COMPLEXITY

LECTURE 2

April 25, 2022

Brom WESTERBAAN

Institute for Computing and Information Sciences
Radboud University Nijmegen

NWI-IBC028

Previously...

Recursive Algorithm $\xrightarrow{\text{analyse}}$ Recurrence Relation $\xrightarrow{\text{solving using e.g., MASTER THEOREM}}$ Asymptotic Complexity

Merge Sort $\rightarrow$ $T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n$ $\rightarrow$ $T(n) = \Theta(n \log(n))$

THIS WEEK

- Walk through an example: Karatsuba Multiplication
- Solving recurrence relations with and without the Master Theorem

B R E A K

Fast Fourier Transform (one of the most important and widely used algorithms.)

COMPLEXITY OF MULTIPLICATION ?

Complexity of Multiplication?

- constant time, $\Theta(1)$, for e.g., int64's
- But what about "Big int"s? (E.g., array of uint64 with bool for sign)

Complexity of Multiplication?

- constant time, $\Theta(1)$, for e.g., int64's
- But what about "Big int"s? (E.g., array of uint64 with bool for sign)
  - Kolmogorov conjectured < 1960: $\Omega(n^2)$, where "n" is the number of digits in the largest factor.

Complexity of Multiplication?

- constant time, $\Theta(1)$, for e.g., int64's

- But what about "Big int"s? (E.g., array of uint64 with bool for sign)

  - Kolmogorov conjectured < 1956: ~~$\Omega(n^2)$~~,
    where "n" is the number of digits (or "limbs") in the largest factor.

  - Karatsuba (just 23 y.o.!), in 1960, $\Theta(n^{\log_2(3)})$
    Still used! (E.g. in golang's math.big.Int)

  (- Current <u>theoretical</u> best:
    Harvey and van der Hoeven, 2021: $O(n \log n)$)

KARATSUBA'S TRICK

Given $a, b \in \mathbb{N}$, and $B \in \{2, 3, \dots\}$, write

$$a = a_1 B + a_0 \qquad b = b_1 B + b_0. \qquad a_0, b_0 < B$$

Then:

$$a \cdot b = a_1 \cdot b_1 B^2 + (a_0 \cdot b_1 + b_0 \cdot a_1) B + a_0 \cdot b_0$$

4 multiplications of numbers at best ½ the size

KARATSUBA'S TRICK

Given $a, b \in \mathbb{N}$, and $B \in \{2, 3, \dots\}$, write

$$a = a_1 B + a_0 \qquad b = b_1 B + b_0. \qquad a_0, b_0 < B$$

Then:

$$a \cdot b = a_1 \cdot b_1 B^2 + (a_0 \cdot b_1 + b_0 \cdot a_1) B + a_0 \cdot b_0$$

4 multiplications of numbers at best ½ the size

But also:

$$a \cdot b = a_1 \cdot b_1 B^2 + ((a_0 + a_1) \cdot (b_0 + b_1) - a_1 \cdot b_1 - a_0 \cdot b_0) B + a_0 \cdot b_0$$

just 3 multiplications, since $a_0 \cdot b_0$ and $a_1 \cdot b_1$ can be reused!

(but more additions)

EXAMPLE

$$\overbrace{1}^{a_1}\overbrace{23}^{a_0} \cdot \overbrace{45}^{b_1}\overbrace{67}^{b_0} = 1\cdot 45\cdot 100^2 + \big((23+1)\cdot(45+67) - 1\cdot 45 - 23\cdot 67\big)\cdot 100 + 23\cdot 67$$

$$= 45\cdot 100^2 + (24\cdot 112 - 45 - 23\cdot 67)\cdot 100 + 23\cdot 67$$

$$24\cdot 112 = 22\cdot 10^2 + (\underbrace{6\cdot 13}_{=78} - 22 - 8)\cdot 10 + 8$$

$$= 2200 + 480 + 8 = 2688$$

$$23\cdot 67 = 12\cdot 10^2 + (\underbrace{5\cdot 13}_{=65} - 12 - 21)\cdot 10 + 21$$

$$= 1200 + 320 + 21 = 1541$$

$$= 45\cdot 100^2 + (\underbrace{2688 - 45 - 1541}_{1102})\cdot 100 + 1541$$

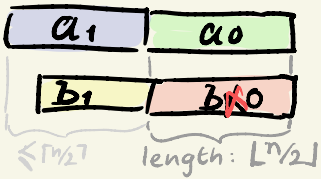
$$= \begin{array}{r} 450000 \\ 110200 \\ 1541\,+ \\ \hline 561741 \end{array}$$

KARATSUBA(a, b):

