Key recurrences for divide-and-conquer algorithms:

\[ T(n) = T\left(\frac{n}{2}\right) + O(1) = O(\log n) \] (binary search)

\[ T(n) = T\left(\frac{n}{2}\right) + O(n) = O(n) \]

\[ T(n) = 2T\left(\frac{n}{3}\right) + O(1) = O(n) \]

\[ T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n\log n) \] (merge sort)

From the desired recurrence we get a high level idea of the algorithm.

Divide & conquer approach:

1) Break into subproblems of the same type
   Typically problems of half the size.
2) Recursively solve these subproblems
3) Combine/merge solutions to subproblems to get solution to whole problem.
Example: Two polynomials of degree $D=2$:

\[
A(x) = 1 + 2x + 3x^2 = a_0 + a_1 x + a_2 x^2 \\
B(x) = 2 - x + 4x^2 = b_0 + b_1 x + b_2 x^2
\]

Goal: Compute their product:

\[
C(x) = A(x)B(x) = (1 + 2x + 3x^2)(2 - x + 4x^2)
\]

\[
= 2 + 3x + 8x^2 + 5x^3 + 12x^4
\]

\[
= c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4
\]

\[
c_0 = a_0 b_0, \quad c_1 = a_0 b_1 + a_1 b_0, \ldots
\]

In general, given the coefficients:

\[
a = (a_0, a_1, \ldots, a_D) \quad \& \quad b = (b_0, b_1, \ldots, b_D)
\]

for polynomials

\[
A(x) = \sum_{i=0}^{D} a_i x^i \quad \& \quad B(x) = \sum_{i=0}^{D} b_i x^i
\]

we want to compute the coefficients

\[
c = (c_0, c_1, \ldots, c_{2D}) \quad \text{for} \quad C(x) = \sum_{i=0}^{2D} c_i x^i = A(x)B(x)
\]

where

\[
c_k = a_0 b_k + a_1 b_{k-1} + \ldots + a_k b_0
\]

\[
= \sum_{i=0}^{k} a_i b_{k-i}
\]
Vector c is called the convolution of vectors a & b.
Naïve approach: \( O(n) \) time for \( \mathbf{c}_k \) & then \( O(d^2) \) total time.
Using FFT: \( O(d \log d) \) total time.

Two ways to represent a polynomial \( A(x) = a_0 + a_1 x + \cdots + a_d x^d \):
1) coefficients: \( a_0, a_1, \ldots, a_d \)
or 2) values: \( A(x_0), A(x_1), \ldots, A(x_d) \)

**Lemma:** A degree \( d \) polynomial is uniquely characterized by its values at any \( d+1 \) distinct points.
(example: line has \( d=1 \) & is defined by 2 points)

We assume input/output is in coefficients representation but the values representation is more useful for multiplying polynomials.

Given \( A(x_0), \ldots, A(x_{2d}) \) & \( B(x_0), \ldots, B(x_{2d}) \) then \( C(x_i) = A(x_i)B(x_i) \) takes \( O(1) \) time per \( i \), \( O(d) \) choices of \( i \) \( \Rightarrow O(d^2) \) total time.

\( \& \) \( C(x_0), \ldots, C(x_{2d}) \) defines \( C(x) \).
**FFT:** converts between coefficients $\leftrightarrow$ values in $O(d \log d)$ time
does it for carefully chosen set of points.

Consider polynomial $A(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$
where $n$ is a power of 2.

Given $a = (a_0, a_1, \ldots, a_{n-1})$
we want to output $A(x_1), \ldots, A(x_{2n})$
for $2n$ points that we choose.
How should we choose these points?

**Key idea:** Suppose we have $n$ points $x_1, \ldots, x_n$
& the other $n$ points are $x_{n+1} = -x_1, \ldots, x_{2n} = -x_n$
So the $2n$ points are $\pm x_1, \pm x_2, \ldots, \pm x_n$

Note $A(x_i)$ & $A(-x_i)$ are the same for the even terms
and opposite for odd terms.

