Chapter 30

Polynomials, Convolution and the FFT(2)

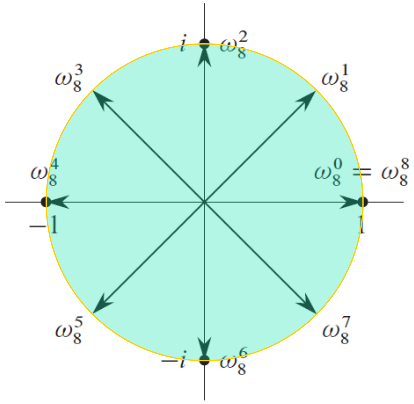
songyou@buaa.edu.cn

Application of DFT

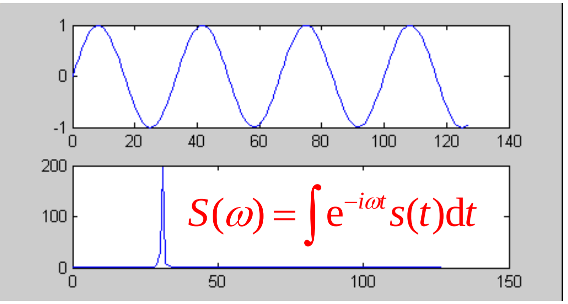
$$\omega_n^k = e^{2\pi ik/n} = \cos(2\pi k/n) + i\sin(2\pi k/n),$$
$$k = 0, 1, \cdots, n-1$$

- wish to evaluate a polynomial $A(x)=\sum_{j=0}^{n-1} a_j x^j$ of degree-bound n at $x = \omega_n^0, \omega_n^1, \omega_n^2, \cdots, \omega_n^{n-1}$.
- Discrete Fourier Transform (DFT, 离散傅里叶变换)

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}, \quad \begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ & & \dots\dots & & \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \dots\dots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix}$$

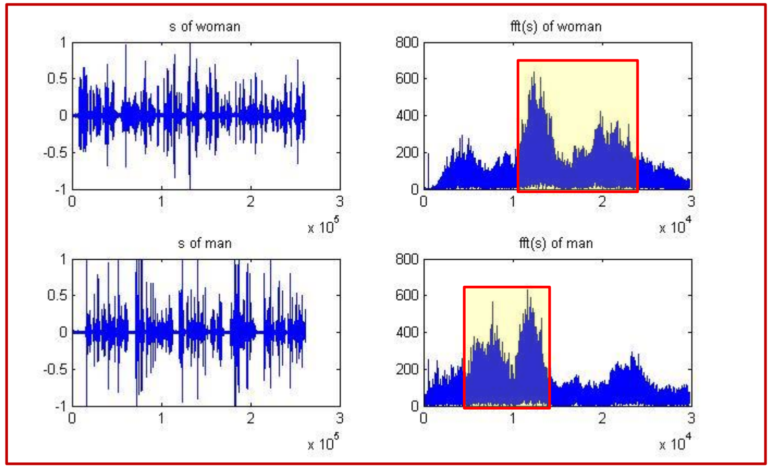


$y = \mathrm{DFT}_n(a)$： 数学意义上，多项式在特殊点的取值（单位复根）



Signal $s(t)$: discrete, $a_i = s(t_i)$

Spcetrum $S(\omega)$: $S(\omega) = \mathrm{DFT}_n(s)$



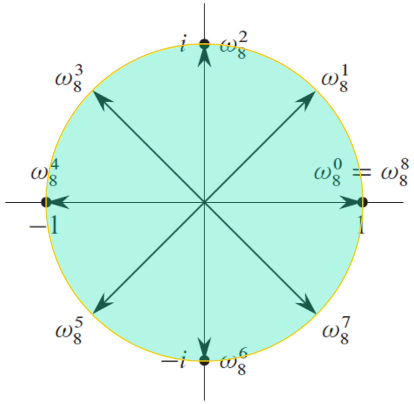
$y = \mathrm{DFT}_n(a)$： 物理意义上，从一个矢量（时域）变换到另一个矢量（频域）

30.2.2 The DFT

$$x_k=(\omega_n)^k=\omega_n^k=\left(e^{\frac{2\pi}{n}i}\right)^k=e^{\frac{2\pi k}{n}i}=\cos(2\pi k/n)+i\sin(2\pi k/n),$$

$$k=0,1,\cdots,n-1$$

$$A(x)=\sum_{j=0}^{n-1}a_jx^j \qquad \begin{pmatrix} y_0\\ y_1\\ \cdots\\ y_{n-1}\end{pmatrix}=\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1}\\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1}\\ & & \ldots\ldots & & \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1}\end{pmatrix}\begin{pmatrix} a_0\\ a_1\\ \cdots\\ a_{n-1}\end{pmatrix}=V(x_0,x_1,\cdots,x_{n-1})\cdot a \qquad (30.4)$$



- wish to evaluate a polynomial $A(x)=\sum_{j=0}^{n-1}a_jx^j$ at $x=\omega_n^0,\omega_n^1,\omega_n^2,\cdots,\omega_n^{n-1}$
- without loss of generality, assume that $n=2^m$, if not, let $a_{n+k}=0$
- Discrete Fourier Transform (DFT): let A is given in coefficient form: $a=(a_0, a_1,\cdots, a_{n-1})^T$, let $x_k=\omega_n^k$, define y_k , for $k=0, 1, \cdots, n-1$, by

$$y_k=A(\omega_n^k)=\sum_{j=0}^{n-1}a_j\omega_n^{kj}\ , \qquad \begin{pmatrix} y_0\\ y_1\\ \cdots\\ y_{n-1}\end{pmatrix}=\begin{pmatrix} 1 & 1 & 1 & \cdots & 1\\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1}\\ & & \ldots\ldots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)}\end{pmatrix}\begin{pmatrix} a_0\\ a_1\\ \cdots\\ a_{n-1}\end{pmatrix}$$

