Fast integer multiplication

- We improved polynomial multiplication by using the viewpoint: $f(x)$ is a function that takes values & can be reconstructed from them.

- We cannot do the same for integers.

- How do we multiply integers differently?
  By reducing them to polynomials!

- Say, a, b & $ab \in \mathbb{N}$ are $\ell = 2^n$ bit numbers.

- Let $k := 2^{\lceil n/2 \rceil}$, $m := 2^{\lfloor n/2 \rfloor}$. So, $km = \ell$.

$\triangleright$ $k, m \in \sqrt{\ell}$.

- Idea: Reduce the $a \cdot b$ computation to the multiplication of polynomials $\hat{a}(x) \cdot \hat{b}(x)$, where $\hat{a}, \hat{b}$ are of degree $< m$.

- Write a as $\sum_{i=0}^{m-1} \hat{a}_i \cdot 2^{ki}$, $\hat{a}_i \in \mathbb{N}$

and b as $\sum_{i=0}^{m-1} \hat{b}_i \cdot 2^{ki}$, $\hat{b}_i \in \mathbb{N}$.

▷ Clearly, $\hat{a}_i, \hat{b}_i < 2^k$.

- Define integral polynomials,

$$\hat{a}(x) := \sum_{0 \leq i \leq m-1} \hat{a}_i \cdot x^i \quad \&$$

$$\hat{b}(x) := \sum_{i} \hat{b}_i \cdot x^i .$$

▷ $\hat{a}, \hat{b}$ have degree $<m$ & their coefficients are k-bits.

▷ The coefficient of x^j in $\hat{a}(x) \cdot \hat{b}(x)$ is $\sum_{0 \leq i \leq j} \hat{a}_i \cdot \hat{b}_{j-i}$, whose magnitude $< m 2^{2k} < 2^{3k}$.

- Idea: We compute the polynomial product $\hat{a}(x)\cdot\hat{b}(x) \pmod{2^{3k}+1}$. Finally, compute at $x=2^k$. It would suffice to work mod $m(2^{2k}+1)$, but we're giving a slower algo.

▷ The ring $R := \mathbb{Z}/\langle 2^{3k}+1\rangle$ has a $(2k)$-th root of unity $\omega := 8$. (Note: $2k > m$)

- So, we can follow the polynomial multiplication algorithm based on DFT$[\omega]$:

$\hat{a}\cdot\hat{b}$ has deg $< m$

(1) Compute DFT$[\omega]$ of $\hat{a}(x)$ & $\hat{b}(x)$ in $R[x]$.
(2) Compute the m products $\hat{a}(\omega^i)\cdot\hat{b}(\omega^i)$ in R.
(3) Compute DFT$[\omega^{-1}]$ of $\{\hat{c}(\omega^i) \mid i\}$, yielding $\hat{c}(x)$.
(4) Output $\hat{c}(2^k)$.

Complexity analysis: (Think of integers in R as k-digit base-8.)

Steps (1) & (3): By the fast DFT algorithm it can be done in $O(m\lg m)$ R-operations, which means $O(km\lg m)$ time. (R-mult. needed are the ones by ω^i.)

Step(2): We need to do m multiplications of $(3k)$-bit numbers. So, we get the recurrence:

$$T(\ell) = m \cdot T(3k) + O(\ell \cdot \lg m).$$

$\Rightarrow$

$$T(\ell) = O(\ell \cdot \lg^{\alpha} \ell), \text{ where } \alpha := \log_2 \tfrac{3}{2}.$$

▷ The extra factor we get is $(3/2)^{\lg\lg\ell} = \lg^{\alpha}\ell$.

Step(4): This is simply rewriting the number in bits. So, time is $O(\ell)$.

<u>Theorem</u>: Two ℓ-bit integers can be multiplied in $\tilde{O}(\ell) = O(\ell \cdot \lg^{\alpha}\ell)$ time, $\alpha := \log_2 3/2$.

Schönhage-Strassen(1971) takes $O(\ell \cdot \lg\ell \cdot \lg\lg\ell)$-time.

- The fastest integer multiplication algorithm known is due to Fürer (2007).

  Its time complexity is $\ell \cdot \lg\ell \cdot 2^{O(\lg^* \ell)}$, where $\lg^*\ell$ is defined to be the least number i s.t. $\underbrace{\lg\lg\cdots\lg}_{i \text{ times}}\ell < 2$

- The idea is to use multivariates & a much smaller R.