Let $n = \max(\text{length}(a), \text{length}(b))$

if $n = 1$: return ORDINARYMULTIPLICATION(a, b)

Write



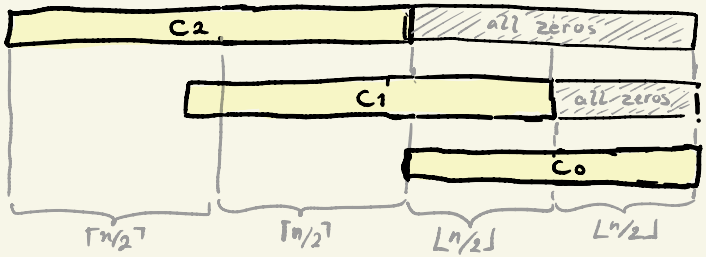
Let $c_2 = \text{KARATSUBA}(a_1, b_1)$

Let $c_0 = \text{KARATSUBA}(a_0, b_0)$

Let $c_1 = \text{KARATSUBA}(a_0 + a_1, b_0 + b_1)$

return

The sum of:



COMPLEXITY ANALYSIS

KARATSUBA(a, b):

Let $n = \max(\text{length}(a), \text{length}(b))$ $\quad \Theta(1)$

if $n = 1$: return ORDINARY MULTIPLICATION(a, b) $\quad \Theta(1)$

Write

$$a = \boxed{a_1}\boxed{a_0}$$
$$b = \boxed{b_1}\boxed{b_0}$$

$\lceil n/2 \rceil$ $\quad$ length: $\lfloor n/2 \rfloor$

$\Theta(n)$

Let $c_2 = \text{KARATSUBA}(a_1, b_1)$ $\quad T(\lceil n/2 \rceil)$

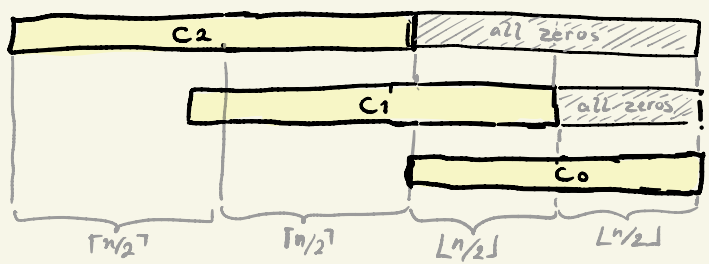
Let $c_0 = \text{KARATSUBA}(a_0, b_0)$ $\quad T(\lfloor n/2 \rfloor)$

Let $c_1 = \text{KARATSUBA}(a_0 + a_1, b_0 + b_1)$ $\quad T(\lceil n/2 \rceil + 1)$

$T(\lceil n/2 \rceil)$
$T(\lfloor n/2 \rfloor)$
$T(\lceil n/2 \rceil + 1)$

return

The sum of:



$\Theta(n)$

KARATSUBA MULTIPLICATION COMPLEXITY

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n)$$

Master Theorem?

SOLVING $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n)$

$$T(n) \leq 3T(\lceil n/2 \rceil + 1) + \Theta(n)$$

$$\Downarrow$$

$$T(n) = 3T(\lceil n/2 \rceil + 1) + O(n)$$

because $f \leq g \in O(h) \implies f \in O(h)$

The Master Theorem still does not apply to T,

but it will to some $S_d(n) := T(n+2d)$ where $d \in \mathbb{Z}$.

$$\begin{aligned} S_d(n) &= 3T(\lceil n/2 \rceil + d + 1) + O(n) \\ &\leq 3T(\lceil n/2 \rceil + 2d) + O(n) \qquad \text{if } 2d \geq d+1 \iff d \geq 1 \\ &= 3S_d(\lceil n/2 \rceil) + O(n). \end{aligned}$$

Thus, when $d \geq 1$,

$$S_d(n) = 3S_d(\lceil n/2 \rceil) + O(n)$$

THUS: $S_d = \Theta(n^{\log_2(3)})$

$\log_2(3) > 1$

SOLVING $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n)$

Since $T(n+2d) = \Theta(n^{\log_2(3)})$,

we get
$$T(n) = T((n-2d)+2d)$$
$$= \Theta((n-2d)^{\log_2 3}) \underset{*}{=} \Theta(n^{\log_2 3}).$$

Computing whether $f(n) = O(g(n))$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \implies f(n) = O(g(n))$$

$$\not\Longleftarrow$$

(and $g(n) = \Omega(f(n))$)

EXAMPLES:

1. $(n+d)^c \in O(n^c)$

$$\lim_{n\to\infty} \frac{(n+d)^c}{n^c} = \lim_{n\to\infty} \left(\frac{n+d}{n}\right)^c$$

$$= \lim_{n\to\infty} \left(1 + \frac{d}{n}\right)^c = 1^c = 1 < \infty$$

since $\frac{d}{n} \xrightarrow{\text{as } n\to\infty} 0$

2.

