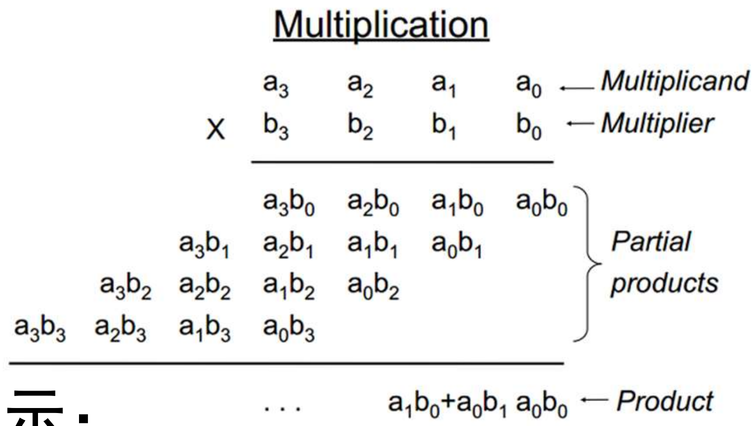
课程实验：大整数乘法

- 计算两个大整数A和B的乘积
  - 为了简化分析，我们假设两个大整数均为N位，例如：
    - $A = 82391748192378492378, B = 23821934892231941738$
    - $C = A \times B$， $C = ?$
- 方法：小学乘法（竖式手算）
  - 时间复杂度为$O(n^2)$



- 我们将大整数以多项式的形式表示：
  - $A(x) = \sum_{j=0}^{N-1} a_j x^j, B(x) = \sum_{j=0}^{N-1} \mathrm{b_j x^j}, C(x) = \sum_{j=0}^{2N-1} c_j x^j$
  - 上述的大整数A, B, C即为多项式$A(x), B(x), C(x)$在$x = 10$处的值

大整数乘法：分治法1

- 我们尝试使用分治法解大整数乘法
  - 例：$A = 2135, B = 4014, C = ?$
  - 解：将A, B均划分为相同长度的两部分（A_1A_2, B_1B_2）
    - $C = A \times B = (21 \times 10^2 + 35) \times (40 \times 10^2 + 14)$
    - $C = 21 \times 40 \times 10^4 + (21 \times 14 + 35 \times 40) \times 10^2 + 35 \times 14$
- 我们定义分治法1：
  - 将一个N位的大整数划分为两个$N/2$位的大整数
    - 令 $A = A_1A_2$，$B = B_1B_2$ (其中A和B为N位整数，A_1, A_2, B_1, B_2 为$N/2$位整数)
  - $A \times B = A_1 \times B_1 \times 10^N + (A_1 \times B_2 + A_2 \times B_1) \times 10^{N/2} + A_2 \times B_2$
  - 时间复杂度：$T(n) = 4T(n/2) + O(n) = O(n^2)$
  - 计算复杂度没有得到改进！如果要改进复杂度，就必须减少子问题数量！

大整数乘法：分治法2

- 方法1中，我们需要计算4次$N/2$位大整数乘法，能否减少？
  - *我们尝试转换系数*$(A_1 \times B_2 + A_2 \times B_1)$
    - $(A_1 \times B_2 + A_2 \times B_1) = (A_1 - A_2) \times (B_2 - B_1) + A_1 \times B_1 + A_2 \times B_2$ ①

    ➔ $A \times B = A_1 \times B_1 \times 10^N + ① \times 10^{N/2} + A_2 \times B_2$
  - 分治法1解$A \times B$的4次乘法转变为3次乘法
- 由此可得改进后的时间复杂度为：
  - $T(n) = 3T\left(\frac{n}{2}\right) + O(n) = O\left(n^{log2^3}\right) \approx n^{1.585}$
- 如果将大整数分成更多段，用更复杂的方式把它们组合起来，将有可能得到更优的算法。

FFT解大整数乘法

□ 多项式的点值表示法

- 例如N次多项式 $A(x)=\sum_{j=0}^{N-1} a_j x^j$ 可以表示为N个点值对所形成的集合

  ➔ $\{(x_0,y_0),(x_1,y_1),(x_2,y_2)\cdots(x_{N-1},y_{N-1})\}$, 其中$y_k=A(x_k)$

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

  - 其中左边的矩阵是范德蒙德矩阵，记作$V(x_0,x_1\cdots x_{n-1})$
  - 如果x_k相异，范德蒙德矩阵可逆 → 系数$(a_0,a_1,\cdots,a_{n-1})^T$有唯一解

