CS 161: Design and Analysis of Algorithms
Divide & Conquer III: Multiplication/FFT

- Divide & Conquer integer multiplication, revisited
- Polynomials
- FFT
Divide & Conquer Multiplication

• Recall our algorithm:
  – Write $x = b^{n/2} x_1 + x_0$, $y = b^{n/2} y_1 + y_0$
  – Need to compute $xy = b^n x_1 y_1 + b^{n/2}(x_1 y_0 + x_0 y_1) + x_0 y_0$
  – $x_1 y_0 + x_0 y_1 = (x_0 + x_1)(y_0 + y_1) - x_1 y_1 - x_0 y_0$
  – Probably can’t reduce to two multiplications
  – What if we we make smaller subproblems?
Divide & Conquer Multiplication

• Subproblems of size n/3:
  - \( x = b^{2n/3} x_2 + b^{n/3} x_1 + x_0 \)
  - \( y = b^{2n/3} y_2 + b^{n/3} y_1 + y_0 \)
  - \( xy = (b^{2n/3} x_2 + b^{n/3} x_1 + x_0)(b^{2n/3} y_2 + b^{n/3} y_1 + y_0) \)
  - Expand, collect terms with \( b^0, b^{n/3}, b^{2n/3}, b^n, b^{4n/3} \)
  - How many subproblems? 9
  - Running Time: \( T(n) = 9 T(n/3) + O(n) \)
    • Solved by \( T(n) = O(n^2) \)
Integers as Polynomials

- If we want to split into subproblems of size $n/k$, write $x = b^{(k-1)n/k} x_{k-1} + \ldots + b^{n/k} x_1 + x_0$
- Let $B = b^{n/k}$. Then $x = B^{k-1} x_{k-1} + \ldots + B x_1 + x_0$
- Can think of $x$ as a polynomial in $B$, where coefficients are integers in $[0,B)$
- To get polynomial coefficients: groups of $n/k$ digits of $x$
- To get $x$: evaluate polynomial at $B$
Polynomials

• \( P(z) = a_d z^d + \ldots + a_1 z + a_0 \)

• Degree(P) = d

• If \( P(z) \) and \( Q(z) \) have degree at most d, then so does \( P(z)+Q(z) \)

• If \( P(z) \) has degree \( d_1 \) and \( Q(z) \) has degree \( d_2 \), then \( P(z)Q(z) \) has degree \( d_1+d_2 \)
Multiplying Integers

• To multiply two n-digit integers \( x \) and \( y \),
  - Interpret \( x \) and \( y \) as degree \( d \) polynomials \( P \) and \( Q \) with \( (n/(d+1)) \)-digit coefficients
    • \( x = P(B) \), \( y = Q(B) \)
  - Multiply the two polynomials to get \( R(z) = P(z)Q(z) \)
  - Evaluate \( R(z) \) at \( B \)
    • \( R(B) = P(B)Q(B) = xy \)
Multiplying Polynomials

\[
P(z) = \sum_{i=0}^{d} a_i z^i
\]

\[
Q(z) = \sum_{i=0}^{d} b_i z^i
\]

\[
a_i = b_i = 0 \forall i > d
\]

\[
R(z) = P(z)Q(z) = \sum_{i=0}^{2d} \left( \sum_{j=0}^{i} a_j b_{i-j} \right) z^i
\]
Multiplying Polynomials

• Coefficients of R are

\[ \sum_{j=0}^{i} a_j b_{i-j} \]

• 2d such coefficients, O(d) adds/multiplies per coefficient \( \rightarrow (d+1)^2 \) adds/multiplies.
Multiplying Integers

• To multiply two n-digit integers x and y,
  – Interpret x and y as degree d polynomials P and Q with $(n/(d+1))$-digit coefficients
    • $x = P(B)$, $y = Q(B)$
  – Multiply the two polynomials to get $R(z) = P(z)Q(z)$
  – Evaluate $R(z)$ at $B$
    • $R(B) = P(B)Q(B) = xy$
Multiply Integers

• Running Time?
  – Interpret as polynomials: \( O(n) \)
  – Multiply polynomials: \((d+1)^2T(n/(d+1))+O(n)\)
    • \((d+1)^2\) multiplications of \(n/(d+1)\) digit integers
    • \((d+1)^2\) additions of \(n/(d+1)\) digit integers
  – Evaluate polynomial at B: \(O(n)\)
  – \(T(n) = (d+1)^2T(n/(d+1))+O(n)\)
  – \(T(n) = O(n^2)\)
Representing Polynomials