So let's split $A(x)$ into
even and odd terms.
For $A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_{n-1} x^{n-1}$

let $A_{\text{even}}(y) = a_0 + a_2 y + a_4 y^2 + \ldots + a_{n-2} y^{(n-2)/2}$

So $a_{\text{even}} = (a_0, a_2, a_4, \ldots, a_{n-2})$

and $A_{\text{odd}}(y) = a_1 + a_3 y + a_5 y^2 + \ldots + a_{n-1} y^{(n-2)/2}$

So $a_{\text{odd}} = (a_1, a_3, a_5, \ldots, a_{n-1})$

Notice that: $A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)$

Hence, $A(x_i) = A_{\text{even}}(x_i^2) + x_i A_{\text{odd}}(x_i^2)$

and $A(x_{n+i}) = A(-x_i) = A_{\text{even}}(x_i^2) - x_i A_{\text{odd}}(x_i^2)$

So given $A_{\text{even}}(y_i), \ldots, A_{\text{even}}(y_n)$ & $A_{\text{odd}}(y_i), \ldots, A_{\text{odd}}(y_n)$

for $y_i = x_i^2, \ldots, y_n = x_n^2$

Then we get in $O(n)$ time

$A(x_i), \ldots, A(x_n), A(x_{n+i}), \ldots, A(x_{2n})$

$A(-x_i), \ldots, A(-x_n)$
$A_{even}(y) \& A_{odd}(y)$ are of degree $\frac{n-2}{2} = \frac{n}{2} - 1$

whereas $A(x)$ has degree $n-1$.

So to solve the problem of evaluating $A(x)$ of degree $n-1$ at $2n$ points, we need to know $A_{even}(y)$ and $A_{odd}(y)$ which are of degree $\frac{n}{2} - 1$ at $n$ points.

$\Rightarrow$ Divide & conquer

to solve the problem for size $n$, need to solve 2 subproblems of size $\frac{n}{2}$

$\Rightarrow T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n)$.

What about the next round? Need that $x_1^2, x_2^2, \ldots, x_n^2$ are $\pm$ pairs so:

\[
x_1^2 = -\left(\frac{x_1^2}{2} + 1\right)
x_2^2 = -\left(\frac{x_2^2}{2} + 2\right)
\vdots
x_n^2 = -\left(\frac{x_n^2}{2} + n\right)
\]

but these are both $\geq 0$ so it's impossible unless we use complex numbers.
Review of complex numbers:

$a + bi$ represented as $(a, b)$ in the complex plane or $(r, \theta)$ in polar coordinates.

For polar $(r, \theta)$:

$$z = r (\cos \theta + i \sin \theta) = re^{i\theta}$$

Euler's formula

Polar is convenient:

Multiplying: $(r_1, \theta_1) \times (r_2, \theta_2) = (r_1r_2, \theta_1 + \theta_2)$

So if $r = 1$, for $z = (1, \theta)$

then $z^n = (1, n\theta)$

Since $-1 = (1, \pi)$

then if $z = (r, \theta)$ then $-z = (r, \theta + \pi)$

The $n^{th}$ complex roots of unity are those $z$ where $z^n = 1$

Thus, they are $z = (1, \theta)$ where $z^n = (1, 2\pi j)$

thus $\theta = \frac{2\pi j}{n}$ where $j = 0, 1, \ldots, n-1$. 
Let's look at the $n$th complex roots of unity
for $n$ which is a power of 2 so $n=2^k$ for some $k$.

$n=2$:

$n=4$:

$n=8$:

Let $w = (1, \frac{2\pi}{n}) = e^{2\pi i/n}$

Then the $n$th roots of unity are $1, w, w^2, \ldots, w^{n-1}$.

Note: $w^j = (1, \frac{2\pi}{n}i) = e^{2\pi i/n}$ for $j=0,1,2,\ldots,n-1$.

Two key properties:

1) Satisfy the property $w^j = -w^{n/2 + j}$ for $j=0,1,\ldots,n/2-1$.

So the $1^{st} \frac{n}{2}$ of the $n$th roots of $n$th roots:

$1 = -w^{n/2}$

$w = -w^{n/2+1}$

$\vdots$

$w^{n-1} = -w^n$.
2) Look at the square of the \( n \)th roots:
\[
(1)^2, (\omega)^2, (\omega^2)^2, (\omega^3)^2, \ldots, (\omega^{n-1})^2
\]
\[
(\omega^i)^2 = \left(1, \frac{2\pi i}{n}\right) \times \left(1, \frac{2\pi i}{n}\right) = \left(1, \frac{2\pi i}{n^2}\right) = \text{the } i^{th} \text{ root of the } \frac{1}{2} \text{ root}
\]
\[
(\omega^{\frac{n}{2}})^2 = (-\omega)^2 = (\omega)^2 \Rightarrow
\]
So if we square the \( n \)th roots we get the \( \frac{1}{2}n \)th roots.