- 点值对乘法
  - 记$A(x)=\sum_{j=0}^{N-1} a_j x^j$为$\{(x_0,y_0),(x_1,y_1),(x_2,y_2)\cdots(x_{N-1},y_{N-1})\}$
  - 记$B(x)=\sum_{j=0}^{N-1} b_j x^j$为$\{(x_0,y'_0),(x_1,y'_1),(x_2,y'_2)\cdots(x_{N-1},y'_{N-1})\}$

  ➔ $A(x)\times B(x)$在$\{x_0,x_1,\cdots,x_{n-1}\}$处的值为$\{y_0y'_0,y_1y'_1,\cdots y_{N-1}y'_{N-1}\}$

FFT解大整数乘法: 思路(1)

- 使用点值表示法解多项式乘法$C(x)=A(x)\times B(x)$

$$A(x)=\sum_{j=0}^{N-1} a_j x^j$$
$$B(x)=\sum_{j=0}^{N-1} b_j x^j$$

直接相乘

时间复杂度为$O(n^2)$

$$C(x)=\sum_{j=0}^{2N-1} c_j x^j$$

选择合适的点 $x_0, x_1, \cdots, x_{2N-1}$,求值

$A(x_0), B(x_0)$
$A(x_1), B(x_1)$
$A(x_2), B(x_2)$
$A(x_3), B(x_3)$
$\cdots$
$A(x_{2N-1}), B(x_{2N-1})$

点值对相乘

时间复杂度为$O(n)$

$C(x_0)$
$C(x_1)$
$C(x_2)$
$C(x_3)$
$\cdots$
$C(x_{2N-1})$

（插值）计算系数 $c_0, c_1, \cdots, c_{2N-1}$

➢ 通过使用快速傅里叶变换（FFT），可以使得计算点值对和系数的时间复杂度降为$O(n\log n)$

FFT解大整数乘法: 思路(2)

- 例：利用点值对表示法求$C = 11 \times 22$
  - 多项式表示法： $A(x) = x + 1, B(x) = 2x + 2$ ➔ $11 = A(10), 22 = B(10)$
  - $A(x), B(x)$均为一次多项式（$N = 2$）➔ $C(x)$至多为$2N - 1$次多项式
  - 为了计算简便，我们取$2N$个x值，分别为$0, 1, 2, 3$,并计算对应的$A(x), B(x)$

x	0	1	2	3
$A(x)$	1	2	3	4
$B(x)$	2	4	6	8

  - 由$C(x) = \sum_{j=0}^{2N-1} c_j x^j = A(x) \times B(x)$计算$C(x)$在对应$x$的值

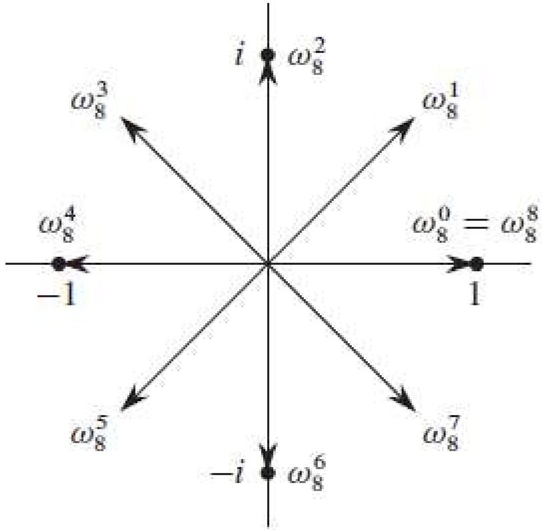
x	0	1	2	3
$C(x)$	2	8	18	32

➔ $\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 2 \\ 0 \end{bmatrix}$ ➔ $11 \times 22 = C(10) = 242$