A "Polynomial" MASTER THEOREM

Let $a \in [1,\infty)$ and $b \in (1,\infty)$ and $T : \mathbb{N} \to [0,\infty)$ be given. Write $\gamma := \log_b(a)$.

Case I $\quad T(n) = \Theta(n^\gamma) \quad \Longleftarrow \quad T(n) = aT(\lceil n/b \rceil) + O(n^d)$ for some $d < \gamma$

(or just $\leq$)

Case II $\quad T(n) = \Theta(n^\gamma \log(n)) \quad \Longleftarrow \quad T(n) = aT(n/b) + \Theta(n^\gamma)$

i.e. $aT(\lfloor n/b \rfloor) + f(n) \leq T(n) \leq aT(\lceil n/b \rceil) + f(n)$ for some $f \in \Theta(n^\gamma)$

Case III $\quad T(n) = \Theta(n^d) \quad \Longleftarrow \quad T(n) = aT(\lfloor n/b \rfloor) + \Theta(n^d)$ for some $d > \gamma$

MASTER THEOREM from the book

Basically the same as the "Polynomial" Master Theorem, except for case III:

"polynomial" Case III $T(n) = \Theta(n^d)$ $\Longleftarrow$ $T(n) = aT(\lfloor n/b \rfloor) + \Theta(n^d)$ for some $d > \gamma$

Becomes:

Case III $T(n) = \Theta(f(n))$ $\Longleftarrow$ $\begin{cases} T(n) = aT(\lfloor n/b \rfloor) + f(n) \\ \text{for some } f(n) = \Omega(n^d) \\ \text{for some } d > \gamma = \log_b(a) \\ \text{provided } f \text{ is regular:} \\ \exists c<1, N\ \forall n \geq N \left[a f(\lceil n/b \rceil) \leq c f(n) \right] \end{cases}$

"Polynomial" Case III can't deal with:

$T(n) = T(\lceil n/2 \rceil) + n \log(n)$

Beyond Master Theorem (§4.3 and §4.4)

$$T(n) = n + T(\lceil n/2 \rceil) + 2T(\lfloor n/5 \rfloor) \qquad T = \Theta(n)$$

Guess: $\Theta(n) \iff O(n) \cap \Omega(n)$

Find N, and $C > 0$ s.t.

$\forall n \geq N \quad T(n) \leq Cn$ by strong induction

Let $n \in \mathbb{N}$ be given such that

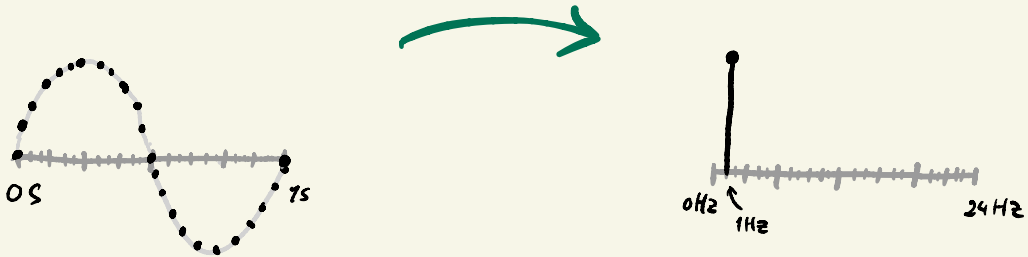
$\forall m < n \quad T(m) \leq Cm$. $\qquad ?\ T(n) \leq Cn$

$N = 20$

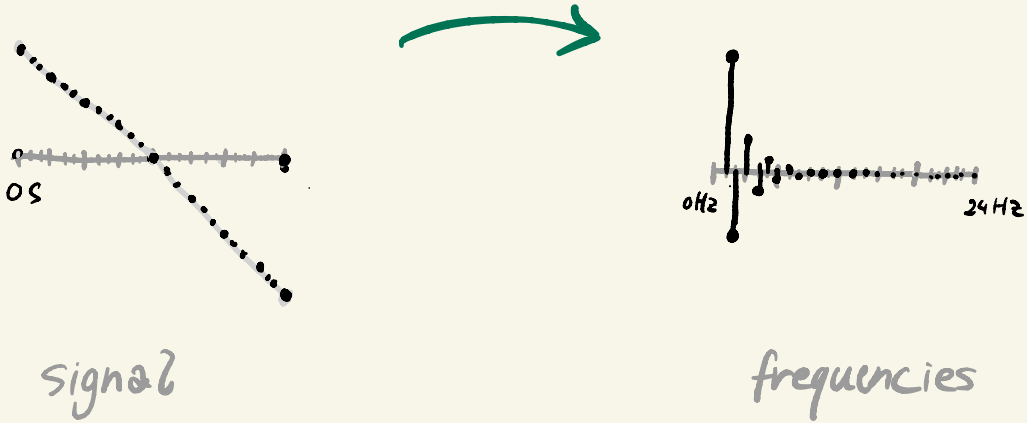
$C = 20 \vee T(0) \vee T(1) \vee \dots \vee T(20)$

