QUANTUM COMPUTATION AND QUANTUM INFORMATION: THE QUANTUM FOURIER TRANSFORM

1.

We consider the linear map in $\mathbb{C}^N$ which acts on the computational basis as

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2i\pi jk}{N}} |k\rangle$$

Let A be the matrix of the transformation in the computational basis.

$$\forall (k,l) \in [\![0, N-1]\!]^2, \quad a_{kl} = \frac{1}{\sqrt{N}} e^{\frac{2i\pi kl}{N}}$$

The adjoint matrix $A^\dagger$ is then

$$\begin{aligned}\forall (k,l) \in [\![0, N-1]\!]^2, \quad b_{kl} &= a_{lk}^* \\ &= \frac{1}{\sqrt{N}} e^{-\frac{2i\pi kl}{N}}\end{aligned}$$

We compute the coefficient k, l of the product $AA^\dagger$:

$$\begin{aligned}\forall (k,l) \in [\![0, N-1]\!]^2, \quad c_{kl} &= \sum_{j=0}^{N-1} a_{kj} b_{jl} \\ &= \frac{1}{N} \sum_{j=0}^{N-1} e^{\frac{2i\pi j}{N}(k-l)} \\ &= \frac{1}{N} \sum_{j=0}^{N-1} (e^{\frac{2i\pi}{N}(k-l)})^j \\ &= \begin{cases} \frac{1}{N} \frac{1-(e^{\frac{2i\pi}{N}(k-l)})^N}{1-e^{\frac{2i\pi}{N}(k-l)}} = 0 & \text{if } e^{\frac{2i\pi}{N}(k-l)} \neq 1, \\ 1 & \text{if } e^{\frac{2i\pi}{N}(k-l)} = 1. \end{cases} \\ &= \begin{cases} 0 & \text{if } k \neq l, \\ 1 & \text{if } k = l. \end{cases} \\ &= \delta_{kl}\end{aligned}$$

which shows that $AA^\dagger = A^\dagger A = I$ i.e. A is unitary.

2.

Here the dimension of the state space is $N = 2^n$. The Fourier transform of the n qubit state $|00\ldots0\rangle$ is

$$A|0\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle$$

we can write k in binary $k_{n-1}\ldots k_1 k_0$

$$A|0\rangle = \frac{1}{2^{n/2}} \sum_{k_0, k1, \ldots, k_{n-1}=0}^{1} |k_{n-1}\ldots k_1 k_0\rangle$$

or in product representation,

$$= \frac{1}{2^{n/2}} \underbrace{(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)\ldots(|0\rangle + |1\rangle)}_{n \text{ qubits}}$$

3.

Let $N = 2^n$ and $Y = (y_k)_{k\in[\![0,N-1]\!]}$ be the classical fourier transform of $X = (x_k)_{k\in[\![0,N-1]\!]}$.

$$\forall k \in [\![0, N-1]\!], \quad y_k = \sum_{j=0}^{N-1} e^{\frac{2i\pi kj}{2^n}} x_j$$

The factor $\frac{1}{\sqrt{N}}$ is omitted for clarity. We can write j in binary $j_{n-1}\dots j_1 j_0$

$$\begin{aligned}
y_k &= \sum_{j_0,j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-1}j_{n-1}+\dots+2j_1+j_0)}{2^n}} x_j \\
&= \sum_{j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-1}j_{n-1}+\dots+2j_1)}{2^n}} x_{j_{n-1}\dots j_1 0} + \sum_{j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-1}j_{n-1}+\dots+2j_1+1)}{2^n}} x_{j_{n-1}\dots j_1 1} \\
&= \sum_{j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-1}j_{n-1}+\dots+2j_1)}{2^n}} x_{j_{n-1}\dots j_1 0} + e^{\frac{2i\pi k}{2^n}} \sum_{j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-1}j_{n-1}+\dots+2j_1)}{2^n}} x_{j_{n-1}\dots j_1 1} \\
&= \sum_{j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-2}j_{n-1}+\dots+j_1)}{2^{n-1}}} x_{j_{n-1}\dots j_1 0} + e^{\frac{2i\pi k}{2^n}} \sum_{j_1,\dots,j_{n-1}=0}^{1} e^{\frac{2i\pi k(2^{n-2}j_{n-1}+\dots+j_1)}{2^{n-1}}} x_{j_{n-1}\dots j_1 1}
\end{aligned}$$

