Our favorite computational model:

b-bit random access machine (RAM)

2^b working registers $r_0, \dots, r_{2^b-1}$ with values in $\{0, \dots, 2^b-1\}$

2^b input $in_0, \dots, in_{2^b-1}$ — u

2^b output $out_0, \dots, out_{2^b-1}$ — u

A program consists of s steps of the following form:

- "$r_i := y$" $(0 \le i < 2^b)$, $0 \le x < 2^b$, assign register i the value x, then go to the next step.
- "$r_i := r_j \pm r_k$" $(0 \le i, j, k < 2^b)$ undef. if result $\notin \{0, \dots, 2^b-1\}$
- "$r_i := \left\lfloor \frac{r_j}{r_k} \right\rfloor$" — — or $r_k = 0$
- "$r_i := r_j \bmod r_k$" — —
- "go to step t" $(1 \le t \le s)$
- "if $r_i = x$, go to step t, otherwise to step u" — u —
- "if $r_i < r_j$
- "$r_i := r_{r_j}$"
- "$r_{r_i} := r_j$"
- "$r_i := in_{r_j}$"
- "$out_{r_i} := r_j$"
- "halt"

Initially, all registers contain 0 except the appropriate input registers.

After the program halts, the output should be in the output registers.

Running time = total number of steps taken

Memory usage = smallest $m \geq 0$ s.t. only registers r_i, in_i, out_i with $0 \leq i < m$ were used (read from or written to).

Upshot: "n" is the intuitive running time. ("On current computers, often $n = 64$.")

We'll write pseudocode ...

Ex There is a program which adds two binary integers with $\leq n$ digits in time $O(n)$, memory $O(n)$ on an $O(\log n)$-bit RAM.

Input: Encode $\sum_{i=0}^{n-1} a_i 2^i$, $\sum_{i=0}^{n-1} b_i 2^i$ as $n, a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}$

Output: Encode $\sum_{i=0}^{n-1} c_i 2^i$ as $n, c_0, \ldots, c_{n-1}$.

Ex Instead of binary encoding, use base t. $\rightsquigarrow$ can add two integers with $\leq n$ base-t digits in time $O(n)$, memory $O(n)$ on an $O(\max(\log n, \log t))$-RAM

Some other interesting models:

- Turing machines
- Multitape Turing machines
- [redacted] Multiple$^{(k)}$ RAM working in parallel (communicating)

  [redacted] (Running time = max. number of steps taken by any of the RAM

[redacted] (Of course, if we can do sth. in time $O(T)$ on k parallel RAM, we can do it in time $O(kT)$ on one RAM. But the converse doesn't always hold!)

- Quantum RAM

⋮

1. Fast multiplication

Let R be a ring with unit (not necessarily commutative).

We'll assume the ring op. in R can be done in $O(1)$ by augmenting the b-bit RAM:

Registers can take values in $\{0,\dots,2^b-1\} \subset \mathbb{Z}$ $\cup$ R.

The op. "$r_i := r_j \star r_k$" apply if both $r_j, r_k \in \{0,\dots,2^b-1\}$ or both $r_j, r_k \in R$.

There's an op. "$r_i :=$ image of $r_j \in \mathbb{Z}$ under the hom. $\mathbb{Z} \to R$".

Question For large n, how quickly can we multiply two pol. $f, g \in R[X]$ of degree $\le n$ on an $O(\log n)$-bit RAM?

(given coeff. of f, g, find coeff. of $f \cdot g$.)

Idea (too-good-to-be-true?) $\deg(fg) \le 2n$.

$\Rightarrow$ If R is a field, one can reconstruct fg from its value $(fg)(p_i) = f(p_i)\,g(p_i)$ at $2n+1$ (distinct) points $p_0, \dots, p_{2n} \in R$.

But how to compute $f(p_i), g(p_i)$ for $i = 0, \dots, 2n$? (evaluation)

And how to compute fg from these $2n+1$ values? (interpolation)

Idea 2 This is easier for powers $p_i = \omega_k^i$ of a root of unity ω_k (with $k > 2n$).

1.1. Fourier transform

Let $n \geq 1$ and assume that $\zeta_n \in R$ is a root of the n-th cyclotomic polynomial $\Phi_n \in \mathbb{Z}[X]$ (def. recursively by $X^n - 1 = \prod_{d \mid n} \Phi_d(X)$).

(the monic pol.)

("a prim. n-th root of unity")

Lemma 1.1.1 a) $\zeta_n^n = 1$

b) For any $d \mid n$, ζ_n^d is a root of $\Phi_{n/d}$.

c) For any $a \in \mathbb{Z}$,

$$\sum_{i \in \mathbb{Z}/n\mathbb{Z}} \zeta_n^{ai} = \begin{cases} 0, & a \not\equiv 0 \bmod n \\ n, & a \equiv 0 \bmod n. \end{cases}$$

Def The Fourier transform of $a=(a_i)_{i\in\mathbb{Z}/n\mathbb{Z}} \in \prod_i R$ (w.r.t. ζ_n) is $\mathcal{F}_{\zeta_n}(a) = (b_j)_{j\in\mathbb{Z}/n\mathbb{Z}} \in \prod_i R$, where

$$b_j = \sum_{i\in\mathbb{Z}/n\mathbb{Z}} a_i \zeta_n^{ij}.$$

Lemma 1.1.2

$$\mathcal{F}_{\zeta_n}\left(\mathcal{F}_{\zeta_n}(a)\right) = (n\cdot a_{-i})_{i\in\mathbb{Z}/n\mathbb{Z}}.$$