$$\begin{aligned}
T(n) &= n + T(\lceil n/2 \rceil) + 2T(\lfloor n/5 \rfloor) \\
&\leq n + C\lceil n/2 \rceil + 2C\lfloor n/5 \rfloor \\
&\leq C\left(n/C + \lceil n/2 \rceil + 2\lfloor n/5 \rfloor\right) \\
&\leq C\left(n/C + n/2 + 1 + 2\, n/5\right) \\
&= C\, n\Big(\underbrace{\tfrac{1}{C}}_{\leq \frac{1}{20}} + 9/10 + \underbrace{\tfrac{1}{n}}_{\leq 1/20}\Big) \leq Cn \quad \text{if } \begin{matrix} C \geq 20 \\ n \geq 20 \end{matrix}
\end{aligned}$$

Fourier Transform (§30.2)



Fourier Transform



APPLICATIONS :

- sound compression (mp3) (throw away frequencies we can't hear)
- image compression (using 2D-waves, JPEG, MPEG)
- noise filtering (remove 50 Hz)
- autotune

  ⋮

- fast polynomial multiplication ← we'll show this!
- fast integer multiplication
- solving differential equations ($\frac{d}{dx} \longleftrightarrow x\cdot$)
- factoring integers into prime numbers in polynomial time on a quantum computer using Shor's algorithm (breaking RSA encryption.)

Before we start, we need some trigonometry...

Recall that $\sin(2\theta) = 2\sin\theta\cos\theta$

$$\cos(2\theta) = 1 - 2\sin^2\theta$$

$$\vdots$$

~~Before we start, we need some trigonometry...~~

~~Recall that $\sin(2\theta) = 2\sin\theta\cos\theta$~~

~~$\cos(2\theta) = 1 - 2\sin^2\theta$~~

~~$\vdots$~~

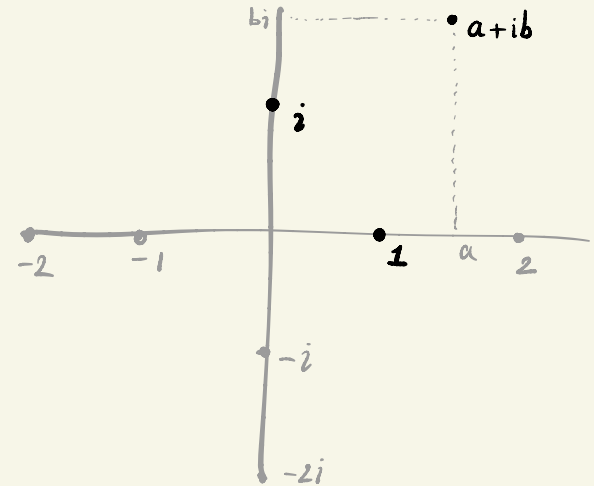
We only need Complex Numbers

Complex Numbers

... are just pairs of real numbers,

$$\mathbb{C} = \{ a + ib : \quad a, b \in \mathbb{R} \}$$

with a new constant i.