We see the first sum is the k^{th} coefficient of the FT of the sequence $(x_{2k})_{k\in[\![0,N/2-1]\!]}$ and the second is the k^{th} coefficient of the FT of $(x_{2k+1})_{k\in[\![0,N/2-1]\!]}$. This shows that to compute FT of sequence of length N, we have to compute 2 FT of sequence of length $\frac{N}{2}$ and do $2N$ complex additions/multiplications. The complexity of the operation $T(N)$ follows the recurrence:

$$T(N) = 2T(\frac{N}{2}) + 2N$$

We can use the Master theorem [1]:

Theorem. Let $a \geqslant 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the non negative integers by the recurrence

$$T(n) = aT(\frac{n}{b}) + f(n)$$

where we interpret $\frac{n}{b}$ to mean either $\lfloor \frac{n}{b} \rfloor$ or $\lceil \frac{n}{b} \rceil$. Then $T(n)$ has the following asymptotic bounds:

(1) If $f(n) = O(n^{\log_b a-\epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
(2) If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a}\log n)$.
(3) If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some constant $\epsilon > 0$, and if $af(\frac{n}{b}) \leqslant cf(n)$ for some constant $c < 1$ and n sufficiently large, then $T(n) = \Theta(f(n))$.

Here we are in the second case of the theorem, so $T(N) = \Theta(N\log(N)) = \Theta(n2^n)$.

Instead of $\mathbb{C}$, the Fourier transform may be used in any ring as soon as we are given a Nth root of unity. The book *The design and analysis of computer algorithms* [2] provides an overview of the FFT, an algorithm using bits operations and application to fast integer multiplication.

5.

The inverse Fourier Transform

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-\frac{2i\pi jk}{N}} |k\rangle$$

is the adjoint of the Fourier Transform. The quantum circuit of figure 1 is obtained from the FT's circuit, replacing each R_k gate by its adjoint

$$R_k^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{2i\pi}{2^k}} \end{bmatrix}$$

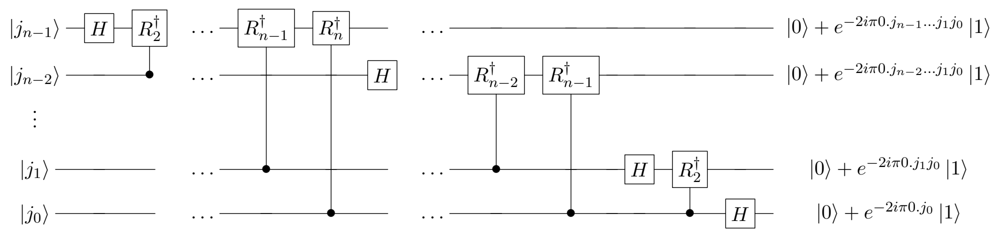
$|j_{n-1}\rangle$
$|j_{n-2}\rangle$
$|j_1\rangle$
$|j_0\rangle$
H
$R_2^\dagger$
$R_{n-1}^\dagger$
$R_n^\dagger$
$R_{n-2}^\dagger$
$|0\rangle + e^{-2i\pi 0.j_{n-1}\ldots j_1 j_0}|1\rangle$
$|0\rangle + e^{-2i\pi 0.j_{n-2}\ldots j_1 j_0}|1\rangle$
$|0\rangle + e^{-2i\pi 0.j_1 j_0}|1\rangle$
$|0\rangle + e^{-2i\pi 0.j_0}|1\rangle$

FIGURE 1. Quantum circuit for IFT.

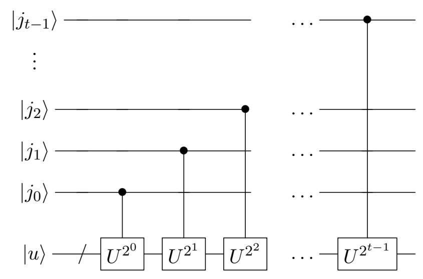

FIGURE 2. Sequence of controlled U.

7.

In figure 2, the t qubits of the first register are prepared with $|j\rangle = |j_{t-1}\ldots j_1 j_0\rangle$, the second register is prepared with some state $|u\rangle$. After the first controlled-U operation, the state is $|j\rangle |U^{j_0 2^0}u\rangle$. After the second controlled-U, the state is $|j\rangle |U^{j_1 2^1}U^{j_0 2^0}u\rangle = |j\rangle |U^{j_0 2^0 + j_1 2^1}u\rangle$ and so on. The final state is $|j\rangle |U^{j_0 2^0 + j_1 2^1 + \cdots + j_{t-1}2^{t-1}}u\rangle = |j\rangle |U^j u\rangle$.

8.

