

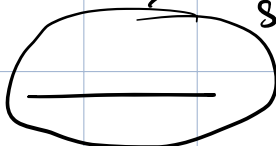
# Analysis

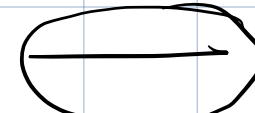
→ Correctness

—  $x[i]$  —

—  $y[i]$  —

$$x[i] < y[i]$$

—  $x[i]$  —  already sorted

—  $y[i]$  — 

$$x[i] = y[i]$$

Running time:

If the input consists of  $n$  strings, each of length  $k$ ,  
then running time =  $\Theta(kn)$

Numbers

If input consists of  $n$  strings, each of  $d$  digits  
in base  $b$ , then run time =  $\Theta(d(b+n))$

$d$  phases:

each phase: count sort of  $n$  numbers in  $0, \dots, b-1$

Sorting  $n$  numbers in  $\{1, \dots, n^2\}$  in  $O(n)$  time  
each number in base  $n$

# digits = 2

Applying radix sort:  $O(2(n+n)) = O(n)$

$$x \in \{1, \dots, n^2\} = (a, b)$$

can be written as

$$x = an + b,$$

$$a \leq n \quad 0 \leq b \leq n - 1$$

## Solutions

6. Counting inversions

30 Jan 2025

### Polynomial Multiplication

$$\text{I/P : } f(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$$

$$g(x) = b_0 + b_1 x + \dots + b_{n-1} x^{n-1}$$

$$f : [a_0, a_1, \dots, a_{n-1}]$$

$$\text{O/P : } f(x) \cdot g(x) = \sum_{i=0}^{2n-2} c_i x^i$$

$$c_i = \sum_{j=0}^i a_j b_{i-j}$$

To find  $c_i$  :  $(i+1)$  multiplications +  $i$  additions

$$\text{Time} = \sum_{i=0}^{2n-1} (2i+1) = O(n^2)$$

Goal : Improve to  $O(n \log n)$

$$n = 2 : \quad \begin{aligned} f &= a_0 + a_1 x \\ g &= b_0 + b_1 x \end{aligned}$$

Similar to  
integer multipli-  
cation

$$\text{coeff of } x : (a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1$$

We can find  $f \cdot g$  using 3 multiplications.

$$\left. \begin{array}{l} f(0) = a_0 \quad ; \quad g(0) = b_0 \\ f(1) = a_0 + a_1 \quad ; \quad g(1) = b_0 + b_1 \end{array} \right\} \begin{array}{l} \text{coeff of } x \\ \text{in } f \cdot g = f(1) \cdot g(1) - f(0) \cdot g(0) \\ \quad - (f(1) - f(0)) \cdot (g(1) - g(0)) \end{array}$$

$$f(-1) = a_0 - a_1 \quad ; \quad g(-1) = b_0 - b_1$$

$$f(x) \cdot g(x) = c_0 + c_1 x + c_2 x^2$$

$$\left. \begin{array}{l} c_0 = f(0) \cdot g(0) \\ c_1 + c_2 + c_3 = f(1) \cdot g(1) \\ c_0 - c_1 + c_2 = f(-1) \cdot g(-1) \end{array} \right\} \begin{array}{l} \text{System of linear} \\ \text{equations} \end{array}$$

$$h(x) = c_0 + c_1 x + \dots + c_d x^d$$

If we know  $h(x_0)$ ,  $h(x_1)$ ,  $\dots$ ,  $h(x_d)$ , we can find  $c_0, c_1, \dots, c_d$ .

---

$$h(x) = c_0 + c_1 x + c_2 x^2$$

$$h(0) = c_0$$

$$h(1) = c_0 + c_1 + c_2$$

$$h(2) = c_0 + 2c_1 + 4c_2$$

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^d \\ 1 & x_1 & x_1^2 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \dots & x_d^d \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{bmatrix}$$

$\curvearrowright$  invertible matrix  
 $h(x_i) = y_i$

Step 1: Find  $f(x_1), f(x_2), \dots, f(x_d)$   
 $g(x_1), g(x_2), \dots, g(x_d)$

If  $\deg(f) = \deg(g)$   
 $= n-1$ ,  
 Then we need  
 $d = 2n - 1$   
 to get all coefficients



Step 2: Find  $f(x_1) \cdot g(x_1)$ ,  $f(x_2) \cdot g(x_2)$ , ...,  $f(x_d) \cdot g(x_d)$