Addition is pointwise,

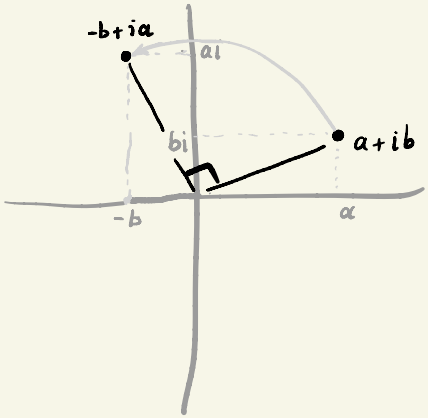
$$a+ib + c+id = (a+c) + i(b+d).$$

Multiplication is fixed by the rule: $i^2 = -1$

$$(a+ib)(c+id) = ac + i(bc+ad) + i^2 bd$$
$$= ac - bd + i(bc+ad)$$

MULTIPLICATION BY i

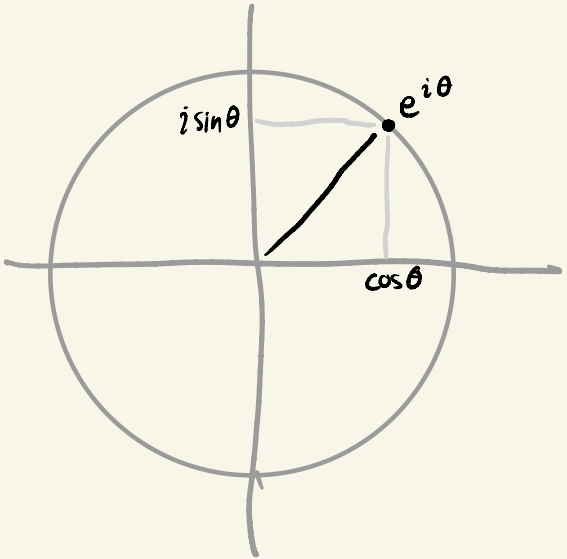
$$i(a+ib) = -b + ia$$



Multiplication by i
is rotation over $90°$!

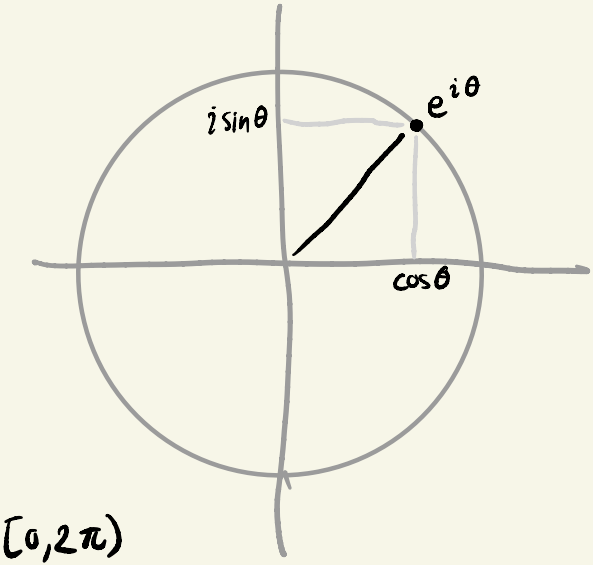
EULER'S FORMULA

$$e^{i\theta} = \cos\theta + i\sin\theta$$



EULER'S FORMULA

$$e^{i\theta} = \cos\theta + i\sin\theta$$



Consequences:

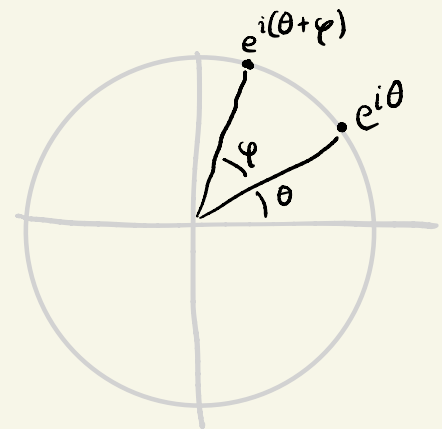
- Any $z \in \mathbb{C}$ can be written

  $z \equiv r e^{i\theta}$ for some $r \in [0,\infty)$, $\theta \in [0, 2\pi)$

- Multiplication by $z \equiv re^{i\theta}$ is

  $$(re^{i\theta})(se^{i\varphi}) = rs\, e^{i(\theta+\varphi)}$$

  scaling by r and rotation by θ.



MORAL:

While a real number represents just scale,
a complex number represents scale and rotation.

Some notation:

$\mathrm{Re}(a+ib) = a$ real part

$\mathrm{Im}(a+ib) = b$ imaginary part

$|a+ib| = \sqrt{a^2+b^2}$ length

'Pure tones'

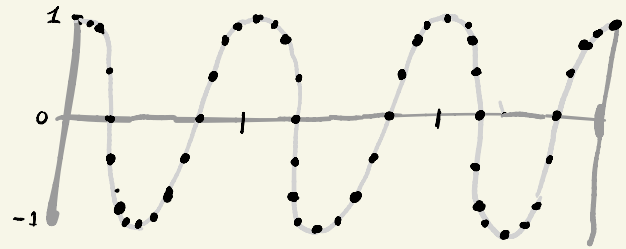
Are represented by:

$$e^{\frac{2\pi i k t}{N}}$$

frequency

time

sample size



$$\mathrm{Re}\left(e^{\frac{2\pi \cdot 3t}{48}}\right) = \cos\left(\frac{2\pi \cdot 3t}{48}\right)$$

'Pure tones'