If we want to evaluate a polynomial
\[
A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}
\]
at the \( n \)th roots of unity

Then let 
\[
\text{Aeven}(y) = a_0 + a_2y + a_4y^2 + \ldots + a_{n-2}y^{(n-2)/2}
\]
\[
\text{Aodd}(y) = a_1 + a_3y + a_5y^2 + \ldots + a_{n-1}y^{(n-1)/2}
\]

Note: \( A(x) = \text{Aeven}(x^2) + x \text{Aodd}(x^2) \)

Degree of \( A(x) \) is \( n-1 \) & degrees of \( \text{Aeven}(y) \& \text{Aodd}(y) \) is \( \frac{n}{2} - 1 \)

To get \( A(x) \) at \( n \)th roots, need \( \text{Aeven}(y) \& \text{Aodd}(y) \) at \( \frac{n}{2} \)th roots

So 2 subproblems of half the size

FFT: takes coefficients of poly \( A(x) \)
& outputs the value of \( A(x) \) at the \( n \)th roots.
FFT algorithm:

Let \( w = e^{2\pi i/n} \)

**FFT(a, w):**

*Input:* vector \( a = (a_0, a_1, \ldots, a_{n-1}) \) which are coefficients for polynomial \( A(x) \) of degree \( \leq n-1 \) where \( n \) is a power of 2, & \( w \) is a \( n \)th root of unity

*Output:* \( A(w^0), A(w), A(w^2), \ldots, A(w^{n-1}) \)

if \( n = 1 \), return \( A(1) \)

let \( a_{\text{even}} = (a_0, a_2, \ldots, a_{n-2}) \) & \( a_{\text{odd}} = (a_1, a_3, \ldots, a_{n-1}) \)

\((s_0, s_1, \ldots, s_{n/2-1}) = \text{FFT}(a_{\text{even}}, w^2)\)

\((t_0, t_1, \ldots, t_{n/2-1}) = \text{FFT}(a_{\text{odd}}, w^2)\)

For \( j = 0 \rightarrow \frac{n}{2} - 1:\)

\[ r_j = s_j + w^j t_j \]

\[ r_{\frac{n}{2} + j} = s_j - w^j t_j \]

Return \( (r_0, r_1, \ldots, r_{n-1}) \)

**Running time:** \( T(n) = 2T\left(\frac{n}{2}\right) + O(n) \)

\[ = O(n \log n) \]
Original problem:

Given \( a = (a_0, a_1, \ldots, a_{n-1}) \) & \( b = (b_0, b_1, \ldots, b_{n-1}) \) which are coefficients for poly \( A(x) \) & \( B(x) \),

compute coefficients \( c = (c_0, \ldots, c_{2n-2}) \) for \( C(x) = A(x)B(x) \).

So we run \( \text{FFT}(a, w) \) & \( \text{FFT}(b, w) \)

for \( w = e^{-2\pi i / 2n} \), the \( 2n \)th root of unity

to get \( A(x) \) & \( B(x) \) at \( x = w^j \) for \( j = 0, 1, \ldots, 2n-1 \)

then \( C(w^j) = A(w^j)B(w^j) \)

So we have \( C(x) \) at \( 2n \) Points

But then we need to interpolate

to get the coefficients for \( C(x) \).

To do that we do a reverse \( \text{FFT} \),

which is just another \( \text{FFT} \).
For points $x_0, x_1, \ldots, x_{n-1}$ notice that:

\[
\begin{bmatrix}
A(x_0) \\
A(x_1) \\
\vdots \\
A(x_{n-1})
\end{bmatrix} =
\begin{bmatrix}
1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\
1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\
\vdots \\
1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{n-1}
\end{bmatrix}
\]

For FFT we have $x_j = w^j$ so:

\[
\begin{bmatrix}
A(1) \\
A(w) \\
A(w^2) \\
\vdots \\
A(w^{n-1})
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & w & w^2 & \cdots & w^{n-1} \\
1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\
\vdots \\
1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{n(n-1)}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_{n-1}
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & w & w^2 & \cdots & w^{n-1} \\
1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\
\vdots \\
1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{n(n-1)}
\end{bmatrix}
\]

So:

\[
A = M_n(w) a
\]

Multiplying $a$ by $M_n(w)$ gives $A$

But we want to go from $A$ to $a$

So we need $M_n(w)^{-1}$

\[
\text{Lemma: } M_n(w)^{-1} = \frac{1}{n} M_n(w^{-1})
\]
What's $w^{-1}$? $w^{-1} = w^{n-1}$ because:

$$w^{n-1} \times w = w^n = 1$$

Then to compute $M_n(w)^{-1} A$ we just run $\text{FFT}(A, w^{n-1})$ this gives:

$$M_n(w^{-1}) A = M_n(w)^{-1} A$$ which is what we want.