By linearity, the phase estimation algorithm takes input $|0\rangle |\Sigma_{u\in A} c_u |u\rangle\rangle$, where A is some orthonormal basis of eigenstates of U, to output $\sum_{u\in A} c_u |\widetilde{\varphi_u}\rangle |u\rangle$, where $\widetilde{\varphi_u}$ is an estimation of the phase of the eigenvalue associated with eigenstate u. If we fix $u_0 \in A$ beforehand, the probability to measure $\widetilde{\varphi_{u_0}}$ when measuring the first register in the computational basis is

$$\begin{aligned}
(\sum_{u\in A} c_u^* \langle\widetilde{\varphi_u}| \langle u|) P_{\widetilde{\varphi_{u_0}}} \otimes I (\sum_{u\in A} c_u |\widetilde{\varphi_u}\rangle |u\rangle) &= (\sum_{u\in A} c_u^* \langle\widetilde{\varphi_u}| \langle u|)(\sum_{\substack{u\in A\\ \widetilde{\varphi_u}=\widetilde{\varphi_{u_0}}}} c_u |\widetilde{\varphi_u}\rangle |u\rangle) \\
&= (\sum_{u\in A} c_u^* \langle\widetilde{\varphi_u}| \langle u|)(\sum_{\substack{u\in A\\ \widetilde{\varphi_u}=\widetilde{\varphi_{u_0}}}} c_u |\widetilde{\varphi_{u_0}}\rangle |u\rangle) \\
&= \sum_{\substack{v\in A\\ u\in A\\ \widetilde{\varphi_u}=\widetilde{\varphi_{u_0}}}} c_v^* c_u \langle\widetilde{\varphi_v}|\widetilde{\varphi_u}\rangle \langle v|u\rangle \\
&= \sum_{\substack{v\in A\\ u\in A\\ \widetilde{\varphi_u}=\widetilde{\varphi_{u_0}}}} c_v^* c_u \langle\widetilde{\varphi_v}|\widetilde{\varphi_u}\rangle \delta_{vu} \\
&= \sum_{\substack{u\in A\\ \widetilde{\varphi_u}=\widetilde{\varphi_{u_0}}}} |c_u|^2 \\
&\geqslant |c_{u_0}|^2
\end{aligned}$$

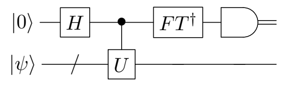

FIGURE 3. Phase estimation circuit with $t = 1$.

I is the identity operator of whatever state space U operates on, while $P_{\widetilde{\varphi_{u_0}}}$ is the orthonormal projector onto the space generated by the vector $|\widetilde{\varphi_{u_0}}\rangle$ of the computational basis. Besides, following the analysis of the book, $\widetilde{\varphi_{u_0}}$ is an approximation to φ_{u_0} to an accuracy 2^{-n} with probability at least $1-\epsilon$ if we make use of $t = n + \lceil \log(2 + \frac{1}{2\epsilon})\rceil$ bits in the first register. We conclude we get the desired approximation of φ_{u_0} at the end of the phase estimation algorithm with probability at least $|c_{u_0}|^2(1-\epsilon)$.

9.

U being unitary with eigenvalues -1 and $+1$, the state space is the direct sum of the two orthogonal eigenspaces $E_{-1} \oplus E_1$. Thus we can uniquely decompose any $|\psi\rangle = |\psi_{-1}\rangle + |\psi_{+1}\rangle$, with $|\psi_{-1}\rangle \in E_{-1}$ ans $|\psi_{+1}\rangle \in E_{+1}$. Then $-1 = e^{i\pi} = e^{2i\pi 0.1}$ and $1 = e^0 = e^{2i\pi 0.0}$ shows that is is sufficient to make use of $t = 1$ wire in the first register in the phase estimation procedure to read directly the phase of any eigenvector. If we use $|0\rangle|\psi\rangle$ as input in the circuit of figure 3, the output before the final measurement will be $|0\rangle|\psi_{+1}\rangle + |1\rangle|\psi_{-1}\rangle$.