$y=\mathrm{DFT}_n(a)$：数学意义上，多项式在特殊点的取值（单位复根）

Take time $\Theta(n^2)$ to compute straightforward?　慢！如何快速计算？

30.2.3 The FFT

An algorithm for the machine calculation of complex Fourier series

JW Cooley, JW Tukey

Mathematics of computation, 1965 • JSTOR

An efficient method for the calculation of the interactions of a 2'factorial ex-periment was introduced by Yates and is widely known by his name. The generalization to 3'was given by Box et al.[1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an NXN matrix which can be factored into m sparse matrices, where m is proportional to log N. This results inma procedure requiring a number of operations proportional to N log N rather than N2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with N= 2'and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

JSTOR

收起 ^

☆ 保存 99 引用 被引用次数：20530 相关文章 所有 36 个版本 »

30.2.3 The FFT

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-1} x^{n-1}$$

Fast Fourier Transform (FFT，快速傅里叶变换)

- takes advantage of the special properties of the complex roots of unity
- we can compute $\mathrm{DFT}_n(a)$ in time $\Theta(n\lg n)$
- employs a divide-and-conquer strategy,

$$\left.\begin{aligned}&\text{even-index, } A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}\\ &\text{odd-index, } A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}\end{aligned}\right\}$$

多项式：一个变成两个。结构完全一样，求1个n次多项式的值 → 求2个$n/2$次多项式的值，分治！

- $A[0]$ contains all the even-index coefficients of A (the binary representation of the index ends in 0)
- $A[1]$ contains all the odd-index coefficients (the binary representation of the index ends in 1). It follows that

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2) \qquad (30.9)$$

30.2.3 The FFT

- Fast Fourier Transform (FFT): employs a divide-and-conquer strategy,

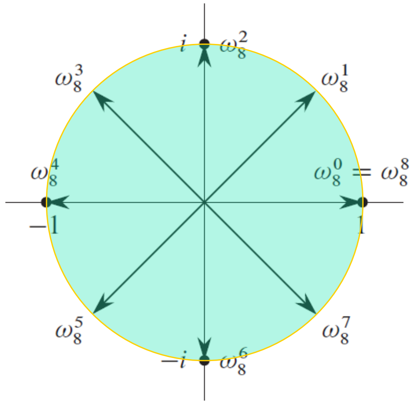
$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

even-index, $A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$

odd-index, $A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2) \qquad (30.9)$$

$$A(\omega_n^k) = A^{[0]}((\omega_n^k)^2) + \omega_n^k A^{[1]}((\omega_n^k)^2)$$



- the problem of evaluating $A(x)$ at $\omega_n^0, \omega_n^1, \omega_n^2, \cdots, \omega_n^{n-1}$ reduces to

(1) evaluating the $A^{[0]}(x)$ and $A^{[1]}(x)$ at the points

$$\boxed{\begin{matrix} (\omega_n^0)^2, & (\omega_n^1)^2, & \cdots, & (\omega_n^{n/2-1})^2 \\ (\omega_n^{0+n/2})^2, & (\omega_n^{1+n/2})^2, & \cdots, & (\omega_n^{n/2-1+n/2})^2 \end{matrix}} \qquad (32.10)$$

and then $\omega_{n/2}^0, \omega_{n/2}^1, \ldots, \omega_{n/2}^{n/2-1}$

(2) combining the results according to equation (30.9).

$A^{[0]}(x)$ 在这 $n/2$ 个点的值，就是 (a_0, a_2, a_{n-2}) 的 DFT。$A^{[1]}(x)$ 同理。

30.2.3 The FFT

Fast Fourier Transform (FFT): divide-and-conquer strategy

even-index, $A^{[0]}(x) = a_0+a_2x+a_4x^2+\cdots+a_{n-2}x^{n/2-1}$

odd-index, $A^{[1]}(x) = a_1+a_3x+a_5x^2+\cdots+a_{n-1}x^{n/2-1}$

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

$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2) \qquad (30.9)$$

$$A(x_r) = A^{[0]}(x_r^{\,2}) + x_r A^{[1]}(x_r^{\,2})$$

$$= A(\omega_n^r) = A^{[0]}((\omega_n^r)^2) + (\omega_n^r)\cdot A^{[1]}((\omega_n^r)^2)$$

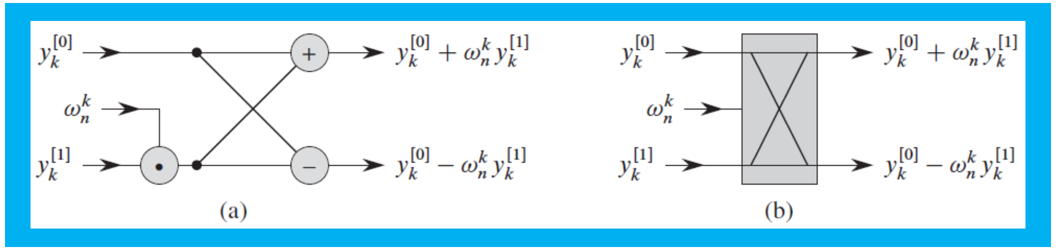
$$= A^{[0]}(\omega_{n/2}^r) + (\omega_n^r)\cdot A^{[1]}(\omega_{n/2}^r),\ r = 0,\ \ldots,\ n/2-1,\ n/2,\ \ldots,\ n-1$$

let $k = 0,\ \ldots,\ n/2-1$, then

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + (\omega_n^k)\cdot A^{[1]}(\omega_{n/2}^k),$$

$$A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^{n/2+k}) + (\omega_n^{n/2+k})\cdot A^{[1]}(\omega_{n/2}^{n/2+k})$$

$$= A^{[0]}(\omega_{n/2}^k) - (\omega_n^k)\cdot A^{[1]}(\omega_{n/2}^k)$$



30.2.3 The FFT

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

$$\begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \cdots\cdots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix}$$

