Integer multiplication and the Fast Fourier Transform

Multiplying n-bit integers

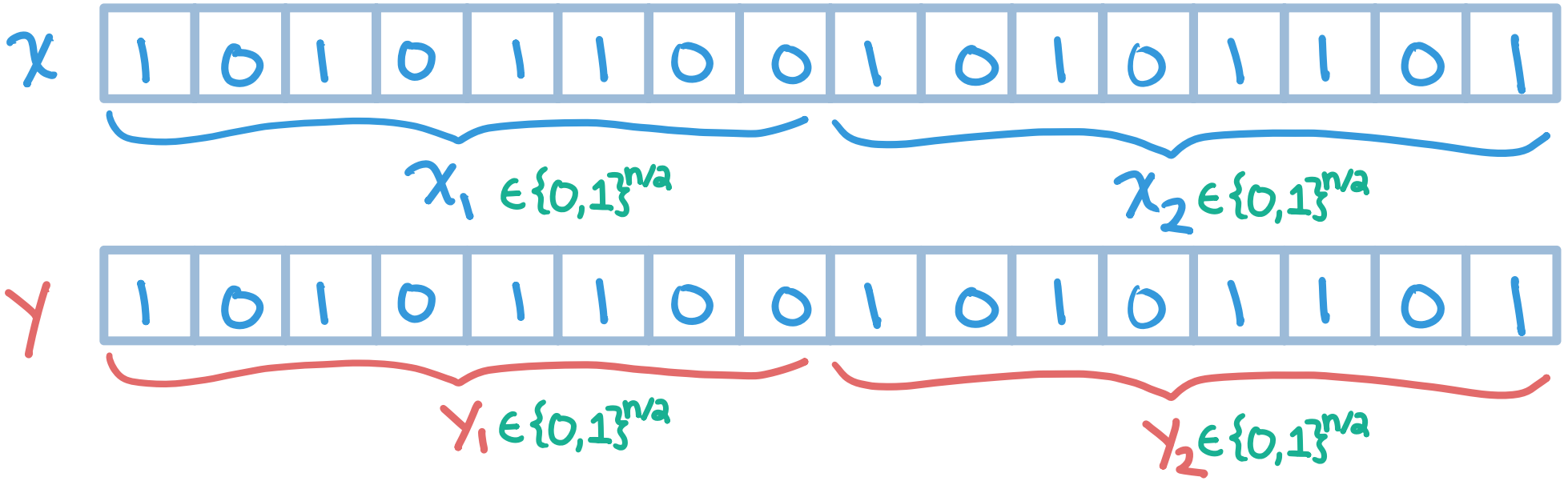
Karatsuba 1960

Input: $x, y \in \{0,1\}^n$

Output: product $(xy) \in \{0,1\}^{2n}$

$$\begin{array}{r} 1234 \\ \times\ 5678 \\ \hline \end{array}$$

Divide bits in half



$$x = x_1 2^{n/2} + x_2, \quad y = y_1 2^{n/2} + y_2$$

$$x \cdot y = x_1 y_1 2^n + (x_1 y_2 + y_1 x_2) 2^{n/2} + x_2 y_2$$

which multiplies 4 pairs of (n/2)-bit numbers

$$x \cdot y = x_1 y_1 2^n + (x_1 y_2 + y_1 x_2) 2^{n/2} + x_2 y_2$$

Karatsuba's Trick

$$(x_1 + x_2)(y_1 + y_2) = x_1 y_1 + x_1 y_2 + x_2 y_1 + x_2 y_2$$

① Recursive specification

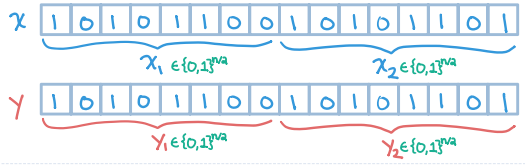
Designing a recursive algorithm

① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

Divide bits in half



$x = x_1 2^{n/2} + x_2, \quad y = y_1 2^{n/2} + y_2$

$x \cdot y = x_1 y_1 2^n + (x_1 y_2 + y_1 x_2) 2^{n/2} + x_2 y_2$

which multiplies 4 pairs of (n/2)-bit numbers

Karatsuba's Trick

$(x_1 + x_2)(y_1 + y_2) = x_1 y_1 + x_1 y_2 + x_2 y_1 + x_2 y_2$

already computed

middle term

one multiply

$x_1 y_2 + x_2 y_1 = x_1 y_1 + x_2 y_2 - (x_1 + x_2)(y_1 + y_2)$

2 multiplies for the price of 1 (more)!

① Recursive specification

Designing a recursive algorithm

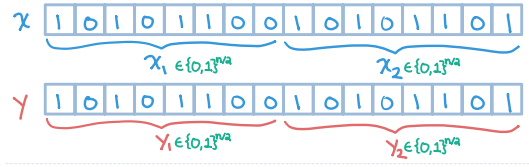
① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

multiply($x[1..n], y[1..n]$) = the ($2n$-bit integer) product of x and y (as n-bit integers).