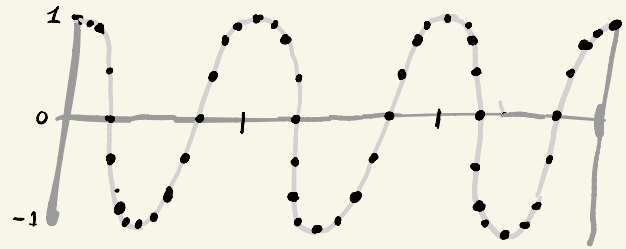
Are represented by:

frequency

time

$$e^{\frac{2\pi i k t}{N}}$$

sample size



$$\mathrm{Re}\left(e^{\frac{2\pi \cdot 3t}{48}}\right) = \cos\left(\frac{2\pi \cdot 3t}{48}\right)$$

Discrete Fourier Transform

given $(a_0, \dots, a_{N-1}) \in \mathbb{C}^N$,
finds $(y_0, \dots, y_{N-1}) \in \mathbb{C}^N$ with:

$$a_t = \frac{1}{N} \sum_{k=0}^{N-1} y_k \, \omega^{-kt} \qquad \text{where } \omega = e^{\frac{2\pi i}{N}}.$$

(to be consistent with the book)

Finding $y_0, y_1, \dots, y_{N-1}$

$\omega = e^{\frac{2\pi i}{N}}$

So $\omega^N = e^{2\pi i} = 1$

Since $a_t = \frac{1}{N} \sum_{k=0}^{N-1} y_k \omega^{-kt}$ we have:

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega = \omega^{N-1} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \cdots & \omega^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{-1} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{pmatrix}$$

So we have to invert this matrix.

$$\frac{1}{N}\begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \cdots & \omega^{2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega & \omega^{2} & \omega^{3} & \cdots & \omega^{-1} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{1} & \omega^{2} & \omega^{3} & \cdots & \omega^{-1} \\ 1 & \omega^{2} & \omega^{4} & \omega^{6} & \cdots & \omega^{2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega \end{pmatrix}$$

$$AB = I \quad \begin{pmatrix} - \\ m \end{pmatrix} \begin{pmatrix} | \\ n \end{pmatrix} = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \end{pmatrix}$$

because: $$\sum_{k=0}^{N-1} \omega^{-km}\,\omega^{kn} = \sum_{k=0}^{N-1} (\omega^{m-n})^{k} \overset{\text{if } m=n}{=\!=\!=} \sum_{k=0}^{N-1} 1^{k} = N$$

$$\overset{\text{if } m\neq n}{=\!=} \frac{1-(\omega^{m-n})^{N}}{1-\omega^{m-n}} = \frac{1-(\omega^{N})^{m-n}}{1-\omega^{m-n}} \overset{\omega^{N}=1}{=} \frac{1-1^{m-n}}{1-\omega^{m-n}} = 0.$$

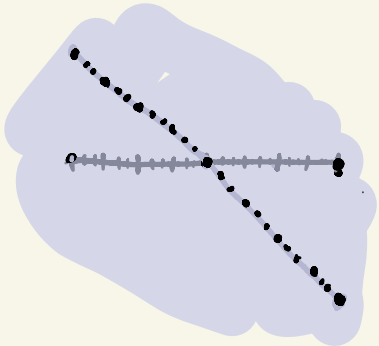
$$\omega^{N} = \left(e^{\frac{2\pi i}{N}}\right)^{N} = e^{2\pi i} = 1$$

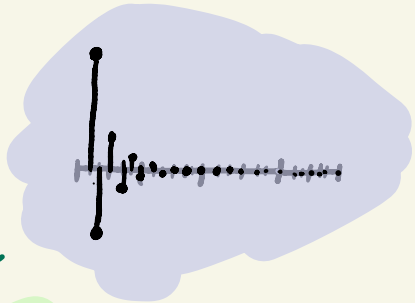
Discrete Fourier Transform

$$\begin{pmatrix} - \\ - \\ - \end{pmatrix} \begin{pmatrix} | \\ | \\ | \end{pmatrix} = \begin{pmatrix} \bullet \\ \bullet \\ \bullet \end{pmatrix} \quad O(n^2)$$

$$y_s = \sum_{k=0}^{N-1} a_t \omega^{ks}$$

$$\xrightarrow{\text{DFT}}$$





$$\xleftarrow{\text{IDFT}}$$

$$a_t = \frac{1}{N} \sum_{k=0}^{N-1} y_k \omega^{-kt}$$

signal

$(a_0, a_1, \dots, a_{N-1}) \in \mathbb{C}^N$

frequencies

$(y_0, y_1, \dots, y_{N-1}) \in \mathbb{C}^N$

Fast Fourier Transform (FFT)