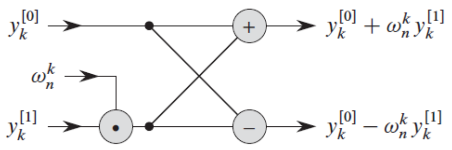
even: $A^{[0]}(x) = a_0+a_2x+a_4x^2+ \cdots +a_{n-2}x^{n/2-1}$

odd: $A^{[1]}(x) = a_1+a_3x+a_5x^2+ \cdots +a_{n-1}x^{n/2-1}$

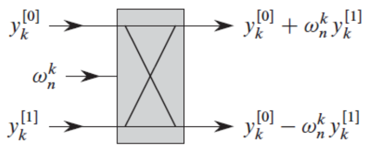
$$A(x) = A^{[0]}(x^2) + xA^{[1]}(x^2)$$

let $k=0, \ldots, n/2-1$, then

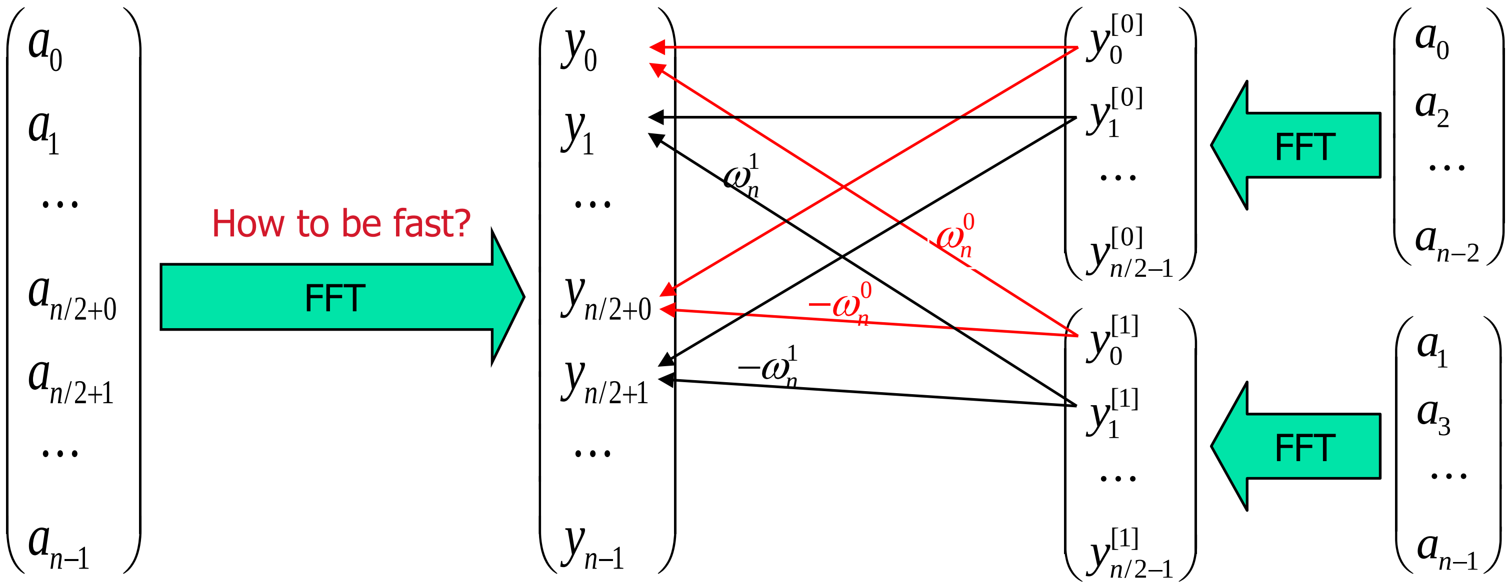
$$y_k = A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + (\omega_n^k)\cdot A^{[1]}(\omega_{n/2}^k)$$
$$= y_k^{[0]} + (\omega_n^k)\cdot y_k^{[1]}$$
$$y_{k+n/2} = A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^{n/2+k}) + (\omega_n^{n/2+k})\cdot A^{[1]}(\omega_{n/2}^{n/2+k})$$
$$= y_k^{[0]} - (\omega_n^k)\cdot y_k^{[1]}$$



(a)



(b)