Divide bits in half



$x = x_1 2^{n/2} + x_2, \quad y = y_1 2^{n/2} + y_2$

$x \cdot y = x_1 y_1 2^n + (x_1 y_2 + y_1 x_2) 2^{n/2} + x_2 y_2$

which multiplies 4 pairs of $(n/2)$-bit numbers

Karatsuba's Trick

already computed

$(x_1 + x_2)(y_1 + y_2) = x_1 y_1 + x_1 y_2 + x_2 y_1 + x_2 y_2$

one multiply — middle term

$x_1 y_2 + x_2 y_1 = x_1 y_1 + x_2 y_2 - (x_1 + x_2)(y_1 + y_2)$

2 multiplies for the price of 1 (more)!

① Recursive specification

`select`$(k, A[1..n])$: Given an array $A[1..n]$ of comparable elements and an integer $k \in \{1, \ldots, n\}$, returns the kth smallest element out of $A[1..n]$.

Designing a recursive algorithm
① Recursive specification
- declares input/output of algo
- "contract" algo must fulfill
- induction hypothesis for correctness

② Implement spec

③ Prove correctness
argue algo satisfies recursive spec for all n by induction on n

② Implement spec

`multiply`$(x[1..n], y[1..n])$

1. If $n = 0$ then return 0.
2. If $n = 1$ then return $x[1] * y[1]$.
3. If n is not even then pad x and y with a leading 0, making n even. // $O(n)$.
4. Let $x_1 = x[1..n/2]$, $x_2 = [n/2+1..n]$, $y_1 = y[1..n/2]$, and $y_2 = y[n/2+1..n]$ (as $n/2$-bit integers). // $O(n)$.
5. Let $x_3[1..n/2+1] = x_1 + x_2$ and $y_3[1..n/2+1] = (y_1 + y_2)$ (as $(n/2+1)$-bit integers). // $O(n)$.
6. Let $Z_1 =$ `multiply`(x_1, y_1), $Z_2 =$ `multiply`(x_2, y_2), and $Z_3 =$ `multiply`(x_3, y_3). // $3T((n+2)/2)$.

/* *Note that multiplying by 2^n is just a bit-shift and takes $O(n)$ time.* */

7. Return $2^n * Z_1 + 2^{n/2}(Z_3 - Z_1 - Z_2) + Z_2$

$T(n) =$

4. Running time

`multiply`$(x[1..n], y[1..n])$

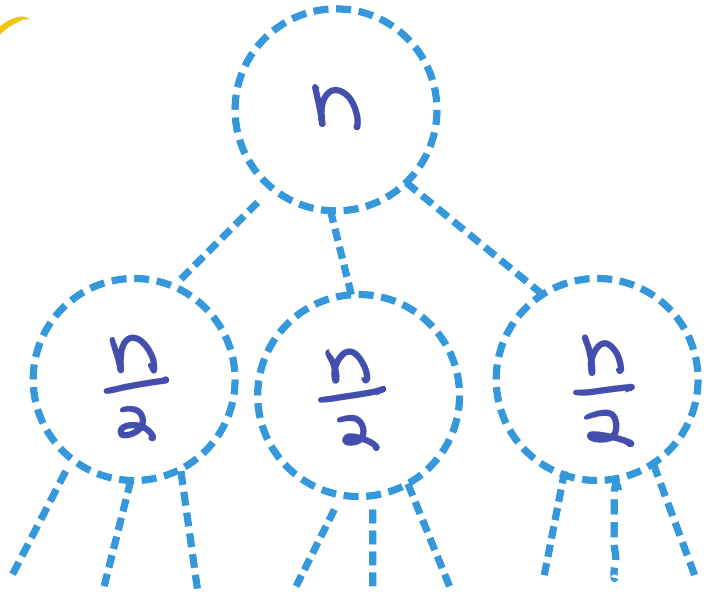
1. If $n = 0$ then return 0.
2. If $n = 1$ then return $x[1] * y[1]$.
3. If n is not even then pad x and y with a leading 0, making n even. // $O(n)$.
4. Let $x_1 = x[1..n/2]$, $x_2 = [n/2+1..n]$, $y_1 = y[1..n/2]$, and $y_2 = y[n/2+1..n]$ (as $n/2$-bit integers). // $O(n)$.
5. Let $x_3[1..n/2+1] = x_1 + x_2$ and $y_3[1..n/2+1] = (y_1 + y_2)$ (as $(n/2+1)$-bit integers). // $O(n)$.
6. Let $Z_1 =$ `multiply`(x_1, y_1), $Z_2 =$ `multiply`(x_2, y_2), and $Z_3 =$ `multiply`(x_3, y_3). // $3T((n+2)/2)$.