• Generally, polynomials represented by coefficients $a_i$

• Theorem: Let $Z$ be a set of size $d+1$ inputs, and let $P(z)$ be a polynomial of degree $d$. Then $P(z)$ is completely determined by the values $P(z_0)$, $P(z_1)$, ..., $P(z_d)$
Proof

• Let P and Q be polynomials of degree d such that \( P(z_i) = Q(z_i) \) for all \( i \)
• Let \( R(z) = P(z) - Q(z) \)
• \( R(z_i) = 0 \) for all \( i \)
• Fact: If a polynomial of degree at most d has d + 1 zeros, then the polynomial is identically 0
• Thus \( R(z) = 0 \), so \( R(z) = Q(z) \)
Computing Coefficients

• Given \( P(z_0), \ldots, P(z_d) \), can compute coefficients of \( P \)

\[
P(z_0) = a_d z_0^d + a_{d-1} z_0^{d-1} + \ldots + a_1 z_0 + a_0
\]

\[
P(z_1) = a_d z_1^d + a_{d-1} z_1^{d-1} + \ldots + a_1 z_1 + a_0
\]

\[\vdots\]

\[
P(z_d) = a_d z_d^d + a_{d-1} z_d^{d-1} + \ldots + a_1 z_d + a_0
\]
Computing Coefficients

• Given $P(n_0), \ldots, P(n_d)$, can compute coefficients of $P$

\[
\begin{pmatrix}
P(z_0) \\
P(z_1) \\
\vdots \\
P(z_d)
\end{pmatrix}
= 
\begin{pmatrix}
1 & z_0 & \cdots & z_0^d \\
1 & z_1 & \cdots & z_1^d \\
\vdots & \vdots & \ddots & \vdots \\
1 & z_d & \cdots & z_d^d
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_d
\end{pmatrix}
\]
Computing Coefficients

• Given $P(z_0), \ldots, P(z_d)$, can compute coefficients of $P$

\[
\begin{pmatrix}
P(z_0) \\
P(z_1) \\
\vdots \\
P(z_d)
\end{pmatrix} = V_z \begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_d
\end{pmatrix}
\]
Vandermonde Matrix

• $V_Z$ is an invertible matrix

\[
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_d \\
\end{pmatrix}
= V_Z^{-1}
\begin{pmatrix}
P(z_0) \\
P(z_1) \\
\vdots \\
P(z_d) \\
\end{pmatrix}
\]
Multiplying Polynomials

• To multiply polynomials P and Q:
  – Pick a set $Z$ of $2d+1$ inputs
  – Compute $P(z_i), Q(z_i)$
  – Compute $R(z_i) = P(z_i)Q(z_i)$
  – Compute coefficients of $R(z)$
Example: \( d=1 \)

- To multiply two degree 1 polynomials \( P \) and \( Q \):
  - Let \( Z = \{0,1,\infty\} \)
  - Compute \( P(0) = a_0, P(1)=a_0+a_1, P(\infty)=a_1 \)
  - Compute \( Q(0) = b_0, P(1)=b_0+b_1, P(\infty)=b_1 \)
  - Compute \( R(0) = a_0b_0, R(1) = (a_0+a_1)(b_0+b_1), R(\infty)=a_1b_1 \)
Example: $d=1$

$$V^{-1}_z = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$
Example: \( d=1 \)

\[
\begin{pmatrix}
  c_0 \\
  c_1 \\
  c_2 \\
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & 0 \\
  -1 & 1 & -1 \\
  0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
  R(0) \\
  R(1) \\
  R(\infty) \\
\end{pmatrix} =
\begin{pmatrix}
  R(0) \\
  R(1) - R(0) - R(\infty) \\
  R(\infty) \\
\end{pmatrix}
\]
Example: d=1

• To multiply two n-digit integers x and y
  – Interpret x and y as degree 1 polynomials P and Q with (n/2)-digit coefficients
    • $P(z) = a_1 z + a_0$, $Q(z) = b_1 z + b_0$
  – Compute $R(0)=a_0b_0$, $R(1)=(a_0+a_1)(b_0+b_1)$, $R(\infty)=a_1b_1$
    • Recursively make 3 n/2-digit multiplications
  – Compute coefficients of R(z):
    • $c_0 = R(0)$, $c_1 = R(1)-R(0)-R(\infty)$, $c_2 = R(\infty)$
  – Evaluate $R(B)=R(b^{n/2})$
Example: d=1