To find  $f(x_1) \rightarrow O(n)$  time ??  $\rightarrow a_0 + a_1 x + a_2 x^2$

Step 1 :  $O(n^2)$

Step 2 :  $O(n)$

$$\begin{array}{r} a_0 + x(a_1 + x(a_2)) \\ \hline a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ a_0 + x(a_1 + x(a_2 + x(a_3))) \end{array}$$

# Discrete Fourier Transform

$$\text{DFT} : \mathbb{C}^n \rightarrow \mathbb{C}^n$$

$$\text{DFT}_n (a_0, a_1, \dots, a_{n-1}) \begin{cases} a_0 + a_1 x \\ + \dots + a_{n-1} x^{n-1} \end{cases}$$

$$= (f(1), f(\omega_n), \dots, f(\omega_n^{n-1}))$$

$$\omega_n = e^{2\pi i/n}$$

$$\text{DFT}_3 (1, -2, 4)$$

$$= (3, 1 - 2\omega + 4\omega^2, 1 - 2\omega^2 + \omega)$$

$$\left. \begin{array}{l} 1 - 2x + 4x^2 \\ \omega_3 = \omega \end{array} \right\}$$

$$\omega_{-n} = e^{-2\pi i/n}$$

$$\text{DFT}_{-n} (a_0, a_1, \dots, a_n) = (f(1), f(\omega_{-n}), \dots, f(\omega_{-n}^{n-1}))$$

$$\text{If } \text{DFT}_n (a_0, a_1, \dots, a_{n-1}) = (b_0, b_1, \dots, b_{n-1})$$

$$\text{then } \frac{1}{n} \text{DFT}_{-n} (b_0, b_1, \dots, b_{n-1}) = (a_0, a_1, \dots, a_{n-1})$$

\* We can find  $\text{DFT}_n (a_0, \dots, a_{n-1})$  in  $O(n \log n)$  time.

$$f(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$$

$$= (a_0 + a_2 x^2 + a_4 x^4 + \dots)$$

$$+ x (a_1 + a_3 x^2 + a_5 x^4 + \dots)$$

$$f(x) = f_{\text{even}}(x) + x \cdot f_{\text{odd}}(x)$$

$$\left. \begin{aligned} f_{\text{even}}(y) &= a_0 + a_2 y + a_4 y^2 + \dots \\ f_{\text{odd}}(y) &= a_1 + a_3 y + a_5 y^2 + \dots \end{aligned} \right\} \begin{array}{l} \text{degrees} \\ \leq n/2 \end{array}$$

$$f(1), f(\omega_n), f(\omega_n^2), \dots, f(\omega_n^{n-2})$$

$$f(x) = f_{\text{even}}(x^2) + x \cdot f_{\text{odd}}(x^2)$$

Recursively find

$$\text{DFT}_{n/2}(f_{\text{even}}), \text{DFT}_{n/2}(f_{\text{odd}})$$

$$\text{DFT}_{n/2} : \text{feven} (1) , \text{feven} \left( \frac{w_n}{2} \right) , \dots , \text{feven} \left( \frac{w_n^n}{2} \right)$$

(feven)



$$\text{Need} : \text{feven} (1) , \text{feven} (w_n^2) , \text{feven} (w_n^4) , \dots$$

$$w_{n/2} = e^{\frac{2\pi i}{n/2}} = e^{\frac{4\pi i}{n}} = w_n^2$$

$$w_{n/2}^k = w_n^{2k}$$

Given  $f$  of degree  $n - 1$ ,

① recursively find  $DFT_{\frac{n}{2}}(\text{even})$ ,  $DFT_{\frac{n}{2}}(\text{odd})$ ,

??

② Find  $DFT_n(f)$  using  $n$  additions

$$T(n) = T(n/2) + O(n)$$

$$T(n) = O(n \log n)$$

Fast Fourier Transform  $\rightarrow$  name of the algorithm to find DFT

$$f(x) = 1 + x + 2x^2 - 4x^3$$

$$g(x) = 2 - 3x + x^2 + x^3$$

$$\text{deg (prod)} = 6$$

take a (power of 2)  $\geq 6$

$$f(1), f(\omega_8^1), f(\omega_8^2), \dots, f(\omega_8^7)$$

$$g(1), g(\omega_8^1), g(\omega_8^2), \dots, g(\omega_8^7)$$

Convenience:  
work with powers  
of 2.

$$h(x) = f(x) \cdot g(x)$$

$$\text{Find: } h(1), h(\omega_8^1), h(\omega_8^2), \dots, h(\omega_8^7)$$

$$\text{Coefficients of } h(x) = \frac{1}{8} \text{DFT}_{-8} [h(1), h(\omega_8^1), \dots, h(\omega_8^7)]$$