/* *Note that multiplying by 2^n is just a bit-shift and takes $O(n)$ time.* */

7. Return $2^n * Z_1 + 2^{n/2}(Z_3 - Z_1 - Z_2) + Z_2$

4. Running time

$T(n) = 3T(\frac{n}{2}) + O(n)$



$h = \log_2 n$

n

$3 \times \frac{n}{2} = \frac{3}{2}n$

$(\frac{3}{2})^2 n$

$(\frac{3}{2})^h n$

`multiply`$(x[1..n], y[1..n])$

1. If $n = 0$ then return 0.
2. If $n = 1$ then return $x[1] * y[1]$.
3. If n is not even then pad x and y with a leading 0, making n even. // $O(n)$.
4. Let $x_1 = x[1..n/2]$, $x_2 = [n/2+1..n]$, $y_1 = y[1..n/2]$, and $y_2 = y[n/2+1..n]$ (as $n/2$-bit integers). // $O(n)$.
5. Let $x_3[1..n/2+1] = x_1 + x_2$ and $y_3[1..n/2+1] = (y_1 + y_2)$ (as $(n/2+1)$-bit integers). // $O(n)$.
6. Let $Z_1 =$ `multiply`(x_1, y_1), $Z_2 =$ `multiply`(x_2, y_2), and $Z_3 =$ `multiply`(x_3, y_3). // $3T((n+2)/2)$.

/* *Note that multiplying by 2^n is just a bit-shift and takes $O(n)$ time.* */

7. Return $2^n * Z_1 + 2^{n/2}(Z_3 - Z_1 - Z_2) + Z_2$

Running time

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$$\sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i n =$$

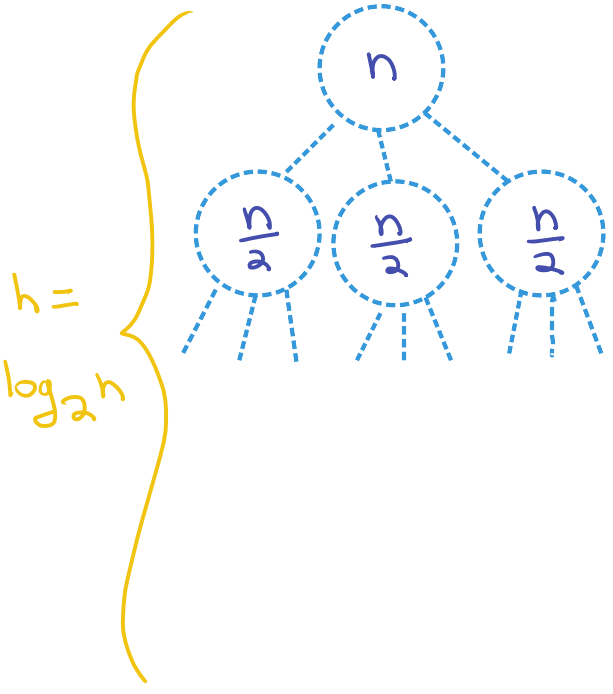
4. Running time

`multiply`$(x[1..n], y[1..n])$

1. If $n = 0$ then return 0.
2. If $n = 1$ then return $x[1] * y[1]$.
3. If n is not even then pad x and y with a leading 0, making n even. // $O(n)$.
4. Let $x_1 = x[1..n/2]$, $x_2 = [n/2+1..n]$, $y_1 = y[1..n/2]$, and $y_2 = y[n/2+1..n]$ (as $n/2$-bit integers). // $O(n)$.
5. Let $x_3[1..n/2+1] = x_1 + x_2$ and $y_3[1..n/2+1] = (y_1 + y_2)$ (as $(n/2+1)$-bit integers). // $O(n)$.
6. Let $Z_1 =$ `multiply`(x_1, y_1), $Z_2 =$ `multiply`(x_2, y_2), and $Z_3 =$ `multiply`(x_3, y_3). // $3T((n+2)/2)$.

/* *Note that multiplying by 2^n is just a bit-shift and takes $O(n)$ time.* */

7. Return $2^n * Z_1 + 2^{n/2}(Z_3 - Z_1 - Z_2) + Z_2$



Running time

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$$\sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i n = O\left(\left(\frac{3}{2}\right)^{\log_2 n} \cdot n\right)$$

$$= O\left(n^{\log_2(3/2)+1}\right)$$

$$= O\left(n^{\log_2(3)}\right)$$

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) = O\left(n^{\log_2 3}\right)$$