• Running Time?
  – Interpret as polynomials: $O(n)$
  – Multiply polynomials: $3T(n/2)+O(n)$
  – Evaluate polynomial at B: $O(n)$
  – $T(n) = 3T(n/2)+O(n)$
  – $T(n) = O(n^{\log_3 2}) = O(n^{1.585})$
Example: $d=2$

- To multiply two degree 2 polynomials $P$ and $Q$:
  - Let $Z = \{0, 1, -1, -2, \infty\}$
  - Compute $P(0), P(1), P(-1), P(-2), P(\infty)$
  - Compute $Q(0), Q(1), Q(-1), Q(-2), Q(\infty)$
  - Compute $R(0), R(1), R(-1), R(-2), R(\infty)$
Example: $d=2$

\[ V_{Z}^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{3} & -1 & \frac{1}{6} & -2 \\
-1 & \frac{1}{2} & \frac{1}{2} & 0 & -1 \\
-\frac{1}{2} & \frac{1}{6} & \frac{1}{2} & -\frac{1}{6} & 2 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix} \]
Example: $d=2$

\[
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3 \\
c_4
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{3} & -1 & \frac{1}{6} & -2 \\
-1 & \frac{1}{2} & \frac{1}{2} & 0 & -1 \\
-\frac{1}{2} & \frac{1}{6} & \frac{1}{2} & -\frac{1}{6} & 2 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
R(0) \\
R(1) \\
R(-1) \\
R(-2) \\
R(\infty)
\end{pmatrix}
\]
Example: \( d = 2 \)

- To multiply two \( n \)-digit integers \( x \) and \( y \)
  - Interpret \( x \) and \( y \) as degree 2 polynomials \( P \) and \( Q \) with \( (n/3) \)-digit coefficients
  - Compute \( R(0) = P(0)Q(0) \), \( R(1) = P(1)Q(1) \), \( R(-1) = P(-1)Q(-1) \), \( R(-2) = P(-2)Q(-2) \), \( R(\infty) = P(\infty)Q(\infty) \)
    - Recursively make 5 \( n/3 \)-digit multiplications
  - Compute coefficients of \( R(z) \):
  - Evaluate \( R(B) = R(b^{n/3}) \)
Example: $d=2$

- **Running Time:**
  - 5 $n/3$-digit multiplications
  - $O(n)$ extra time
  - $T(n) = 5 \cdot T(n/3) + O(n)$
  - $T(n) = O(n^{\log_3 5}) = O(n^{1.465})$
General $d$

- Make $2d+1$ recursive calls of size $n/(d+1)$
- $T(n) = (2d+1) T(n/(d+1)) + O(n)$
- $T(n) = O(n^{\log_{d+1}(2d+1)})$
- Can make $O(n^{1+\varepsilon})$ for arbitrarily small $\varepsilon$
- Hidden constants grow very rapidly as $\varepsilon$ goes to 0
Observation

• Every recursive call, we:
  – Interpret integers as polynomials
  – Change representation of polynomials
  – Multiply in this representation by making recursive integer multiplication calls
  – Change representation of product back to coefficient representation
  – Evaluate polynomial at the base B
Simplification

• What if instead we:
  – Interpret n-digit integers as degree (n-1) polynomials
  – Change representation of polynomials
  – Multiply polynomials in this representation
  – Change representation back
  – Evaluate polynomial at the base b
Changing Representation

• To change representation of degree d polynomial seems to require $d^2$ operations

\[
\begin{pmatrix}
P(z_0) \\
P(z_1) \\
\vdots \\
P(z_d)
\end{pmatrix} =
\begin{pmatrix}
1 & z_0 & \cdots & z_0^d \\
1 & z_1 & \cdots & z_1^d \\
\vdots & \vdots & \ddots & \vdots \\
1 & z_d & \cdots & z_d^d
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\vdots \\
a_d
\end{pmatrix}
\]

• Idea: can we pick the inputs $z_i$ to make our job easier?
Changing Representation