30.2.3 The FFT

$$\text{even, } A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$$

$$\text{odd, } A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$$

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

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

$$\omega_n^k \qquad (\omega_n^k)^2 = \omega_{n/2}^k$$

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$

$$A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^k) - (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$

$$k = 0, \ldots, n/2-1$$

```
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
```

$$(\omega_n^0)^2, (\omega_n^1)^2, \cdots, (\omega_n^{n/2-1})^2$$

$$(\omega_n^{0+n/2})^2, (\omega_n^{1+n/2})^2, \cdots, (\omega_n^{n/2-1+n/2})^2$$

$$\omega_{n/2}^0, \omega_{n/2}^1, \ldots, \omega_{n/2}^{n/2-1}$$

30.2.3 The FFT

even, $A^{[0]}(x)=a_0+a_2x+a_4x^2+\cdots+a_{n-2}x^{n/2-1}$

odd, $A^{[1]}(x)=a_1+a_3x+a_5x^2+\cdots+a_{n-1}x^{n/2-1}$

$$A(x)=\sum_{j=0}^{n-1}a_jx^j$$

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

$$\omega_n^k \qquad (\omega_n^k)^2=\omega_{n/2}^k$$

```
RECURSIVE-FFT(a)
 1  n = a.length
 2  if n == 1
 3      return a
 4  ω_n = e^{2πi/n}
 5  ω = 1
 6  a^[0] = (a_0, a_2, ..., a_{n-2})
 7  a^[1] = (a_1, a_3, ..., a_{n-1})
 8  y^[0] = RECURSIVE-FFT(a^[0])
 9  y^[1] = RECURSIVE-FFT(a^[1])
10  for k = 0 to n/2 - 1
11      y_k = y_k^[0] + ω y_k^[1]
12      y_{k+(n/2)} = y_k^[0] - ω y_k^[1]
13      ω = ω ω_n
14  return y
```

$$(\omega_n^0)^2,(\omega_n^1)^2,\cdots,(\omega_n^{n/2-1})^2$$

$$(\omega_n^{0+n/2})^2,(\omega_n^{1+n/2})^2,\cdots,(\omega_n^{n/2-1+n/2})^2$$

$$\omega_{n/2}^0,\ \omega_{n/2}^1,\ \ldots,\ \omega_{n/2}^{n/2-1}$$

Line 2-3

$$y_0=a_0\omega_1^0=a_0\cdot 1=a_0$$

30.2.3 The FFT

even, $A^{[0]}(x)=a_0+a_2x+a_4x^2+\cdots+a_{n-2}x^{n/2-1}$

odd, $A^{[1]}(x)=a_1+a_3x+a_5x^2+\cdots+a_{n-1}x^{n/2-1}$

$$A(x)=\sum_{j=0}^{n-1}a_jx^j$$

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

ω_n^k $\qquad (\omega_n^k)^2=\omega_{n/2}^k$

$$(\omega_n^0)^2,(\omega_n^1)^2,\cdots,(\omega_n^{n/2-1})^2$$

$$(\omega_n^{0+n/2})^2,(\omega_n^{1+n/2})^2,\cdots,(\omega_n^{n/2-1+n/2})^2$$

$$\omega_{n/2}^0,\omega_{n/2}^1,\ldots,\omega_{n/2}^{n/2-1}$$

RECURSIVE-FFT(a)

```
1  n = a.length
2  if n == 1
3      return a
4  ω_n = e^{2πi/n}
5  ω = 1
6  a^[0] = (a_0, a_2, ..., a_{n-2})
7  a^[1] = (a_1, a_3, ..., a_{n-1})
8  y^[0] = RECURSIVE-FFT(a^[0])
9  y^[1] = RECURSIVE-FFT(a^[1])
10 for k = 0 to n/2 - 1
11     y_k = y_k^[0] + ω y_k^[1]
12     y_{k+(n/2)} = y_k^[0] - ω y_k^[1]
13     ω = ω ω_n
14 return y
```

Line 8-9

$$y_k^{[0]}=A^{[0]}(\omega_{n/2}^k)=A^{[0]}(\omega_n^{2k}),$$

$$y_k^{[1]}=A^{[1]}(\omega_{n/2}^k)=A^{[1]}(\omega_n^{2k}).$$

Line 11

$$y_k=y_k^{[0]}+\omega_n^k y_k^{[1]}$$

$$=A^{[0]}(\omega_n^{2k})+\omega_n^k A^{[1]}(\omega_n^{2k})=A(\omega_n^k)$$

30.2.3 The FFT

even, $A^{[0]}(x)=a_0+a_2x+a_4x^2+\cdots+a_{n-2}x^{n/2-1}$

odd, $A^{[1]}(x)=a_1+a_3x+a_5x^2+\cdots+a_{n-1}x^{n/2-1}$

$$A(x)=\sum_{j=0}^{n-1}a_jx^j$$

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

ω_n^k $\qquad (\omega_n^k)^2=\omega_{n/2}^k$

$(\omega_n^0)^2, (\omega_n^1)^2, \cdots, (\omega_n^{n/2-1})^2$

$(\omega_n^{0+n/2})^2, (\omega_n^{1+n/2})^2, \cdots, (\omega_n^{n/2-1+n/2})^2$

$\omega_{n/2}^0, \omega_{n/2}^1, \ldots, \omega_{n/2}^{n/2-1}$

RECURSIVE-FFT(a)

```
1   n = a.length
2   if n == 1
3       return a
4   ω_n = e^{2πi/n}
5   ω = 1
6   a^[0] = (a_0, a_2, ..., a_{n-2})
7   a^[1] = (a_1, a_3, ..., a_{n-1})
8   y^[0] = RECURSIVE-FFT(a^[0])
9   y^[1] = RECURSIVE-FFT(a^[1])
10  for k = 0 to n/2 - 1
11      y_k = y_k^[0] + ω y_k^[1]
12      y_{k+(n/2)} = y_k^[0] - ω y_k^[1]
13      ω = ω ω_n
14  return y
```