$$\approx n^{1.585}$$

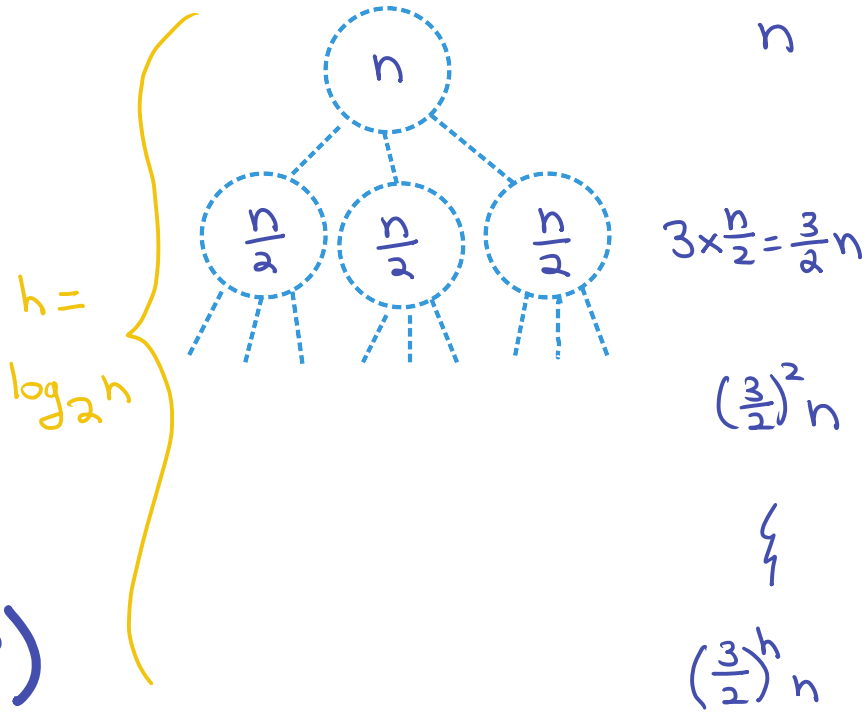
4. Running time

`multiply`$(x[1..n], y[1..n])$

1. If $n = 0$ then return 0.
2. If $n = 1$ then return $x[1] * y[1]$.
3. If n is not even then pad x and y with a leading 0, making n even. // $O(n)$.
4. Let $x_1 = x[1..n/2]$, $x_2 = [n/2+1..n]$, $y_1 = y[1..n/2]$, and $y_2 = y[n/2+1..n]$ (as $n/2$-bit integers). // $O(n)$.
5. Let $x_3[1..n/2+1] = x_1 + x_2$ and $y_3[1..n/2+1] = (y_1 + y_2)$ (as $(n/2+1)$-bit integers). // $O(n)$.
6. Let $Z_1 =$ `multiply`(x_1, y_1), $Z_2 =$ `multiply`(x_2, y_2), and $Z_3 =$ `multiply`(x_3, y_3). // $3T((n+2)/2)$.