When we measure the first register, we obtain 0 with probability

$$\begin{aligned}(\langle 0|\langle\psi_{+1}| + \langle 1|\langle\psi_{-1}|)P_0 \otimes I(|0\rangle|\psi_{+1}\rangle + |1\rangle|\psi_{-1}\rangle) &= (\langle 0|\langle\psi_{+1}| + \langle 1|\langle\psi_{-1}|)(|0\rangle|\psi_{+1}\rangle) \\ &= \langle 0|0\rangle\langle\psi_{+1}|\psi_{+1}\rangle \\ &= \langle\psi_{+1}|\psi_{+1}\rangle\end{aligned}$$

or 1 with probability

$$\begin{aligned}(\langle 0|\langle\psi_{+1}| + \langle 1|\langle\psi_{-1}|)P_1 \otimes I(|0\rangle|\psi_{+1}\rangle + |1\rangle|\psi_{-1}\rangle) &= (\langle 0|\langle\psi_{+1}| + \langle 1|\langle\psi_{-1}|)(|1\rangle|\psi_{-1}\rangle) \\ &= \langle 1|1\rangle\langle\psi_{-1}|\psi_{-1}\rangle \\ &= \langle\psi_{-1}|\psi_{-1}\rangle\end{aligned}$$

The state will collapse respectively into $\frac{1}{\sqrt{\langle\psi_{+1}|\psi_{+1}\rangle}}|0\rangle|\psi_{+1}\rangle$ or $\frac{1}{\sqrt{\langle\psi_{-1}|\psi_{-1}\rangle}}|1\rangle|\psi_{-1}\rangle$. Thus if we read 0 in the first register, that means that we have an eigenvector associated to eigenvalue $+1$ in the second register, and if we read 1 in the first register, that means that we have an eigenvector associated to eigenvalue -1 in the second register.

Once we have noticed that the FT in dimension $N = 2^1$ is just the Hadamard operator, we conclude the phase estimation circuit in this particular case is the just the same as the circuit of exercice 4.34.

10.

$$\begin{aligned} x^2 &= \quad 25 &&= 4 \\ x^3 &= \quad 20 &&= -1 \\ x^4 &= \quad 4^2 &&= 16 \\ x^5 &= 16 \times 5 &&= 80 \\ & &&= 17 \\ x^6 &= (-1)^2 &&= 1 \end{aligned}$$

11.

Theorem (*Euler*)**.** For $N \in \mathbb{N}^*$, let

$$\varphi(N) = \#\{m \in [\![1, N]\!], m \wedge N = 1\}$$

We have

$$\forall x \in \mathbb{N}^*, \quad x \wedge N = 1 \Rightarrow x^{\varphi(N)} = 1 \mod N$$

Then by definition of the order r, $r \leqslant \varphi(N) \leqslant N$.

12.

Since $x \wedge N = 1$, from Bezout's Theorem $\exists (u,v) \in \mathbb{Z}^2$ such that $ux + vN = 1$ that is $\exists u$ such that $ux = 1 \mod N$ which shows that x has a multiplicative inverse $x^{-1} = u$ in the ring $(\frac{\mathbb{Z}}{N\mathbb{Z}}, +, \times)$. We define the linear map $U^{'}$ on $(\mathbb{C}^2)^{\otimes L} \cong \mathbb{C}^{2^L}$ that acts on the computational basis as

$$\forall y \in \{0,1\}^L, \quad U' \left| y \right\rangle = \begin{cases} \left| x^{-1} y \mod N \right\rangle & \text{if } y < N, \\ y & \text{if } y \in [\![N, 2^L - 1]\!]. \end{cases}$$

We have

$$\begin{aligned}
\forall y_1, y_2 \in \{0,1\}^L, \quad \langle y_1 | U(y_2) \rangle = 1 &\Leftrightarrow y_1 = y_2 \in [\![N, 2^L-1]\!] \text{ or } (y_1, y_2 < N \text{ and } xy_2 = y_1 \mod N) \\
&\Leftrightarrow y_1 = y_2 \in [\![N, 2^L-1]\!] \text{ or } (y_1, y_2 < N \text{ and } \exists k \in \mathbb{Z}, xy_2 = y_1 + kN) \\
&\Leftrightarrow y_1 = y_2 \in [\![N, 2^L-1]\!] \text{ or } (y_1, y_2 < N \text{ and } \exists k \in \mathbb{Z}, y_2 = x^{-1}y_1 + x^{-1}kN) \\
&\Leftrightarrow y_1 = y_2 \in [\![N, 2^L-1]\!] \text{ or } (y_1, y_2 < N \text{ and } \exists k' \in \mathbb{Z}, y_2 = x^{-1}y_1 + k'N) \\
&\Leftrightarrow y_1 = y_2 \in [\![N, 2^L-1]\!] \text{ or } (y_1, y_2 < N \text{ and } x^{-1}y_1 = y_2 \mod N) \\
&\Leftrightarrow \langle U'(y_1) | y_2 \rangle = 1
\end{aligned}$$

so, since $\langle U'(y_1)|y_2\rangle, \langle y_1|U(y_2)\rangle \in \{0,1\}$,

$$\forall y_1, y_2 \in \{0,1\}^L, \quad \langle y_1 | U(y_2) \rangle = \langle U'(y_1) | y_2 \rangle$$

This shows that $U' = U^\dagger$. since it is obvious that U is invertible and $U^\dagger = U^{-1}$, we have shown that U is unitary.