Line 12

$$\begin{aligned} y_{k+n/2} &= y_k^{[0]}-\omega_n^k y_k^{[1]} \\ &= A^{[0]}(\omega_n^{2k})+\omega_n^{k+n/2}A^{[1]}(\omega_n^{2k}) \\ &= A^{[0]}(\omega_n^{2k+n})+\omega_n^{k+n/2}A^{[1]}(\omega_n^{2k+n}) \\ &= A(\omega_n^{k+n/2}) \end{aligned}$$

30.2.3 The FFT

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

even, $A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$

odd, $A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$

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

$$\omega_n^k \qquad (\omega_n^k)^2 = \omega_{n/2}^k$$

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$

$$A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^k) - (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$

$$k = 0, \ldots, n/2 - 1$$

```
RECURSIVE-FFT(a)
1   n = a.length
2   if n == 1
3       return a
4   ω_n = e^(2πi/n)
5   ω = 1
6   a^[0] = (a_0, a_2, ..., a_(n-2))
7   a^[1] = (a_1, a_3, ..., a_(n-1))
8   y^[0] = RECURSIVE-FFT(a^[0])
9   y^[1] = RECURSIVE-FFT(a^[1])
10  for k = 0 to n/2 - 1
11      y_k = y_k^[0] + ω y_k^[1]
12      y_(k+(n/2)) = y_k^[0] - ω y_k^[1]
13      ω = ω ω_n
14  return y
```

in line 11-12, each $y_k^{[1]}$ is multiplied by ω_n^k, the product is both *added to* and *substracted* from $y_k^{[0]}$. each ω_n^k is used in both its positive and negative forms, ω_n^k is called *twiddle factors*.

旋转因子

Running Time?

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

30.2.3 The FFT

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

even, $A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$

odd, $A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$

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

$\omega_n^k \qquad (\omega_n^k)^2 = \omega_{n/2}^k$

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$
$$A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^k) - (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$
$$k = 0, \ldots, n/2 - 1$$

RECURSIVE-FFT(a)

```
1  n = a.length
2  if n == 1
3      return a
4  ω_n = e^{2πi/n}
5  ω = 1
6  a^[0] = (a_0, a_2, ..., a_{n-2})
7  a^[1] = (a_1, a_3, ..., a_{n-1})
8  y^[0] = RECURSIVE-FFT(a^[0])
9  y^[1] = RECURSIVE-FFT(a^[1])
10 for k = 0 to n/2 - 1
11     y_k = y_k^[0] + ω y_k^[1]
12     y_{k+(n/2)} = y_k^[0] - ω y_k^[1]
13     ω = ω ω_n
14 return y
```

in line 11-12, each $y_k^{[1]}$ is multiplied by ω_n^k, the product is both *added to* and *substracted* from $y_k^{[0]}$. each ω_n^k is used in both its positive and negative forms, ω_n^k is called *twiddle factors*.

旋转因子

Running Time?

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

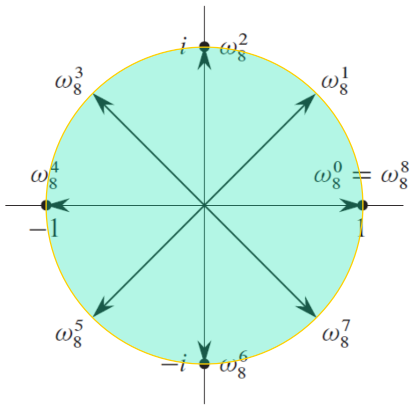
30.2.4 Interpolation at the complex roots of unity

DFT

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \cdots\cdots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix} = V_n a$$

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

$$x_k = \omega_n^k$$



inverse DFT

$$a = \mathrm{DFT}_n^{-1}(y) = V_n^{-1} y$$

coefficient form $a = (a_0, a_1, \cdots, a_{n-1})^T$	Evaluation (DFT) → ← Interpolation (iDFT)	point-value form $\{(x_j, y_j)\}$

Theorem 30.7

For j, k = 0, 1, . . . , n-1, the (j, k) entry of V_n^{-1} is ω_n^{-kj}/n .

$$y = A(x) = \sum_{j=0}^{n-1} a_j x^j$$

30.2.4 Interpolation at the complex roots of unity

- *Lemma* 30.6 (Summation lemma, 求和引理)

  For any integer $n \geq 1$ and nonnegative integer k not divisible by n , ($k \neq m \cdot n$), we have

$$\sum_{j=0}^{n-1} (\omega_n^k)^j = 0.$$

DFT

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \cdots\cdots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix} = V_n a$$

inverse DFT

$$a = \mathrm{DFT}_n^{-1}(y) = V_n^{-1} y$$

- *Theorem* 30.7

  For j, k = 0, 1, . . . , n-1, the (j, k) entry of V_n^{-1} is ω_n^{-kj} / n .