/* *Note that multiplying by 2^n is just a bit-shift and takes $O(n)$ time.* */

7. Return $2^n * Z_1 + 2^{n/2}(Z_3 - Z_1 - Z_2) + Z_2$

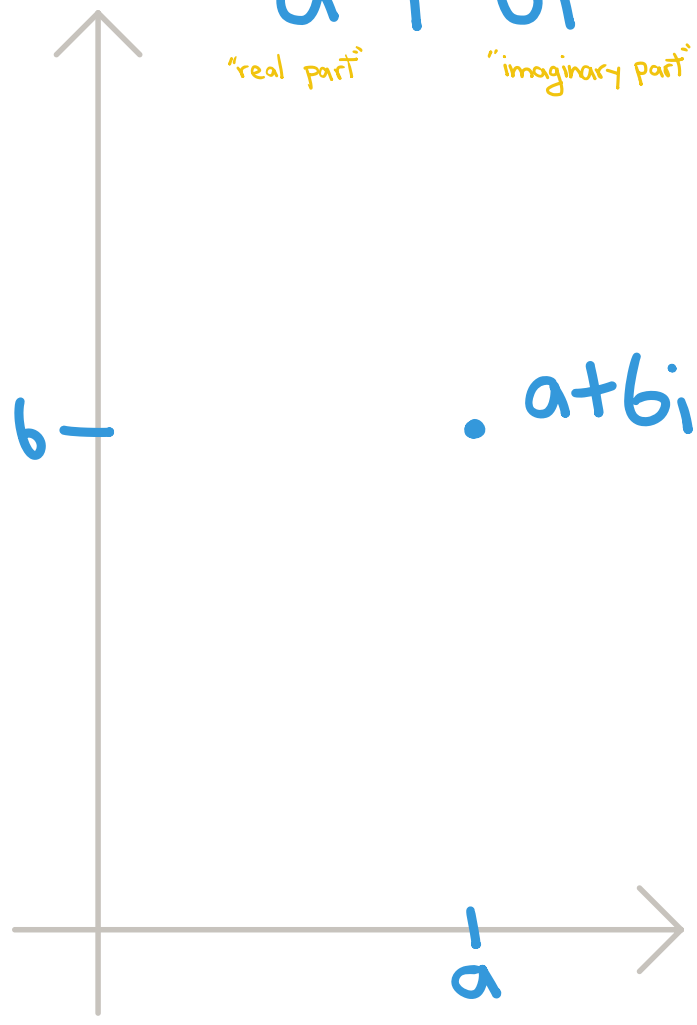


Integer multiplication and the Fast Fourier Transform

Complex numbers $\mathbb{C}$

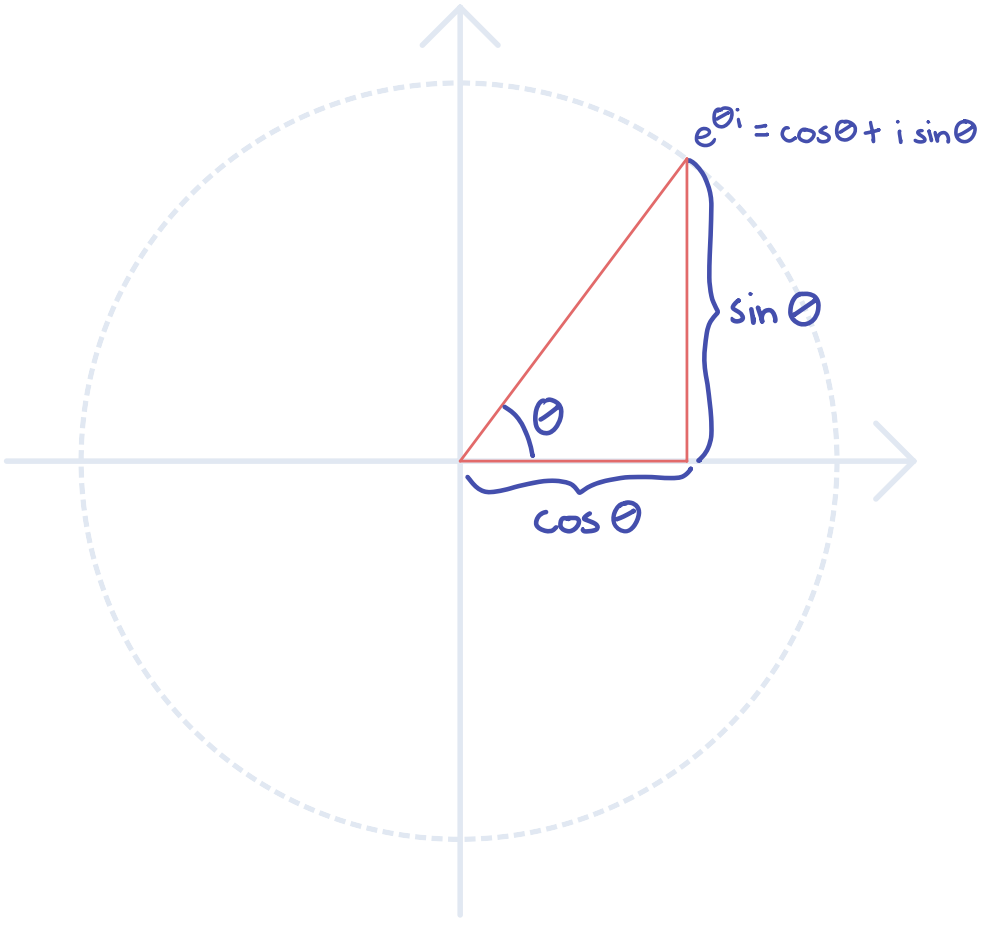
$a + bi$

"real part" "imaginary part"



Euler's formula

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



$a + bi$

"real part" "imaginary part"

"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

nth roots of unity:

any $w \in \mathbb{C}$ s.t. $w^n = -1$

e.g.

$a + bi$

"real part" "imaginary part"

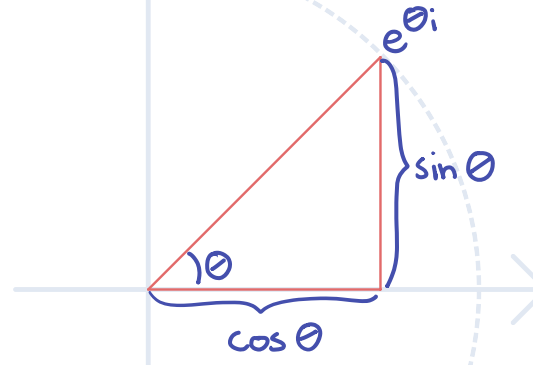
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

(nth) discrete Fourier transform F_n

Input:

Output:

$a + bi$

"real part" "imaginary part"

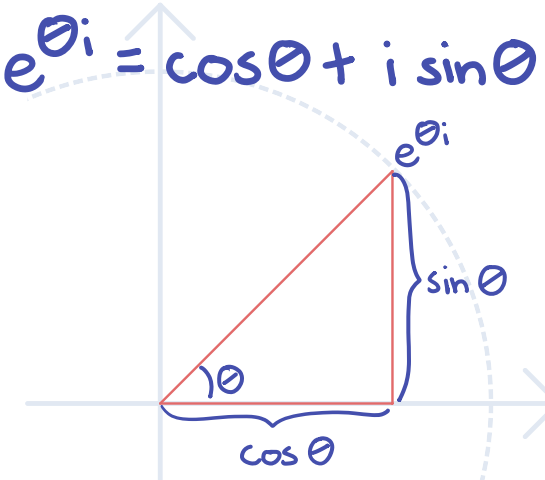
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