13.

$(\left|u_s\right\rangle)_{s \in [\![0, r-1]\!]}$ is defined to be the IFT of the sequence $(\left|x^k \mod N\right\rangle)_{k \in [\![0,r-1]\!]}$:

$$\forall s \in [\![0, r-1]\!], \quad \left|u_s\right\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2i\pi sk}{r}} \left|x^k \mod N\right\rangle$$

Thus the equalities

$$\forall k \in [\![0, r-1]\!], \quad \left|x^k \mod N\right\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2i\pi sk}{r}} \left|u_s\right\rangle$$

just state the fact that $(\left|x^k \mod N\right\rangle)_{k \in [\![0,r-1]\!]}$ is the FT of the sequence $(\left|u_s\right\rangle)_{s \in [\![0,r-1]\!]}$. Let's check this. Let $k \in [\![0, r-1]\!]$,

$$\begin{aligned}
\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2i\pi sk}{r}} \left|u_s\right\rangle &= \frac{1}{r} \sum_{s=0}^{r-1} e^{\frac{2i\pi sk}{r}} \sum_{j=0}^{r-1} e^{-\frac{2i\pi sj}{r}} \left|x^j \mod N\right\rangle \\
&= \frac{1}{r} \sum_{j=0}^{r-1} (\sum_{s=0}^{r-1} (e^{\frac{2i\pi(k-j)}{r}})^s) \left|x^j \mod N\right\rangle \\
&= \frac{1}{r} \sum_{j=0}^{r-1} r\delta_{jk} \left|x^j \mod N\right\rangle \\
&= \left|x^k \mod N\right\rangle
\end{aligned}$$

For $k = 0$ we obtain

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|u_s\right\rangle = \left|1\right\rangle$$

15.

The easiest way is to think with the prime decomposition of the integers x and y. Let $d = x \wedge y$ and $m = x \vee y$. Let $p_0, p_1, \ldots, p_n$ be the prime numbers which appear in either prime decomposition. We can write

$$\begin{aligned} x &= p_0^{\alpha_0} p_1^{\alpha_1} \ldots p_n^{\alpha_n} \\ y &= p_0^{\beta_0} p_1^{\beta_1} \ldots p_n^{\beta_n} \end{aligned}$$

where $\alpha_i, \beta_i \in \mathbb{N}$. Then it is clear that

$$\begin{aligned} d &= p_0^{\gamma_0} p_1^{\gamma_1} \ldots p_n^{\gamma_n} \\ m &= p_0^{\delta_0} p_1^{\delta_1} \ldots p_n^{\delta_n} \end{aligned}$$

where $\gamma_i = \min(\alpha_i, \beta_i)$ and $\delta_i = \max(\alpha_i, \beta_i)$. We have $\alpha_i + \beta_i = \gamma_i + \delta_i$. Then,

$$\begin{aligned} md &= p_0^{\gamma_0} p_1^{\gamma_1} \ldots p_n^{\gamma_n} p_0^{\delta_0} p_1^{\delta_1} \ldots p_n^{\delta_n} \\ &= p_0^{\gamma_0+\delta_0} p_1^{\gamma_1+\delta_1} \ldots p_n^{\gamma_n+\delta_n} \\ &= p_0^{\alpha_0+\beta_0} p_1^{\alpha_1+\beta_1} \ldots p_n^{\alpha_n+\beta_n} \\ &= xy \end{aligned}$$

16.

Let $x \geqslant 2$.

$$\begin{aligned} \int_x^{x+1} \frac{1}{y^2}\,\mathrm{d}y &= \frac{1}{x} - \frac{1}{x+1} \\ &= \frac{1}{x(x+1)} \end{aligned}$$

since

$$x + 1 \leqslant \frac{3}{2}x \Leftrightarrow 2 \leqslant x$$

$$\int_x^{x+1} \frac{1}{y^2}\,\mathrm{d}y = \frac{1}{x(x+1)} \geqslant \frac{2}{3x^2}$$

If we sum these inequalities

$$\sum_{q=2}^{+\infty} \frac{1}{q^2} \leqslant \frac{3}{2} \sum_{q=2}^{+\infty} \int_q^{q+1} \frac{1}{y^2}\,\mathrm{d}y = \frac{3}{2} \int_2^{+\infty} \frac{1}{y^2}\,\mathrm{d}y = \frac{3}{4}$$

and finally

$$\sum_{\substack{q \in \mathbb{N}^* \\ q \text{ is prime}}} \frac{1}{q^2} \leqslant \sum_{q=2}^{+\infty} \frac{1}{q^2} \leqslant \frac{3}{4}$$

17.