  Proof We show that $V_n^{-1} V_n = I_n$

$$[V_n^{-1} V_n]_{j\,j'} = \sum_{k=0}^{n-1} (\omega_n^{-kj} / n)(\omega_n^{kj'}) = \sum_{k=0}^{n-1} \omega_n^{k(j'-j)} / n = \begin{cases} 1 & \text{if } j' = j \\ 0 & \text{if } j' \neq j \end{cases}$$

since $-(n-1) < j' - j < n-1$, apparently $j' - j \neq mn$, if $m \neq 0$

30.2.4 Interpolation at the complex roots of unity

$$y = A(x) = \sum_{j=0}^{n-1} a_j x^j$$

DFT

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj} \ , \ (k = 0,1,\cdots,n-1) \qquad (30.8)$$

inverse DFT

$$a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k \omega_n^{-kj} \ , \qquad (j = 0,1,\cdots,n-1) \qquad (30.11)$$

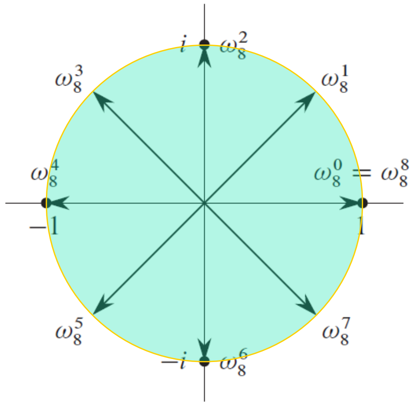
DFT

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \cdots\cdots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix} = V_n a$$

inverse DFT

$$a = \mathrm{DFT}_n^{-1}(y) = V_n^{-1} y$$

The inverse DFT （逆DFT） can be computed in $\Theta(n\lg n)$ time, for (30.8), by replacing ω_n by ω_n^{-1} , and divide each element of the result by n.



30.2.4 Interpolation at the complex roots of unity

$$y = A(x) = \sum_{j=0}^{n-1} a_j x^j$$

DFT

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj} \;, \; (k = 0, 1, \cdots, n-1) \qquad (30.8)$$

inverse DFT

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-kj} \;, \qquad (j = 0, 1, \cdots, n-1) \qquad (30.11)$$

DFT

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \cdots\cdots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{pmatrix} = V_n a$$

inverse DFT

$$a = \mathrm{DFT}_n^{-1}(y) = V_n^{-1} y$$

coefficient form
$a = (a_0, a_1, \cdots, a_{n-1})^T$

(FFT, $\Theta(n\lg n)$)
evaluation →

← interpolation
(inverse FFT)
$\Theta(n\lg n)$

point-value form
$\{(x_j, y_j)\}$

30.2.4 Interpolation at the complex roots of unity

$a_0, a_1, \cdots, a_{n-1}$
$b_0, b_1, \cdots, b_{n-1}$

Ordinary multiplication
Time $\Theta(n^2)$

$c_0, c_1, \cdots\cdots, c_{2n-2}, 0$

1 Extend point-value,
Time $\Theta(n)$.

$a_0, a_1, \cdots, a_{n-1}, 0, \cdots, 0$
$b_0, b_1, \cdots, b_{n-1}, 0, \cdots, 0$

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

$$C = A \cdot B?$$

4 Interpolation,
Inverse DFT,
Time $\Theta(n\lg n)$

2 Evaluation, DFT,
Time $\Theta(n\lg n)$

$A(\omega_{2n}^0), \quad B(\omega_{2n}^0)$
$A(\omega_{2n}^1), \quad B(\omega_{2n}^1)$
$\vdots$
$A(\omega_{2n}^{2n-1}), \quad B(\omega_{2n}^{2n-1})$

3 Pointwise multiplication
Time $\Theta(n)$

$C(\omega_{2n}^0)$
$C(\omega_{2n}^1)$
$\vdots$
$C(\omega_{2n}^{2n-1})$

iDFT

Convolution
卷积

$a \otimes b$
$= (a_0, a_1, \cdots, a_{n-1}) \otimes (b_0, b_1, \cdots, b_{n-1})$
$= (c_0, c_1, \cdots\cdots, c_{2n-2}, 0)$
where, $c_j = a_0 b_j + a_1 b_{j-1} + \ldots + a_k b_{j-k} + \ldots + a_j b_0$

Theorem 30.8 (Convolution theorem, 卷积定理) For any two vectors a and b of length n, where n is a power of 2,

$$a \otimes b = \mathrm{DFT}_{2n}^{-1}(\mathrm{DFT}_{2n}(a) \cdot \mathrm{DFT}_{2n}(b))$$

where the vectors a and b are padded with 0's to length $2n$ and $\cdot$ denotes the component-wise product of two $2n$-element vectors.

30.3 Efficient FFT implementations

- The practical applications of the DFT, such as signal processing, demand the utmost speed.

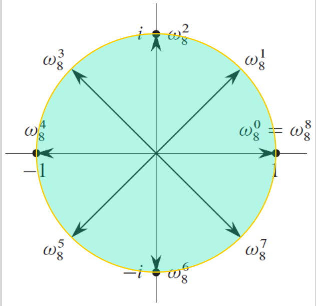
- Two efficient FFT implementations
  - iterative FFT algorithm
  - butterfly operation algorithm (parallel FFT circuit)

30.3.1 An iterative FFT implementation

$$A(x)=\sum_{j=0}^{n-1} a_j x^j = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1}$$

