Fast integer division

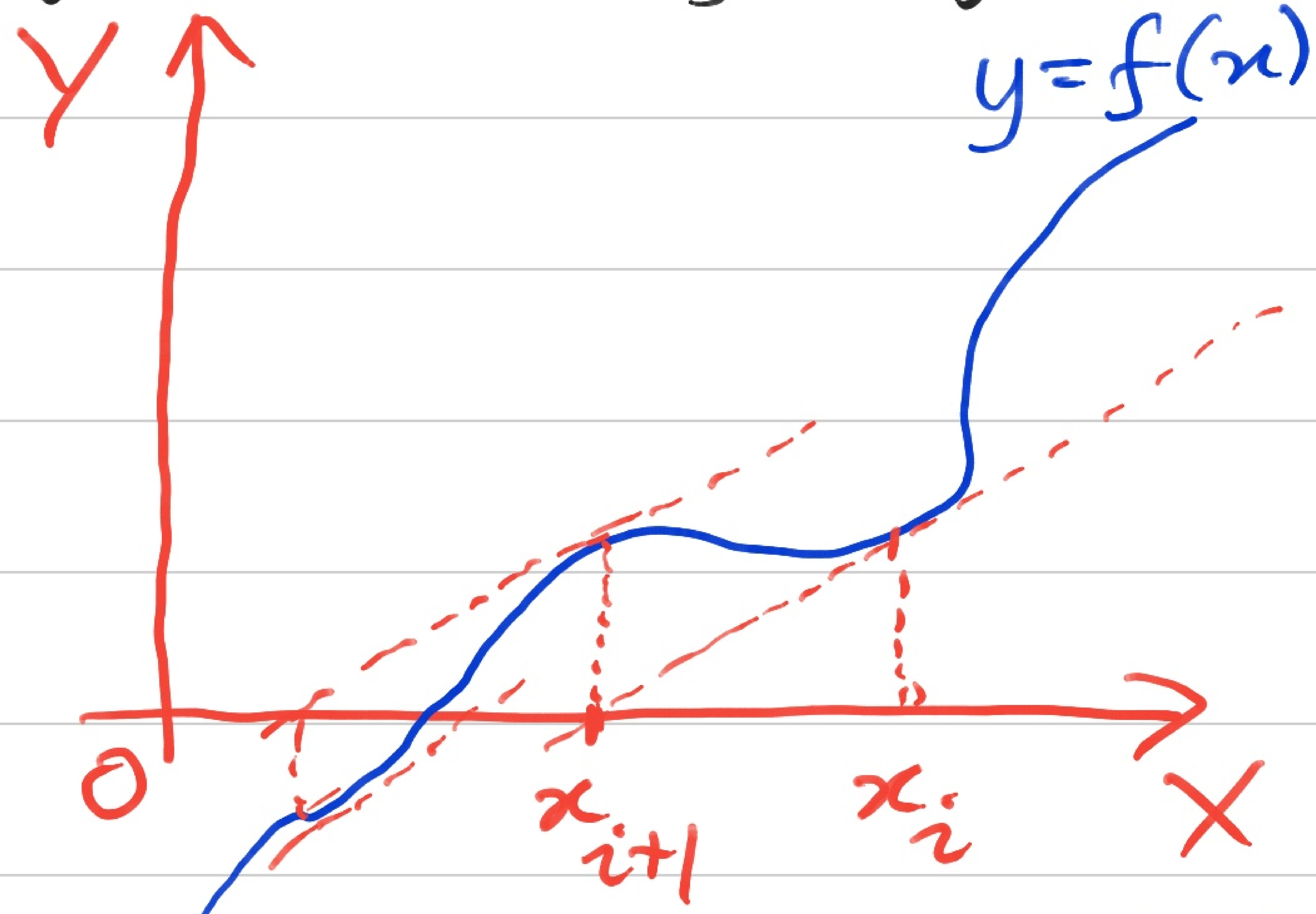
- The qn. of computing a/b up to some decimal places, reduces to that of computing $1/b$.
- If b is ℓ-bits & we want $1/b$ up to ℓ places. School-method takes $O(\ell^2)$-time.
- Could we make it $\tilde{O}(\ell)$ using fast integer multiplication?

Newton's Approximation (1685)

- It's an iterative way to find roots of a function $y = f(x)$.

▷ $$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \text{---} (1)$$

- Start with $(x_0, 0)$ & get closer to a root x_* of f.



$$f'(x_i) = \frac{y - f(x_i)}{x - x_i}$$

- Newton's algo: 1) Compute $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

  for $i = 0, 1, 2, \ldots$

  to get $|f(x_i)|$ "small".

  2) Output $x_0, x_1, x_2, \ldots$ as <u>approximations to a root</u> of f.