17.1. The assertion $N = a^b \Rightarrow b \leqslant L$ is obviously wrong if $N = a = 1$. Since we aim to prove an asymptotical result, we can assume that $N \geqslant 2$.

$$\begin{aligned} N = a^b &\Leftrightarrow \log N = b \log a \\ &\Leftrightarrow \frac{\log N}{\log a} = b \qquad (N \geqslant 2 \Rightarrow a \geqslant 2 \Rightarrow \log a \geqslant 1 > 0) \\ &\Rightarrow b \leqslant \log N \\ &\Leftrightarrow b \leqslant \lfloor \log N \rfloor = L - 1 < L \quad (b \in \mathbb{N}) \end{aligned}$$

17.2. Let $N = 2^l + a_{l-1}2^{l-1} + \cdots + a_1 2 + a_0$ with $l+1 \leqslant L$ and $a_i \in \{0,1\}$.

$$\begin{aligned} N &= 2^l(1 + a_{l-1}2^{-1} + \cdots + a_1 2^{-l+1} + a_0 2^{-l}) \\ &= 2^l(1+f) \end{aligned}$$

with $f \in [0,1[$.

$$\begin{aligned} \log N &= l + \log(1 + a_{l-1}2^{-1} + \cdots + a_1 2^{-l+1} + a_0 2^{-l}) \\ &= l + \log(1+f) \end{aligned}$$

where log is $\log_2$. This shows that to compute an approximation to $\log N$, we just need an approximation of log in range $[1,2[$ or any interval of the form $[t,2t[$ for instance $[\frac{3}{4},\frac{1}{2}[$. Besides,

$$\begin{aligned} \forall x \in]-1,1], \quad \ln(1+x) &= x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots + (-1)^{n+1}\frac{x^n}{n} + \ldots \\ &= \sum_{k=1}^{+\infty}(-1)^{k+1}\frac{x^k}{k} \end{aligned}$$

Let's write it until order $L-1$:

$$\forall x \in]-1,+\infty[, \quad \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + \cdots + (-1)^{L+1}\frac{x^{L-1}}{L-1} + \sum_{k=L}^{+\infty}(-1)^{k+1}\frac{x^k}{k}$$

For $x \in [0,\frac{1}{2}[$, this is an alternating series and we can bound the rest by

$$\begin{aligned} |\sum_{k=L}^{+\infty}(-1)^{k+1}\frac{x^k}{k}| &\leqslant |(-1)^L\frac{x^L}{L}| \\ &\leqslant \frac{1}{2^L L} \end{aligned}$$

For $x \in [-\frac{1}{4},0[$, we can use Lagrange formula to bound the rest by

$$\begin{aligned} \exists \xi \in [-\frac{1}{4},0[, \quad |\sum_{k=L}^{+\infty}(-1)^{k+1}\frac{x^k}{k}| &= |\frac{\log^{(L)}(\xi)}{(L)!}x^L| \\ &= \frac{(L-1)!}{(1+\xi)^L L!}|x^L| \\ &= \frac{1}{(1+\xi)^L L}|x^L| \\ &\leqslant \frac{1}{(1+\xi)^L L}\frac{1}{4^L} \\ &\leqslant \frac{1}{(\frac{3}{4})^L L}\frac{1}{4^L} \\ &= \frac{1}{3^L L} \end{aligned}$$

This shows that we can use the Taylor series up to order $L-1$ to approximate $\ln(x)$ with precision 2^{-L} on the range $[\frac{3}{4},\frac{1}{2}[$. This is to simplify the complexity analysis. In actual implementation though better and faster approximation are used: See the book by Cheney [4] for mathematical fundations of the approximation of functions by polynomials including the Remez algorithm. See also this insightful post [3] which discusses tradeoffs between accuracy and speed in approximating this log function, taking into account error induced by floating-point representation of real numbers. Here [5] can be found an actual implementation of the C standard library.