最美丽的公式:

$$e^{i\pi}+1=0$$



$$x=\omega_n^0,\omega_n^1,\omega_n^2,\cdots,\omega_n^{n-1};$$

$$\omega_n = e^{2\pi i/n} = \cos(2\pi/n)+i\sin(2\pi/n)$$

$$y_k = A(\omega_n^k)=\sum_{j=0}^{n-1} a_j \omega_n^{kj} \;,$$

$$\begin{pmatrix} y_0 \\ y_1 \\ \cdots \\ y_{n-1}\end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ & & \cdots\cdots & & \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1}\end{pmatrix}$$

even, $A^{[0]}(x)=a_0+a_2x+a_4x^2+\cdots+a_{n-2}x^{n/2-1};$

odd, $A^{[1]}(x)=a_1+a_3x+a_5x^2+\cdots+a_{n-1}x^{n/2-1}.$

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

$$\omega_n^k \qquad (\omega_n^k)^2 = \omega_{n/2}^k$$

let $k=0,\ \ldots,\ n/2-1,$ then

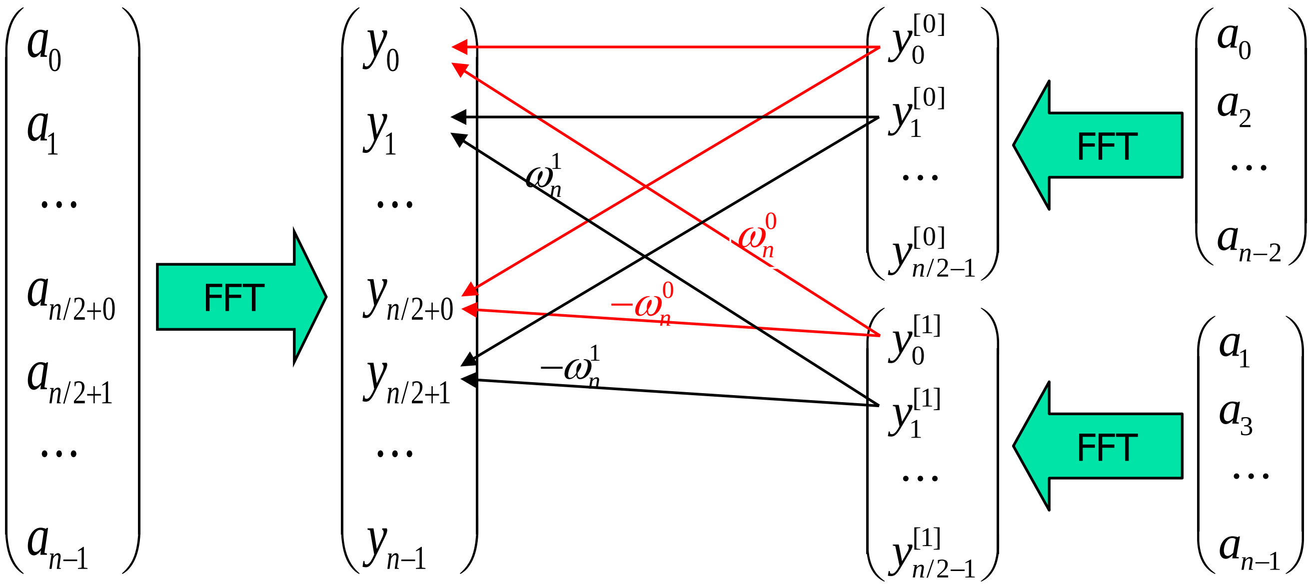
$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k)+(\omega_n^k)\cdot A^{[1]}(\omega_{n/2}^k),$$

$$A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^k)-(\omega_n^k)\cdot A^{[1]}(\omega_{n/2}^k)$$

$$y_k = y_k^0 + \omega_n^k y_k^1,$$

$$y_{k+n/2} = y_k^0 - \omega_n^k y_k^1$$

30.3.1 An iterative FFT implementation



RECURSIVE-FFT(a)

1 $n = a.length$
2 **if** $n == 1$
3   **return** a
4 $\omega_n = e^{2\pi i/n}$
5 $\omega = 1$
6 $a^{[0]} = (a_0, a_2, \ldots, a_{n-2})$
7 $a^{[1]} = (a_1, a_3, \ldots, a_{n-1})$
8 $y^{[0]} =$ RECURSIVE-FFT$(a^{[0]})$
9 $y^{[1]} =$ RECURSIVE-FFT$(a^{[1]})$
10 **for** $k = 0$ **to** $n/2 - 1$
11     $y_k = y_k^{[0]} + \omega\, y_k^{[1]}$
12     $y_{k+(n/2)} = y_k^{[0]} - \omega\, y_k^{[1]}$
13     $\omega = \omega\, \omega_n$
14 **return** y

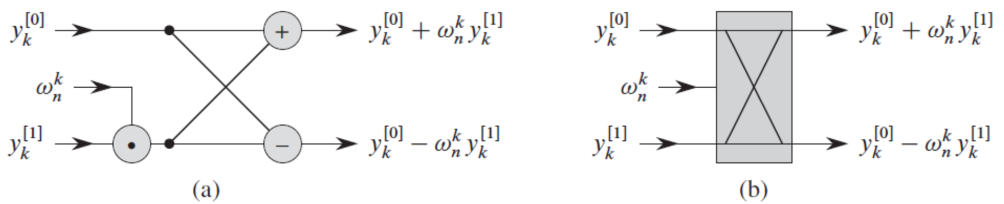
lines 11-13, RECURSIVE-FFT involves computing the value $\omega_n^k y_k^{[1]}$ twice, change the loop to compute it only once.

for $k = 0$ **to** $n/2 - 1$
    $t = \omega\, y_k^{[1]}$
    $y_k = y_k^{[0]} + t$
    $y_{k+(n/2)} = y_k^{[0]} - t$
    $\omega = \omega\, \omega_n$

butterfly operation（蝶型操作）
adding and subtracting t from $y_k^{[0]}$

30.3.1 An iterative FFT implementation

$(k = 0, 1, \cdots, n/2 - 1)$



y_k

$y_{k+n/2}$

RECURSIVE-FFT(a)

```
n = a.length
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
```

butterfly operation（蝶型操作）
adding and subtracting t from $y_k^{[0]}$

```
for k = 0 to n/2 - 1
    t = ω y_k^[1]
    y_k = y_k^[0] + t
    y_{k+(n/2)} = y_k^[0] - t
    ω = ω ω_n
```