Input: $a_0, a_1, a_2, \cdots, a_{n-1} \in \mathbb{C}$

Output: n values

$$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \cdots, p(\omega_n^{n-1}))$$

$$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \cdots, p(\omega_n))$$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

(i.e., F_n and F_n^* are essentially inverse)

$a + bi$

"real part" "imaginary part"

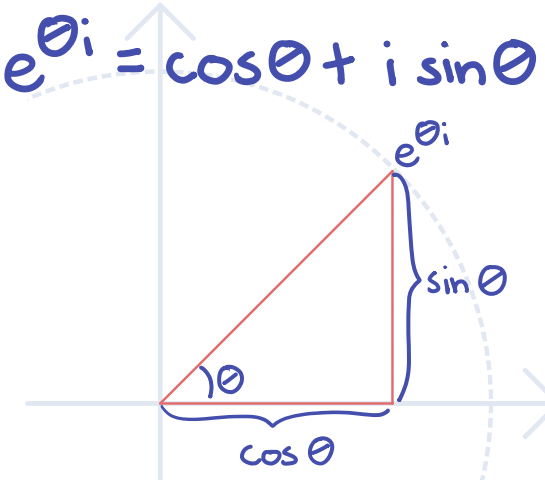
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

Input: $a_0, a_1, a_2, \cdots, a_{n-1} \in \mathbb{C}$

Output: n values

$$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \cdots, p(\omega_n^{n-1}))$$

$$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \cdots, p(\omega_n))$$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$ and $F_n^* a$ can be computed in $O(n \log n)$ time.

$a + bi$

"real part" "imaginary part"

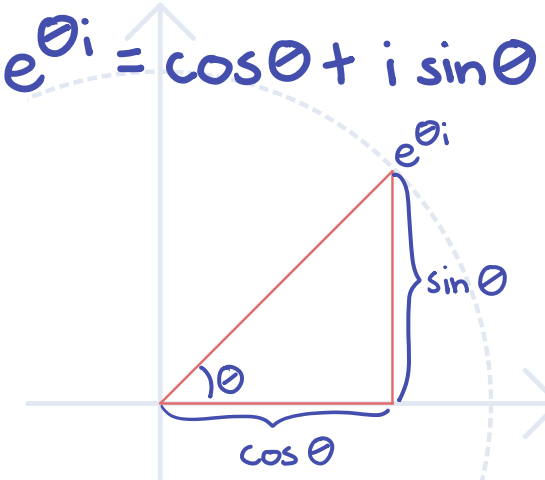
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

Application: multiplying polynomials

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

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

goal: compute

$$r(x) = p(x)\,q(x)$$

$a + bi$

"real part" "imaginary part"

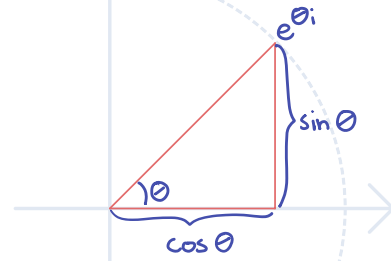
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

$$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \cdots, p(\omega_n^{n-1}))$$

$$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \cdots, p(\omega_n))$$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$, $F_n^* a$ in $O(n \log n)$

Input: $p(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$

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

goal: $r(x) = p(x)q(x) = c_0 + c_1 x + \cdots + c_{2n-2} x^{n-2}$

Idea: $r(\omega_{2n}^i) = p(\omega_{2n}^i)\, q(\omega_{2n}^i)$

(p, q)

$a + bi$

"real part" "imaginary part"

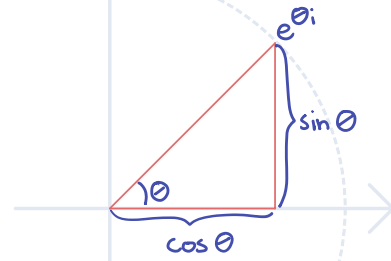
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \cdots, p(\omega_n^{n-1}))$

$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \cdots, p(\omega_n))$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$, $F_n^* a$ in $O(n \log n)$

Input: $a_0, a_1, a_2, \cdots, a_{n-1} \in \mathbb{C}$

Output: n values

$$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \cdots, p(\omega_n^{n-1}))$$

$$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \cdots, p(\omega_n))$$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$ and $F_n^* a$ can be computed in $O(n \log n)$ time.

$a + bi$

"real part" "imaginary part"

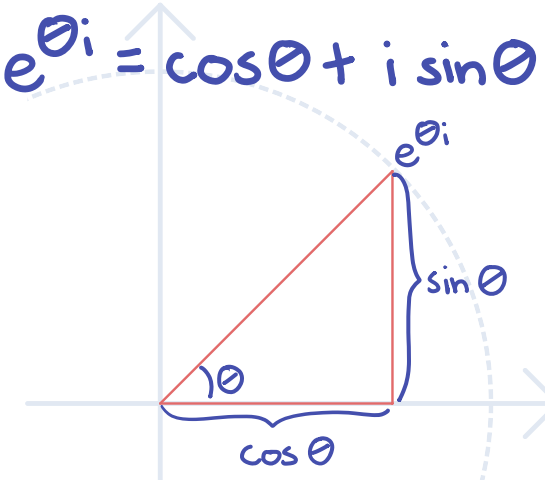
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

Idea: $\omega_{2n}^2 = \omega_n$

let $a = (a_0, a_1, \cdots, a_{2n-1})$.