Computing the DFT $(y_0, y_1, \ldots, y_{N-1})$ from $(a_0, a_1, \ldots, a_{N-1})$ naively takes $\Theta(N^2)$ arithmetic operations on $\mathbb{C}$ (or rather flops,) because of the matrix times vector involved:

$$\begin{pmatrix} \vdots \\ y \\ \vdots \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix} \begin{pmatrix} \vdots \\ a \\ \vdots \end{pmatrix}.$$

And for an arbitrary matrix this is probably the best we can do. However, this matrix is <u>not</u> arbitrary, and we can use a clever divide-and-conquer trick to get down to $\Theta(N \log(N))$!

... if N is a power of 2.

FFT : Divide and Conquer

Let N be a power of 2. Write; given $(a_0, \dots, a_{N-1}) \in \mathbb{C}^N$,

$$\mathrm{DFT}(a_0, \dots, a_{N-1}) = \left(\sum_{k=0}^{N-1} a_k \, \omega_N^{ks} \right)_{s=0}^{N-1} \quad \text{where} \quad \omega_N = e^{\frac{2\pi i}{N}}.$$

Then, splitting even and odd terms:

$$\mathrm{DFT}(a_0, \dots, a_{N-1})_s = \sum_{k=0}^{N/2-1} a_{2k} \, \omega_N^{2ks} + \sum_{k=0}^{N/2-1} a_{2k+1} \, \omega_N^{2(k+1)s}$$

$$= \underbrace{\sum_{k=0}^{N/2-1} a_{2k} (\omega_N^2)^{ks}}_{\mathrm{DFT}(a_0, a_2, \dots, a_{N-2})_s} + \omega^s \underbrace{\sum_{k=0}^{N/2-1} a_{2k+1} (\omega_N^2)^{ks}}_{\mathrm{DFT}(a_1, a_3, \dots, a_{N-1})_s}$$

because

$$\omega_N^2 = e^{4\pi i / N} = \omega_{N/2}$$

$\mathrm{FFT}(a_0, \ldots, a_{N-1})$: (N power of 2)

if $N = 1$: return (a_0)

let $\omega = e^{2\pi i/N}$

let $\omega_s = 1$ // will be ω^s

let $y_{even} = \mathrm{FFT}(a_0, a_2, \ldots, a_{N-2})$

let $y_{odd} = \mathrm{FFT}(a_1, a_3, \ldots, a_{N-1})$

for $s = 0$ to $N/2 - 1$:

  let $y = (y_{even})_s + \omega_s \cdot (y_{odd})_s$

  $\omega_s = \omega_s \cdot \omega$

return $(y_0, \ldots, y_{N-1})$

$FFT(a_0, \dots, a_{N-1})$: COMPLEXITY ANALYSIS (N power of 2)

```
if N=1 : return (a_0)
let ω = e^{2πi/N}
let ω_s = 1 // will be ω^s
let y_even = FFT(a_0, a_2, ..., a_{N-2})
let y_odd = FFT(a_1, a_3, ..., a_{N-1})

for s = 0 to N/2-1 :
    let y = (y_even)_s + ω_s·(y_odd)_s
    ω_s = ω_s · ω
return (y_0, ..., y_{N-1})
```

$\Theta(1)$

$2T(N/2)$

$\Theta(N)$

$FFT(a_0, \dots, a_{N-1})$: COMPLEXITY ANALYSIS (N power of 2)

```
if N=1 : return (a_0)
let ω = e^{2πi/N}
let ω_s = 1 // will be ω^s
let y_even = FFT(a_0, a_2, ..., a_{N-2})
let y_odd = FFT(a_1, a_3, ..., a_{N-1})

for s = 0 to N/2-1 :
    let y = (y_even)_s + ω_s·(y_odd)_s
    ω_s = ω_s · ω
return (y_0, ..., y_{N-1})
```

$\Theta(1)$

$2T(N/2)$

$\Theta(N)$

$$T(N) = 2T(N/2) + \Theta(N)$$

$$\Downarrow$$

$$T(N) = \Theta(N \log(N))$$

INVERSE?

For the inverse DFT, IDFT, there is a similar algorithm with runtime $\Theta(N \log(N))$.

(Replace ω by ω^{-1}, and add $/N$ at the end.)

Application: fast polynomial multiplication

Recall that a polynomial over $\mathbb{C}$ is of the form:

$$f(x) = a_0 + a_1 x + \cdots + a_{N-1} x^{N-1} = \sum_{k=0}^{N-1} a_k x^{N-1},$$

where $a_0, a_1, \ldots, a_{N-1}$ are the coefficients of f.

