a power of a prime. If $a < N$ is chosen uniformly at random with $gcd(a, N) = 1$ then Prob(r is even and $a^{r/2} \not\equiv -1 \mod N) \geq 1/2$.*
>
> For a proof of this result see Preskill's notes page 307 et seq., Nielsen/Chuang appendix 4.3 or A. Ekert and R. Jozsa, *Reviews of Modern Physics*, vol 68, p733-753 1996, appendix B.

This explains why we need to perform the checks at 1) and 2).

7) Is Shor's algorithm efficient?

Best classical algorithm $e^{\sqrt[3]{n}}$ See earlier these notes.

Quantum: $O(n^3)$ [Not discussed the proof, too long].

EXPONENTIAL ADVANTAGE; from exponential scaling to polynomial.