- For integer division the relevant curve is:

  $$y = f(x) := \underline{\frac{1}{x} - b} \, .$$

  $\triangleright$ $f'(x) = -1/x^2$.

$\triangleright$ $\underline{x_{i+1}} = x_i - \dfrac{x_i^{-1} - b}{-x_i^{-2}} = x_i(2 - bx_i)$

$\underline{y_{i+1}} := 1 - bx_{i+1} = 1 - bx_i(2 - bx_i) = y_i^2$

$= x_i + \underline{x_i}\,\underbrace{(1 - bx_i)}_{y_i}$

maintain their 2^i places

- Let $x_0 := 2^{-l}$ & $2^{l-1} \leq b < 2^l$.

Lemma: $\forall i \geq 0$, $|\underline{x_i - b^{-1}}| \leq 1/b \cdot 2^{2^i}$.

Pf: $[i=0]$: $|x_0 - b^{-1}| = \left|\frac{1}{2^l} - \frac{1}{b}\right| = \dfrac{|b - 2^l|}{b\,2^l} \leq \dfrac{2^{l-1}}{b \cdot 2^l}$

- Let it hold up to i.

$\left|x_{i+1} - \frac{1}{b}\right| = \left|2x_i - bx_i^2 - \frac{1}{b}\right| = b \cdot \left|x_i - \frac{1}{b}\right|^2 \leq b \cdot \dfrac{1}{b^2 2^{2^{i+1}}} = \dfrac{1}{b \cdot 2^{2^{i+1}}}$ $\square$

$\Rightarrow$ To know $1/b$ up to ℓ places, it suffices to iterate up to $i = O(\lg \ell)$.

<u>Complexity analysis</u>: Let <u>$M(m)$</u> be the time taken to multiply two m-bit integers. Then, computing b^{-1} (up to <u>ℓ</u> places) takes:

$$\Rightarrow \sum_{i=1}^{\lg \ell} M(2^i) \leq M\left(\sum_{i=1}^{\lg \ell} 2^i\right) \leq M(2\ell) = \tilde{O}(\ell).$$

[super-linear $M(\cdot)$]

▷ Division in $M(\lg a)$-time [for a/b].

- Recall gcd computation — Its j-th step is
$r_{j-2} - q_j r_{j-1} = r_j$; with $r_{-1} = a$ & $r_0 = b$.
$[|a| > |b|]$

$\Rightarrow$ The time complexity of gcd is:

$$\sum_{1 \le j \le i} M(\lg q_j) \le M\left(\sum_{j=1}^{i} \lg q_j\right)$$

$$\le M\left(\sum_{j=1}^{i} (\lg r_{j-2} - \lg r_{j-1})\right) \le M(\lg a)$$

$$\le \tilde{O}(\lg a)\text{-time.}$$

▷ $\gcd(a,b)$, $a^{-1} \bmod b$ & CR are doable in $O(M(\lg a))$-time.

Revisit Integer Multiplication

- **Input:** a & b of $\ell = 2^n$ bits.

- Recall $\hat{a}(x)$, $\hat{b}(x)$ are polynomials of $\deg < m$.
  $\left[\, m := 2^{\lfloor n/2 \rfloor},\ k := 2^{\lceil n/2 \rceil} \,\right]$

$\Rightarrow$ Coeffs. of $\hat{a}$, $\hat{b}$ are $< 2^k$.

$\Rightarrow$ coeffs. of $\hat{a} \cdot \hat{b}$ are $< 2^{2k} \cdot m < 8^k$

- Instead work over $R := \mathbb{Z} / \langle m \cdot (4^k + 1) \rangle$

▷ $\omega := 4 \bmod \langle 4^k + 1 \rangle$ has order $2k > m$

▷ $\gcd(m, 4^k + 1) = 1$.

$\Rightarrow$ arithmetic over R $\overset{[CR]}{\equiv}$ arithmetic mod m

& " " $\langle 4^k+1 \rangle$

▷ $\hat{a}(x) \cdot \hat{b}(x) \bmod m$ can be computed in $O(\ell)$-time. [use basic algorithms]

▷ $\hat{a} \cdot \hat{b} \bmod \langle 4^k+1 \rangle$, via DFT[$\omega$], is recursion based computation. Gives recurrence:

$$T(\ell) \leq O(\ell) + m \cdot T(2k) + O(\ell \cdot \lg \ell)$$

$$\Rightarrow \quad T'(\ell) \leq 2 \cdot T'(2k) + O(\lg \ell) \quad [T'(\ell) := T(\ell)/\ell]$$

$$\Rightarrow T'(\ell) = O(\lg \ell \cdot \lg\lg \ell).$$