In addition to the Taylor error, there is an error occuring when computing the polynomial using floating-point arithmetic. If we store the significand of the floating-point variables in binary on L+1 bits, and use

$O(L)$ bits to do arithmetic operations, each operation will incurr a relative error of at most $\epsilon = 2^{-L-1}$, i.e.

$$\begin{aligned}
x \oplus y &= (x+y)(1+\xi)\\
x \ominus y &= (x-y)(1+\xi)\\
x \otimes y &= (x \times y)(1+\xi)\\
x \oslash y &= (x \div y)(1+\xi)
\end{aligned}$$

where $|\xi| \leqslant \epsilon$ and the values on the left are the value computed exactly and then rounded on $L+1$ digits. For the details on floating point arithmetic see [6]. The previous polynomial can be rewritten as:

$$\begin{aligned}
P(x) &= \sum_{k=1}^{n}(-1)^{k+1}\frac{1}{k}x^k\\
&= x(1 + x(-\frac{1}{2} + x(\frac{1}{3} + \cdots + ((-1)^{n-1}\frac{1}{n-1} + (-1)^n\frac{1}{n}x)\ldots)))
\end{aligned}$$

This shows that the evaluation costs n fused multiply-add operations. If one rounding error occurs for each of the multiply-add, we have the following bound on the error due to floating-point arithmetic (see [7] for a detailed analysis)):

$$\begin{aligned}
|\bar{P}(x) - \tilde{P}(x)| &= |\sum_{j=1}^{n-1}(\xi_j\sum_{i=j}^{n}(-1)^{i+1}\frac{1}{i}x^i)|\\
&= |\sum_{i=1}^{n-1}(\sum_{j=1}^{i}\xi_j)(-1)^{i+1}\frac{1}{i}x^i + (\sum_{j=1}^{n-1}\xi_j)(-1)^n\frac{1}{n}x^n|\\
&\leqslant \sum_{i=1}^{n-1}(\sum_{j=1}^{i}\epsilon)\frac{1}{i}|x^i| + (\sum_{j=1}^{n-1}\epsilon)\frac{1}{n}|x^n|\\
&= \epsilon\sum_{i=1}^{n-1}|x^i| + \epsilon\frac{n-1}{n}|x^n|\\
&\leqslant \epsilon\sum_{i=1}^{n}\frac{1}{2^i}\\
&= \epsilon(1 - (\frac{1}{2})^n)\\
&\leqslant \epsilon
\end{aligned}$$

and we add the error due to just storing the coefficients of the polynomial on L bits: for instance $\frac{1}{3} = 0.010101\ldots$ is rounded when storing in binary. If $\tilde{a}_i$ is the rounded value of $a_i = (-1)^i\frac{1}{i}$, the error will be:

$$\begin{aligned}
|P(x) - \tilde{P}(x)| &\leqslant \sum_{i=1}^{n}|a_i - \tilde{a}_i||x^i|\\
&\leqslant \sum_{i=1}^{n}\epsilon|a_i||x^i|\\
&= \epsilon\sum_{i=1}^{n}\frac{1}{i}|x^i|\\
&\leqslant \epsilon\sum_{i=1}^{n}|x^i|\\
&\leqslant \epsilon
\end{aligned}$$

Taking into consideration the three types of error, we see that the taylor series of order L is a approximation to $\ln(x)$ on range $[\frac{3}{4}, \frac{1}{2}[$ with precision 2^{-L} since :

$$\begin{aligned} & \frac{1}{2^{L+1}(L+1)} + 2\epsilon \leqslant 2^{-L} \\ \Leftrightarrow \quad & \frac{2^{-L-1}}{L} + 2^{-L} \leqslant 2^{-L} \\ \Leftrightarrow \quad & 2^{-L-1}(\frac{1}{L} + 1) \leqslant 2^{-L} \\ \Leftrightarrow \quad & L \geqslant 1 \end{aligned}$$

This analysis shows that the procedure Log2 computes an approximation of $\log(N)$ to precision 2^{-L}. Binary addition-substraction costs $\Theta(L)$ operations, grade-school multiplication-division costs $\Theta(L^2)$. Multiplication complexity can be improved to:

- $O(L^{\log_2(3)})$ using Karatsuba algorithm [2].
- $O(L\log(L)\log\log(L))$ using Schönhage-Strassen algorithm [2].
- $O(L\log L\log^* L$ using Furer algorithm [8].

Faster division $x \div y$ consists in computing $\frac{1}{y}$ in $O(\log(L))$ multiplications, then doing $x \div y = x \times \frac{1}{y}$ (cf. [9]).