Pf $\mathcal{F}_{\zeta_n}(a)=b$ with $b_j = \sum_i a_i \zeta_n^{ij}$

$$\mathcal{F}_{\zeta_n}(b)=c \text{ with } c_k = \sum_j b_j \zeta_n^{jk} = \sum_{i,j} a_i \underbrace{\zeta_n^{(i+k)j}}_{\xi:=n \text{ if } i+k=0,\ 0 \text{ otherwise}} = n a_{-i}. \quad \square$$

Cor 1.1.3 If n is invertible in R, the problem of evaluating at roots of unity is equiv. to the problem of interpolating from roots of unity.

~~Naïve alg to compute $\mathcal{F}_{\zeta_n}(a)$ (given a, ζ_n) needs time~~

Question Given n, $a=(a_i)_i$, and ζ_n, how quickly can we compute $\mathcal{F}_{\zeta_n}(a)$? **Qol**

$\Rightarrow c_k = n \cdot a_{-k \bmod n}$ ~~where $i \equiv -k \bmod n$~~. □

Thm 1.1.3 (Cooley–Tukey 1965, Gauß)

Let $n = pq$ for integers $p, q \geq 1$. Let $\zeta_p = \zeta_n^q$, $\zeta_q = \zeta_n^p$.

~~If ζ_n is a root of Φ_n, then $\zeta_p := \zeta_n^q$ is a root of Φ_p and $\zeta_q := \zeta_n^p$ is a root of Φ_q.~~

We can reduce the problem of computing a length n FT to computing p length q FT:

Let $a \in R^n$.

For $l = 0, \dots, p-1$, let $b^{(l)} := \mathcal{F}_{\zeta_q}(a_l, a_{p+l}, \dots, a_{(q-1)p+l})$

Then, $\mathcal{F}_{\zeta_n}(a) = (c_j)_{j}$ where $c_j = \sum_{l=0}^{p-1} b_j^{(l)} \zeta_n^{lj}$ for $j \in \mathbb{Z}/n\mathbb{Z}$.

Pf

$$b_j^{(l)} = \sum_{i=0}^{q-1} a_{ip+l}\, \zeta_q^{ij} = \sum_{i=0}^{q-1} a_{ip+l}\, \zeta_n^{ipj}$$

$$\Rightarrow \sum_{l=0}^{p-1} b_j^{(l)} \zeta_n^{lj} = \sum_{l=0}^{p-1} \sum_{i=0}^{q-1} a_{ip+l}\, \zeta_n^{(ip+l)j}$$

$$= \sum_{k \in \mathbb{Z}/n\mathbb{Z}} a_k\, \zeta_n^{kj}$$

(any $k \in \mathbb{Z}/n\mathbb{Z}$ can be written uniquely as $k = ip + l$) □

Cor 1.1.4 (Radix r Cooley-Tukey FFT alg.)

Let $r \geq 2$, $e \geq 0$, $n = r^e$.

Then, you can compute $\mathcal{F}_{\zeta_n}(a)$ with $\mathcal{O}(r^{e+1}(e+1))$
$= \mathcal{O}(n \cdot r \cdot (\log_r n + 1))$

add./mult. op. in R.

(Think of r as fixed, usually $r = 2$. $\rightsquigarrow$ time $\mathcal{O}(n \log n)$ for large n)

Alg ~~Apply the CT Thm. recursively with $p = r$, $q = r^{e-1}$~~

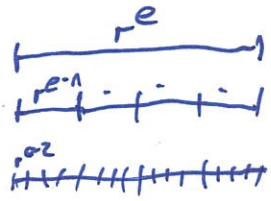
Idea: Apply the C-T Thm. (recursively) with $p = r$, $q = r^{e-1}$.
To compute $\mathcal{F}_{\zeta_{r^e}}(a)$:

1) For $c = 0, \dots, r-1$:
Recursively compute $b^{(c)} = \mathcal{F}_{\zeta_{r^{e-1}}}(a_c, a_{r+c}, \dots, a_{(r^{e-1}-1)r+c})$.
(time $\leq C \cdot r^{e-1} \cdot (e-1)$ by induction)
(time $\leq C \cdot r^{e-1}$)

2) Compute $1, \zeta_n, \dots, \zeta_n^{n-1}$. (time $\mathcal{O}(n) = \mathcal{O}(r^e)$)

3) For $j = 0, \dots, r^{e-1}$:
compute $c_j = \sum_{c=0}^{r-1} b_j^{(c)} \zeta_n^{cj}$. (time $\mathcal{O}(r)$)
(time $\mathcal{O}(r^{e+1})$)

4) Return $\mathcal{F}_{\zeta_{r^e}}(a) = (c_j)_j$.



~~total~~ time
$C r^{e+1} \cdot e + \mathcal{O}(r^{e+1})$
$\leq C r^{e+1}(e+1)$
if $C \geq$ the constant in $\mathcal{O}(\dots)$.

□

Rmks

All multiplications in R performed in the alg. are mult. by powers of S_n.