FFT解大整数乘法: 单位复根

- 欧拉公式：$e^{iu} = cos(u) + i\ sin(u)$
- 单位复根：满足$\omega^n = 1$的复数ω为n次单位复根
  - n次单位复根共有n个，分别为$\omega_n^k = e^{i\,2\pi k/n}, k = 0, 1, 2, \cdots, n-1$
  - n个n次单位复根均匀分布在以复平面的原点为圆心的单位半径的圆周上
  - 性质：
    - 相消引理：$\omega_{dn}^{dk} = \omega_n^k$
    - 折半引理：$(\omega_n^{k+n/2})^2 = \left(\omega_n^k\right)^2$

      $$\omega_n^{k+n/2} = -\omega_n^k$$
  - 我们选取n次单位复根来计算n次多项式的值
  - 我们假设n是2的幂



FFT解大整数乘法: 分治法

- 定义$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$
  - $y = (y_0, y_1, \cdots, y_{n-1})$ 是$a = (a_0, a_1, \cdots, a_{n-1})$的离散傅里叶变换(DFT)
    - 记作$y = \mathrm{DFT}_n(a)$
  - 将系数a_j性质分为两部分：奇数位系数$a^{[1]}$和偶数位系数$a^{[0]}$
    - $a^{[0]} = (a_0, a_2, \cdots, a_{n-2}), a^{[1]} = (a_1, a_3, \cdots, a_{n-1})$
    - $A^{[0]}(x) = \sum_{j=0}^{\frac{n}{2}-1} a_j^{[0]} x^j, A^{[1]}(x) = \sum_{j=0}^{\frac{n}{2}-1} a_j^{[1]} x^j$

    ➔ $A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$

$$y_k = A^{[0]}(\omega_{n/2}^k) + \omega_n^k A^{[1]}(\omega_{n/2}^k) \quad （消去引理）$$

$$y_{k+n/2} = A^{[0]}(\omega_{n/2}^k) - \omega_n^k A^{[1]}(\omega_{n/2}^k) \quad （折半引理）$$

FFT解大整数乘法: FFT

RECURSIVE-FFT(a)

$n = a.length$ // n is a power of 2
if $n == 1$
    return a
$\omega_n = e^{2\pi i/n}$
$\omega = 1$
$a^{[0]} = (a_0, a_2, \ldots, a_{n-2})$
$a^{[1]} = (a_1, a_3, \ldots, a_{n-1})$
$y^{[0]} = \text{RECURSIVE-FFT}(a^{[0]})$
$y^{[1]} = \text{RECURSIVE-FFT}(a^{[1]})$
for $k = 0$ **to** $n/2 - 1$
    $y_k = y_k^{[0]} + \omega\, y_k^{[1]}$
    $y_{k+(n/2)} = y_k^{[0]} - \omega\, y_k^{[1]}$
    $\omega = \omega\, \omega_n$
return y // y is assumed to be a column vector

FFT解大整数乘法: IDFT

IDFT：将点值表示转换成系数表示

- 范德蒙德矩阵的逆矩阵：

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \cdots & \omega^{(n-1)2} \\ \vdots & \vdots & & \vdots \\ 1 & \omega^{n-1} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}^{-1} = \frac{1}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \cdots & \omega^{-(n-1)2} \\ \vdots & \vdots & & \vdots \\ 1 & \omega^{-(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{pmatrix}$$

- IDFT与DFT运算过程一致
  - 将$\boldsymbol{a}$与$\boldsymbol{y}$的位置互换
  - 用$\boldsymbol{\omega_n^{-1}}$代替$\boldsymbol{\omega_n}$，对FFT输出结果乘以$\frac{1}{n}$

使用FFT解大整数乘法：

$$\boldsymbol{C} = \mathbf{IDFT}_{2n}\left(\mathbf{DFT}_{2n}(\boldsymbol{A}) \cdot \mathbf{DFT}_{2n}(\boldsymbol{B})\right)$$

课程实验: FFT解大整数乘法

□ 目标:

1. 实现利用分治法2求解大整数乘法;

2. 能够理解快速傅里叶变换求解大整数乘法的思想，体会其中的分治思想。（选做）

□ 要求:

- 1.语言：C、C++、Python以及Java
- 2.代码：独立完成，需要对主要的部分提供注释
- 3.文档：包括求解思想和运行结果两部分（代码单独提供），求解思想即为完整的分析过程（写出自己的思考为佳），运行结果为对特定数据的运行结果（包括两部分：自己本地测试不少于5组100位以下数据（自拟，结果截图），并将代码提交到Leetcode Multiply Strings（https://leetcode.com/problems/multiply-strings/）测试运行，将Submission Detail截图）
- 4.提交：将代码（只要包含代码的文件）和文档提交
- 5.命名：代码和文档打包为：实验一+学号+姓名