In the end computing $\log_2 N$ has an $O(L^3)$ time complexity. If we are given an approximating polynomial and are assured it gives the desired precision for any input size considered, the complexity is $O(L^2)$. The complexity of finding $\lfloor \log_2(N) \rfloor$ given the binary representation of N is $O(L)$.

```
Log2($N, L$)
    // $L \geqslant l + 1$ where $l = \lfloor \log_2(N) \rfloor$, i.e. $2^l \leqslant N < 2^{l+1}$.
    for $j = 1$ to $L$
        $A[j] = (-1)^{j+1}\frac{1}{j}$
    $m = \lfloor \log_2(N) \rfloor$                    // $N = 2^m(1+f)$
    $f = \frac{N}{2^m} - 1$                          // no rounding error in $f$.
    if $f \geqslant \frac{1}{2}$                           // map range [1.5, 2[ to [0.75, 1[.
        $f = \frac{1/2-f}{2}$
        $m = m + 1$
    $q = 0$
    for $j = L$ downto $0$
        $q = q \times f + A[j]$
    $q = q \div \ln(2)$
    return q + m
```

18.

$$\begin{aligned} x^2 &= 16 \\ x^3 &= 64 = -27 \\ x^4 &= 108 = 17 \\ x^5 &= 68 = -23 \\ x^6 &= 92 = 1 \end{aligned}$$

shows that the order of x is 6.

19.

1 is not composite and all the odd integers less than 15 are prime except $9 = 3^2$.

20.

Let $l \in [\![0, N-1]\!]$.

$$\begin{aligned}\sum_{x=0}^{N-1} e^{-\frac{2i\pi lx}{N}} f(x) &= \sum_{k=0}^{\frac{N}{r}-1}\sum_{x=0}^{r-1} e^{-\frac{2i\pi l(kr+x)}{N}} f(kr+x)\\ &= \sum_{k=0}^{\frac{N}{r}-1}\sum_{x=0}^{r-1} e^{-\frac{2i\pi lkr}{N}} e^{-\frac{2i\pi lx}{N}} f(x)\\ &= \sum_{x=0}^{r-1}(\sum_{k=0}^{\frac{N}{r}-1} e^{-\frac{2i\pi lkr}{N}})e^{-\frac{2i\pi lx}{N}} f(x)\\ &= \sum_{x=0}^{r-1}(\sum_{k=0}^{\frac{N}{r}-1} e^{-\frac{2i\pi lkr}{N}})e^{-\frac{2i\pi lx}{N}} f(x)\end{aligned}$$

we have

$$\sum_{k=0}^{\frac{N}{r}-1} e^{-\frac{2i\pi lkr}{N}} = \begin{cases} \dfrac{N}{r} & \text{if } \frac{lr}{N} \in \mathbb{N},\\ \dfrac{1-(e^{-\frac{2i\pi lr}{N}})^{\frac{N}{r}}}{1-e^{-\frac{2i\pi lr}{N}}} = 0 & \text{otherwise.}\end{cases}$$

so if $l = l'\frac{N}{r}$, with $l' \in [\![0, r-1]\!]$,

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{-\frac{2i\pi lx}{N}} f(x) = \frac{1}{\sqrt{Nr}}\frac{1}{\sqrt{r}}\sum_{x=0}^{r-1} e^{-\frac{2i\pi l'x}{r}} f(x)$$

otherwise

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{-\frac{2i\pi lx}{N}} f(x) = 0$$

References

[1] Thomas H. Cormen and Charles E. Leiserson : *Introduction to algorithms*, MIT Press (2009)

[2] Aho, Alfred V.;Hopcroft, John E.;Ullman, Jeffrey D.: *The design and analysis of computer algorithms*, Addison-Wesley (1974)

[3] Goldberg, David : *Fast approximate Logarithms*, `https://tech.ebayinc.com/engineering/fast-approximate-logarithms-part-i-the-basics/`.

[4] Cheney, E.W. : *Introduction to approximation theory*, AMS Chelsea publishing (1998).

[5] *musl an implmentation of the standard library for Linux-based systems*, `http://git.musl-libc.org/cgit/musl/tree/src/math/log2.c`.

[6] Goldberg, David : *What every computer scientist should know about floating-point arithmetic*, ACM computing surveys, Vol 23 (1991).

[7] Oliver, J. : *rounding error propagation in polynomial evaluation schemes*, Journal of Computational and applied Mathematics, volume 5, no 2 (1979).

[8] Fürer, Martin : *Faster integer multiplication*, Proceedings of the 39th annual ACM symposium on theory of computing (2007).

[9] *division: multiplicative algorithms*, Advanced Computer Arithmetic EE 486,`https://web.stanford.edu/class/ee486/doc/chap5.pdf`.