$(F_{2n} a)_k =$

$=$

$a + bi$

"real part" "imaginary part"

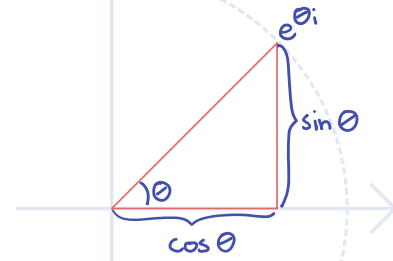
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \cdots, \omega_n^{n-1}$ are all nth roots of unity

$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \cdots, p(\omega_n^{n-1}))$

$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \cdots, p(\omega_n))$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$, $F_n^* a$ in $O(n \log n)$

Idea: $\omega_{2n}^2 = \omega_n$

let $a = (a_0, a_1, \dots, a_{2n-1})$.

$$(F_{2n}a)_k = a_0 + a_1\omega_{2n}^k + \dots + a_{2n-1}(\omega_{2n}^k)^{2n-1}$$

$$= \underbrace{(a_0 + a_2(\omega_{2n}^k)^2 + \dots + a_{2n-2}(\omega_{2n}^k)^{2n-2})}_{\text{(even)}}$$

$$+ \underbrace{(a_1\omega_{2n}^k + a_3(\omega_{2n}^k)^3 + \dots + a_{2n-1}(\omega_{2n}^k)^{2n-1})}_{\text{(odd)}}$$

$a + bi$
"real part" "imaginary part"

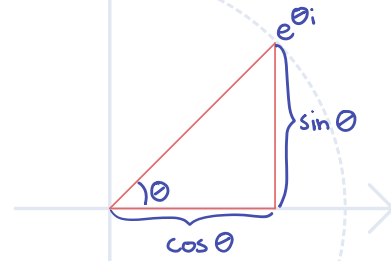
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}$ are all nth roots of unity

$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \dots, p(\omega_n^{n-1}))$

$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \dots, p(\omega_n))$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$, $F_n^* a$ in $O(n \log n)$

Idea: $\omega_{2n}^2 = \omega_n$

let $a = (a_0, a_1, \dots, a_{2n-1})$.

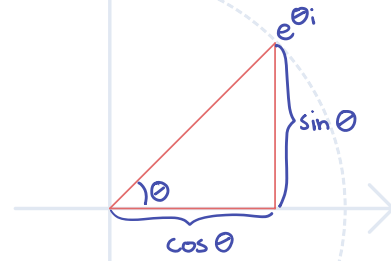
$$(F_{2n}a)_k = a_0 + a_1\omega_{2n}^k + \dots + a_{2n-1}(\omega_{2n}^k)^{2n-1}$$

$$= (a_0 + a_2(\omega_{2n}^k)^2 + \dots + a_{2n-2}(\omega_{2n}^k)^{2n-2}) \quad \text{(even)}$$

$$+ (a_1\omega_{2n}^k + a_3(\omega_{2n}^k)^3 + \dots + a_{2n-1}(\omega_{2n}^k)^{2n-1}) \quad \text{(odd)}$$

$a + bi$
"real part" "imaginary part"
"imaginary i" s.t. $i^2 = -1$
$(a+bi)(c+di) = ac + (bc+ad)i - bd$
$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

Euler's formula:
$e^{\theta i} = \cos\theta + i\sin\theta$



Euler's identity:
- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}$ are all nth roots of unity

$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \dots, p(\omega_n^{n-1}))$

$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \dots, p(\omega_n))$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$, $F_n^* a$ in $O(n \log n)$

Idea: $\omega_{2n}^2 = \omega_n$

let $a = (a_0, a_1, \dots, a_{2n-1})$.

$$(F_{2n}a)_k = a_0 + a_1\omega_{2n}^k + \dots + a_{2n-1}(\omega_{2n}^k)^{2n-1}$$

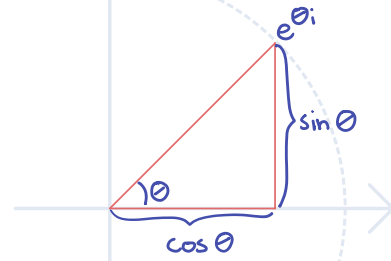
$$= (a_0 + a_2(\omega_{2n}^k)^2 + \dots + a_{2n-2}(\omega_{2n}^k)^{2n-2}) \quad \text{(even)}$$

$$+ (a_1\omega_{2n}^k + a_3(\omega_{2n}^k)^2 + \dots + a_{2n-1}(\omega_{2n}^k)^{2n-1}) \quad \text{(odd)}$$

