- 书面作业讲解
  - TC第28.1节练习2、3、6、7
  - TC第28.2节练习1、2、3
  - TC第28.3节练习1、3
  - TC第28章问题1

TC第28.1节练习3

- 答案：$(-3/19, -1/19, 59/19)^T$

TC第28.1节练习6

- n>1时：

$$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 0 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 0 \end{pmatrix}$$

- n=1时：

$$(0) = (1) \cdot (0)$$

TC第28.1节练习7

- k=n时到底需不需要执行最外层循环？

```
LU-DECOMPOSITION(A)
n = A.rows
let L and U be new n × n matrices
initialize U with 0s below the diagonal
initialize L with 1s on the diagonal and 0s above the diagonal
for k = 1 to n
    u_kk = a_kk
    for i = k + 1 to n
        l_ik = a_ik / u_kk          // l_ik holds v_i
        u_ki = a_ki                 // u_ki holds w_i^T
    for i = k + 1 to n
        for j = k + 1 to n
            a_ij = a_ij − l_ik u_kj
return L and U
```

```
LUP-DECOMPOSITION(A)
n = A.rows
let π[1..n] be a new array
for i = 1 to n
    π[i] = i
for k = 1 to n
    p = 0
    for i = k to n
        if |a_ik| > p
            p = |a_ik|
            k' = i
    if p == 0
        error "singular matrix"
    exchange π[k] with π[k']
    for i = 1 to n
        exchange a_ki with a_k'i
    for i = k + 1 to n
        a_ik = a_ik / a_kk
        for j = k + 1 to n
            a_ij = a_ij − a_ik a_kj
```

TC第28.2节练习1

- 利用矩阵平方来得到矩阵乘法

$$D = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}$$

$$D^2 = \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}\begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} A^2 & AB \\ 0 & 0 \end{pmatrix}$$

TC第28.2节练习3

- 矩阵乘法→求行列式
  - 求行列式 ≤ LUP分解 $\det(A) = \det(P^{-1})\det(L)\det(U) = (-1)^S\left(\prod_{i=1}^{n} l_{ii}\right)\left(\prod_{i=1}^{n} u_{ii}\right)$
  - LUP分解 ≤ 矩阵乘法 练习28.2-2
- 求行列式→矩阵乘法
  - 矩阵乘法 ≤ 求逆矩阵 定理28.1
  - 求逆矩阵 ≤ 求行列式+求伴随矩阵 $\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\operatorname{adj}(\mathbf{A})$
  - 求伴随矩阵 ≤ 求行列式 derivative inequality

- 教材讨论
  - TC第30章

问题1：多项式的表示

- 什么是coefficient representation？
  什么是point-value representation？
- 一个cr可以对应几个pvr？
  一个pvr可以对应几个cr？
  为什么？

问题1：多项式的表示(续)

- 基于这两种表示
  - 以下运算的时间复杂度是多少？
  - 你能基于此对比它们的优劣吗？

	coefficient representation	point-value representation
加法的时间		
乘法的时间		
求值的时间		

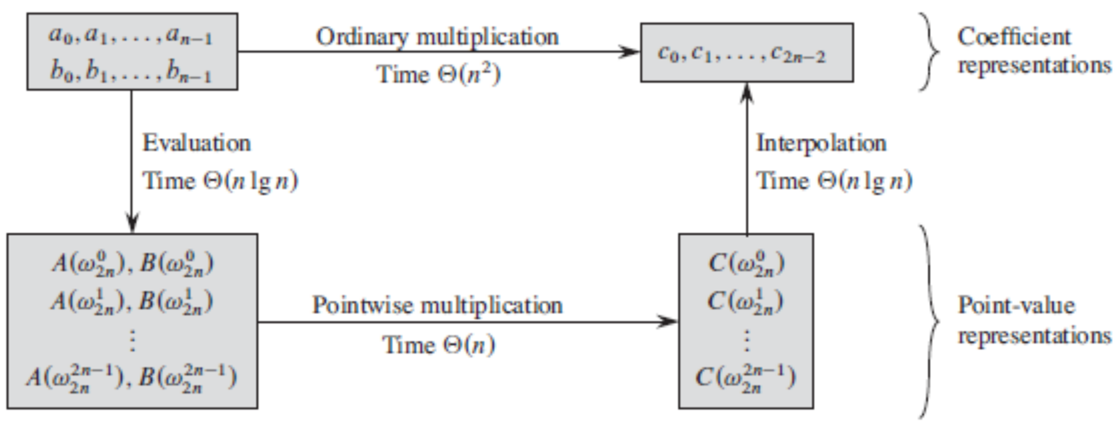
问题1：多项式的表示(续)