- Say $d = 2k + 1$

$$P(z) = a_{2k+1}z^{2k+1} + ... + a_0$$

$$= (a_{2k}z^{2k} + a_{2k-2}z^{2k-2} + ... + a_0) + (a_{2k+1}z^{2k+1} + a_{2k-1}z^{2k-1} + ... + a_1z)$$

$$= P_{even}(z^2) + zP_{odd}(z^2)$$

$$P_{even}(z) = a_{2k}z^k + a_{2k-2}z^{k-1} + ... + a_0$$

$$P_{odd}(z) = a_{2k+1}z^k + a_{2k-1}z^{k-1} + ... + a_1z$$
Divide and Conquer

• Let $Z = \{z_0, -z_0, z_1, -z_1, \ldots, -z_k, z_k\}$
• Let $Z' = \{z_0^2, z_1^2, \ldots, z_k^2\}$
• To evaluate $P$ on all the points in $Z$:
  – Evaluate $P_{\text{even}}$ and $P_{\text{odd}}$ on all the points in $Z'$

\[
P(z) = P_{\text{even}}(z^2) + zP_{\text{odd}}(z^2)
\]

\[
P(\pm z_i) = P_{\text{even}}(z_i^2) \pm z_i P_{\text{odd}}(z_i^2)
\]
Divide and Conquer

• To evaluate $P$ on $d+1=2k+2$ points, simply evaluate $P_{\text{even}}$ and $P_{\text{odd}}$ on $k+1$ points each, and then add or subtract results

• $T(d) = 2 \cdot T((d+1)/2) + O(d)$

• Solved with $T(d) = O(d \log d)$
Problem!

• We evaluate $P_{\text{even}}$ and $P_{\text{odd}}$ on $z_i^2$
• To recursively apply this trick, we need the $z_i^2$ values to be in ± pairs
• But if $z_i$ is a real number, $z_i^2$ is always non-negative!
• Must use imaginary/complex numbers
Complex Numbers

• Imaginary number i: $i^2 = 1$
• Complex numbers have the form: $a + bi$
• $(a + bi) + (c + di) = (a + c) + (b + d)i$
• $(a + bi)(c + di) = ac + bc i + ad i + bd i^2$
  
  $= (ac-bd) + (bc+ad) i$
Complex Numbers

• Fact: $e^{i\theta} = \cos(\theta) + i \sin(\theta)$
• $e^{i2\pi} = 1$
• Alternative representation of complex numbers:
  – Re$^{i\theta}$ where R and $\theta$ are real numbers
  – Same representation if we use $\theta+2\pi k$ for any integer $k$
  – $(\text{Re}^{i\theta}) (\text{Se}^{i\varphi})=(\text{RS})e^{i(\theta+\varphi)}$
Complex Numbers

• Roots of unity:
  – Solutions to $z^n = 1$ are called nth roots of unity
  – Clearly, 1 is an nth root of unity. Are there others?
  – $(\text{Re} e^{i\theta})^n = 1 = (1)e^{i(0)}$
  – $R = 1$
  – $\theta n = 0 + 2\pi k$ for some integer $k$
  – $\theta = k \left( \frac{2\pi}{n} \right)$
Complex Numbers

- Roots of unity:
  - $\theta = k \frac{2\pi}{n}$ for some integer $k$
  - i.e., $z = e^{ik\frac{2\pi}{n}}$
  - Can replace $k$ with $k+n$, so only $n$ different values: $k = 0, 1, \ldots, n-1$
Complex Numbers

• Primitive nth root of unity:
  - $z^n = 1$
  - $z^k \neq 1$ for $0 \leq k < n$
  - Example: $e^{i2\pi/n}$
  - Fact: Let $\omega$ be a primitive nth root of unity. Then
    \{1, \omega, \omega^2, \ldots, \omega^{n-1}\} all nth roots of unity, and are all distinct
Complex Numbers

• Fact: Let $\omega$ be a primitive $n$th root of unity. Then \{1, $\omega$, $\omega^2$, ..., $\omega^{n-1}$\} all $n$th roots of unity, and are all distinct
  
  - $(\omega^k)^n = (\omega^n)^k = 1^k = 1$
  
  - If $\omega^k = \omega^{k'}$, assume w.l.o.g. $k < k'$.
  
  - Then $\omega^{k'-k} = 1$
  
  - But $0 < k' - k < n$, so $\omega$ cannot be primitive