$=$

$a + bi$
"real part" "imaginary part"
"imaginary i" s.t. $i^2 = -1$
$(a+bi)(c+di) = ac + (bc+ad)i - bd$
$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

Euler's formula:
$e^{\theta i} = \cos\theta + i\sin\theta$



Euler's identity:
- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}$ are all nth roots of unity

$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \dots, p(\omega_n^{n-1}))$

$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \dots, p(\omega_n))$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a$, $F_n^* a$ in $O(n \log n)$

Fast Fourier Transform $(a_0, \dots, a_n)$:

1. if $n=0$ return $\emptyset$
2. if $n=1$ return (a_0)
3. let $(y_0, \dots, y_{\lceil n/2 \rceil - 1}) = FFT(a_{even})$
4. let $(z_0, \dots, z_{\lfloor n/2 \rfloor - 1}) = FFT(a_{odd})$
5. return $(x_0, \dots, x_{n-1})$

where $x_k = y_k + \omega_n^k z_k \quad k = 0, \dots, n-1$

$T(n) =$

$a + bi$

"real part" "imaginary part"

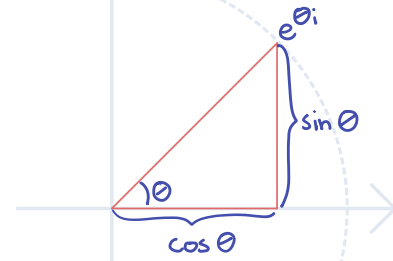
"imaginary i" s.t. $i^2 = -1$

$(a+bi)(c+di) = ac + (bc+ad)i - bd$

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

Euler's formula:

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



Euler's identity:

- $e^{\pi i} = -1$
- $e^{2\pi i} = 1$

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

$1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}$ are all nth roots of unity

$F_n a = (p(1), p(\omega_n), p(\omega_n^2), \dots, p(\omega_n^{n-1}))$

$F_n^* a = (p(1), p(\omega_n^{n-1}), p(\omega_n^{n-2}), \dots, p(\omega_n))$

where $p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^{n-1}$

Fact: $F_n^* F_n a = F_n F_n^* a = na$

Theorem: $F_n a, F_n^* a$ in $O(n \log n)$

Matrix Multiplication

Let A and B be two $n \times n$ matrices.

Matrix Multiplication

Let A and B be two $n \times n$ matrices.

$$(AB)_{ij} = \sum_k A_{ik} B_{kj}$$

$$AB = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

Strassen, 1969

(8 multiplies)

$$(A_{11} + A_{22})(B_{11} + B_{22}) = A_{11}B_{11} + A_{22}B_{11} + A_{11}B_{22} + A_{11}B_{22} \tag{4.3}$$
$$(A_{11} - A_{21})(B_{11} - B_{12}) = A_{11}B_{11} - A_{21}B_{11} - A_{11}B_{12} + A_{21}B_{12} \tag{4.4}$$
$$(A_{12} - A_{21})(B_{21} + B_{22}) = A_{12}B_{12} - A_{21}B_{21} - A_{21}B_{22} + A_{12}B_{22} \tag{4.5}$$
$$A_{11}(B_{12} - B_{22}) = A_{11}(B_{12} - B_{22}) \tag{4.6}$$
$$(A_{11} + A_{12})B_{22} = A_{11}B_{22} + A_{12}B_{22} \tag{4.7}$$
$$(A_{11} + A_{22})(B_{11} + B_{22}) = A_{11}B_{11} + A_{11}B_{11} + A_{22}B_{11} + A_{22}B_{22} \tag{4.8}$$
$$(A_{11} - A_{21})(B_{11} + B_{12}) = A_{11}B_{11} - A_{21}B_{11} + A_{11}B_{12} - A_{21}B_{12} \tag{4.9}$$

Then

$$\begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$
$$= \begin{pmatrix} (4.3) + (4.4) - (4.7) + (4.5) & (4.6) + (4.7) \\ (4.8) + (4.4) & (4.6) + (4.3) - (4.8) - (4.9) \end{pmatrix}$$

$T(n) =$

Exercise 4.1. Above we observed that two degree n polynomials $p(x)$ and $q(x)$ can be multiplied in $O(n \log n)$ time. Consider the case when the degrees are very unbalanced; say, $p(x)$ has degree n and $q(x)$ has degree k for $k \ll n$. Design and analyze an algorithm computing the product $p(x)q(x)$ in $O(n \log k)$ time.

Exercise 4.2. Let $p(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} a_{ij} x^i y^j$ and $q(x,y) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} x^i y^j$ be two multivariate polynomials each with maximum degree m in x and maximum degree n in y. Consider the product $r(x,y) = p(x,y)q(x,y)$, which has maximum degree $2m$ in x and maximum degree $2n$ in y. It is easy to see that r can be computed in $O(m^2n^2)$ time with nested loops. Design and analyze a faster algorithm computing the product r.