Multiplying polynomials is not simply a matter of multiplying coefficients, because e.g.

$$(a_0 + a_1 x)(b_0 + b_1 x) = a_0 b_0 + (a_0 b_1 + a_1 b_0) x + a_1 b_1 x^2.$$

In general:

$$\left(\sum_{k=0}^{N-1} a_k x^k\right)\left(\sum_{k=0}^{N-1} b_k x^k\right) = \sum_{k=0}^{2N-2}\left(\sum_{\ell=0}^{k} a_\ell b_{k-\ell}\right) x^k$$

Hence, the naive algorithm will be: $\Theta(n^2)$

WE CAN DO BETTER ...

By multiplying DFT's. Given $a, b \in \mathbb{C}^N$, $s \in \{0, \dots, N-1\}$

$$\mathrm{DFT}(a)_s \cdot \mathrm{DFT}(b)_s = \left(\sum_{k=0}^{N-1} a_k \omega^{sk}\right)\left(\sum_{\ell=0}^{N-1} b_\ell \omega^{s\ell}\right)$$

$$= \sum_{k,\ell=0}^{N-1} a_k b_\ell \omega^{s(k+\ell)}$$

$$= \sum_{n=0}^{N-1}\left(\sum_{m=0}^{N-1} a_m b_{n-m \bmod N}\right)\omega^{sn}$$

Almost the same as polynomial mult.

$$\sum_{k=0}^{2N-2}\left(\sum_{\ell=0}^{k} a_\ell b_{k-\ell}\right)x^k$$

But ω 'wraps' around:

$$\omega^{N+k} = \omega^k$$

$$x^{N+k} \neq x^k$$

WE CAN DO BETTER ...

By multiplying DFT's. Given $a, b \in \mathbb{C}^N$, $s \in \{0, \dots, N-1\}$

$$\mathrm{DFT}(a)_s \cdot \mathrm{DFT}(b)_s = \left(\sum_{k=0}^{N-1} a_k \omega^{sk}\right)\left(\sum_{\ell=0}^{N-1} b_\ell \omega^{s\ell}\right)$$

$$= \sum_{k,\ell=0}^{N-1} a_k b_\ell \omega^{s(k+\ell)}$$

$$= \sum_{n=0}^{N-1}\left(\sum_{m=0}^{N-1} a_m b_{n-m \bmod N}\right)\omega^{sk}$$

Double N so that $a_n = b_n = 0$ for $n > N/2$

$$= \sum_{n=0}^{N-1}\left(\sum_{m=0}^{n} a_m b_{n-m}\right)\omega^{sk}$$

Almost the same as polynomial mult.

$$\sum_{k=0}^{2N-2}\left(\sum_{\ell=0}^{k} a_\ell b_{k-\ell}\right)x^k$$

But ω 'wraps' around:

$$\omega^{N+k} = \omega^k$$

$$x^{N+k} \neq x^k$$

FAST POLYNOMIAL MULTIPLICATION

To compute the coefficients of

$$\left(\sum_{k=0}^{N-1} a_k x^k\right)\left(\sum_{k=0}^{N-1} b_k x^k\right),$$

1. write $a = (a_0, \dots, a_{N-1}, 0, 0, \dots, 0)$, $b = (b_0, \dots, b_{N-1}, 0, 0, \dots, 0) \in \mathbb{C}^M$, where M is a power of 2 with $M > 2N$,
2. Compute $y := DFT(a)$ and $z := DFT(b)$,
3. multiply y and z pointwise : $u_s := (y_s \cdot z_s)_{s=0}^{M}$.
4. Compute $c := IDFT(u)$, and return the first $2N$ elements of c.

FAST POLYNOMIAL MULTIPLICATION

To compute the coefficients of

$$\left(\sum_{k=0}^{N-1} a_k x^k\right)\left(\sum_{k=0}^{N-1} b_k x^k\right),$$

1. write $a = (a_0, \dots, a_{N-1}, 0, 0, \dots, 0)$, $b = (b_0, \dots, b_{N-1}, 0, 0, \dots, 0) \in \mathbb{C}^M$, where M is a power of 2 with $M > 2N$, $\Theta(N)$

2. Compute $y := DFT(a)$ and $z := DFT(b)$, $\Theta(N \log(N))$

3. multiply y and z pointwise: $u_s := (y_s \cdot z_s)_{s=0}^{M}$. $\Theta(N)$

4. Compute $c := IDFT(u)$, and return the first $2N$. $\Theta(N \log N)$

OVER ALL: $\Theta(N \log N)$

Fin

Next week: no lecture
(spring break)

May 9: start of Part II
(COMPLEXITY of PROBLEMS)