- 基于这两种表示
  - 以下运算的时间复杂度是多少？
  - 你能基于此对比它们的优劣吗？

	coefficient representation	point-value representation
加法的时间	$\Theta(n)$	$\Theta(n)$
乘法的时间	$\Theta(n^2)$	$\Theta(n)$
求值的时间	$\Theta(n)$	?

问题2：表示的转换

- 你理解这个流程了吗？
  - 目的是什么？
  - 手段是什么？



问题2：表示的转换(续)

- DFT和FFT分别是什么意思？
  它们之间是什么关系？
- 你能阐述FFT的基本思路吗？
  - 目标是求什么？
  - 用什么策略来求？
  - halving lemma在这里起了什么作用？

问题2：表示的转换 (续)

- DFT和FFT分别是什么意思？它们之间是什么关系？

$$\begin{aligned} y_k &= A(\omega_n^k) \\ &= \sum_{j=0}^{n-1} a_j \omega_n^{kj} \end{aligned}$$

- 你能阐述FFT的基本思路吗？
  - 目标是求什么？

    $$A(x) = \sum_{j=0}^{n-1} a_j x^j \qquad \omega_n^0, \omega_n^1, \omega_n^2, \ldots, \omega_n^{n-1}$$

  - 用什么策略来求？

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

    $$\begin{aligned} A^{[0]}(x) &= a_0 + a_2x + a_4x^2 + \cdots + a_{n-2}x^{n/2-1} \\ A^{[1]}(x) &= a_1 + a_3x + a_5x^2 + \cdots + a_{n-1}x^{n/2-1} \end{aligned}$$

  - halving lemma在这里起了什么作用？

问题2：表示的转换 (续)

- 递归FFT的运行时间如何递归表示？
- 你还记得怎么解这种递归式吗？

```
RECURSIVE-FFT(a)
n = a.length                // n is a power of 2
if n == 1
    return a
ω_n = e^{2πi/n}
ω = 1
a^[0] = (a_0, a_2, ..., a_{n-2})
a^[1] = (a_1, a_3, ..., a_{n-1})
y^[0] = RECURSIVE-FFT(a^[0])
y^[1] = RECURSIVE-FFT(a^[1])
for k = 0 to n/2 - 1
    y_k = y_k^[0] + ω y_k^[1]
    y_{k+(n/2)} = y_k^[0] - ω y_k^[1]
    ω = ω ω_n
return y                    // y is assumed to be a column vector
```

问题2：表示的转换 (续)

- 递归FFT的运行时间如何递归表示？
- 你还记得怎么解这种递归式吗？

```
RECURSIVE-FFT(a)
n = a.length                      // n is a power of 2
if n == 1
    return a
ω_n = e^{2πi/n}
ω = 1
a^[0] = (a_0, a_2, ..., a_{n-2})
a^[1] = (a_1, a_3, ..., a_{n-1})
y^[0] = RECURSIVE-FFT(a^[0])
y^[1] = RECURSIVE-FFT(a^[1])
for k = 0 to n/2 - 1
    y_k = y_k^[0] + ω y_k^[1]
    y_{k+(n/2)} = y_k^[0] - ω y_k^[1]
    ω = ω ω_n
return y                          // y is assumed to be a column vector
```

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

问题2：表示的转换 (续)

- 你理解interpolation的高效解法了吗？

$$a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-kj}$$

vs.

$$\begin{aligned} y_k &= A(\omega_n^k) \\ &= \sum_{j=0}^{n-1} a_j \omega_n^{kj} \end{aligned}$$

问题2：表示的转换 (续)

- 迭代FFT的基本思路是什么？
- 叶子的顺序是如何确定的？
  你能给出直观解释吗？

BIT-REVERSE-COPY (a, A)

```
n = a.length
for k = 0 to n − 1
        A[rev(k)] = a_k
```