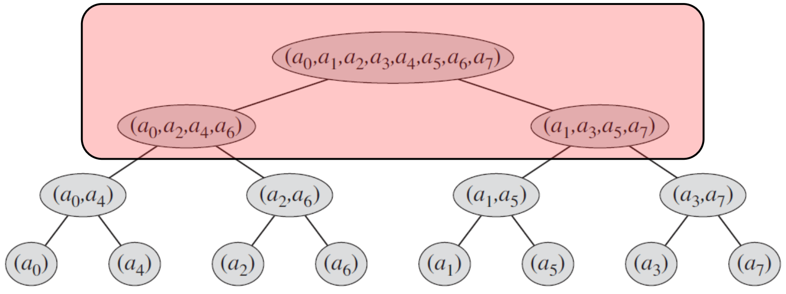
30.3.1 An iterative FFT implementation

Make the FFT algorithm iterative rather than recursive in structure.

RECURSIVE-FFT(a)

$n = a.length$
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

Recursive calls of RECURSIVE-FFT in a tree structure.



for $k = 0$ **to** $n/2 - 1$
  $t = \omega\, y_k^{[1]}$
  $y_k = y_k^{[0]} + t$
  $y_{k+(n/2)} = y_k^{[0]} - t$
  $\omega = \omega\, \omega_n$

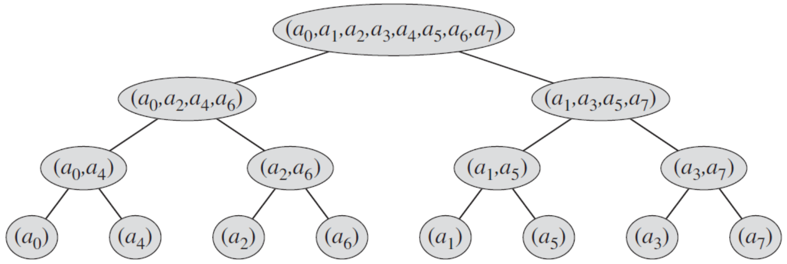
30.3.1 An iterative FFT implementation

iterative FFT algorithm

arrange the elements of *a* into the order in which they appear in the leave. 把 *a* 按递归树的顺序重新排列

1. compute the DFT of each pair using one butterfly operation, replace the pair with its DFT, the vector then holds (*n*/2)th 2-element DFT's;
   用蝶形操作，计算每一对数的DFT，得到*n*/2组两个元素的DFT
2. take the (*n*/2)th DFT's in pairs, compute the DFT of the four vector elements, 2 butterfly operations, the new vector holds (*n*/4)th 4-element DFT's;
   把这*n*/2组两个元素的DFT，再两两成对，每一对使用两次蝶形操作计算出4个元素的DFT（共有*n*/4组）
3. continue in this manner.

Recursive calls of RECURSIVE-FFT in a tree structure.



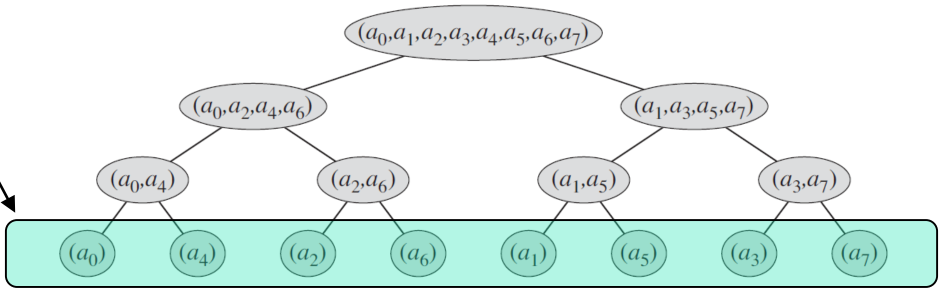
$$
\begin{aligned}
&\textbf{for } k = 0 \textbf{ to } n/2 - 1\\
&\quad t = \omega\, y_k^{[1]}\\
&\quad y_k = y_k^{[0]} + t\\
&\quad y_{k+(n/2)} = y_k^{[0]} - t\\
&\quad \omega = \omega\, \omega_n
\end{aligned}
$$

- 第一层循环，分层：高度（从下往上）（如果是递归的话，相当于递归的深度）
- 第二层循环，分组：求 m 个数的FFT(即每组有 m 个数)，共有 n/m 组数（即每组有 m 个数，共有 n 个数，跟源数据个数相等）
- 第三层循环，蝶算：蝶形操作（每组有 m 个数，有 $m/2$ 个蝶形操作，一共 n/m组数，所有蝶形操作 $(m/2)*(n/m) = n/2$ 个 ）

initial $A[0..\ n\text{-}1]$

```
ITERATIVE-FFT (a)
BIT-REVERSE-COPY (a, A)
n←length[a]  // n is 2^k
for s←1 to lg n
    m←2^s
    ω_m←e^{2πi/m}
    for k←0 to n-1 by m //步长为m, 共n/m组
        ω←1
        for j←0 to m/2 -1  // 每组m个数
            t←ωA[k+j+m/2]
            u←A[k+j]
            A[k+j]←u+t
            A[k+j+m/2]←u-t
            ω←ω · ω_m
```

Recursive calls of RECURSIVE-FFT in a tree structure.

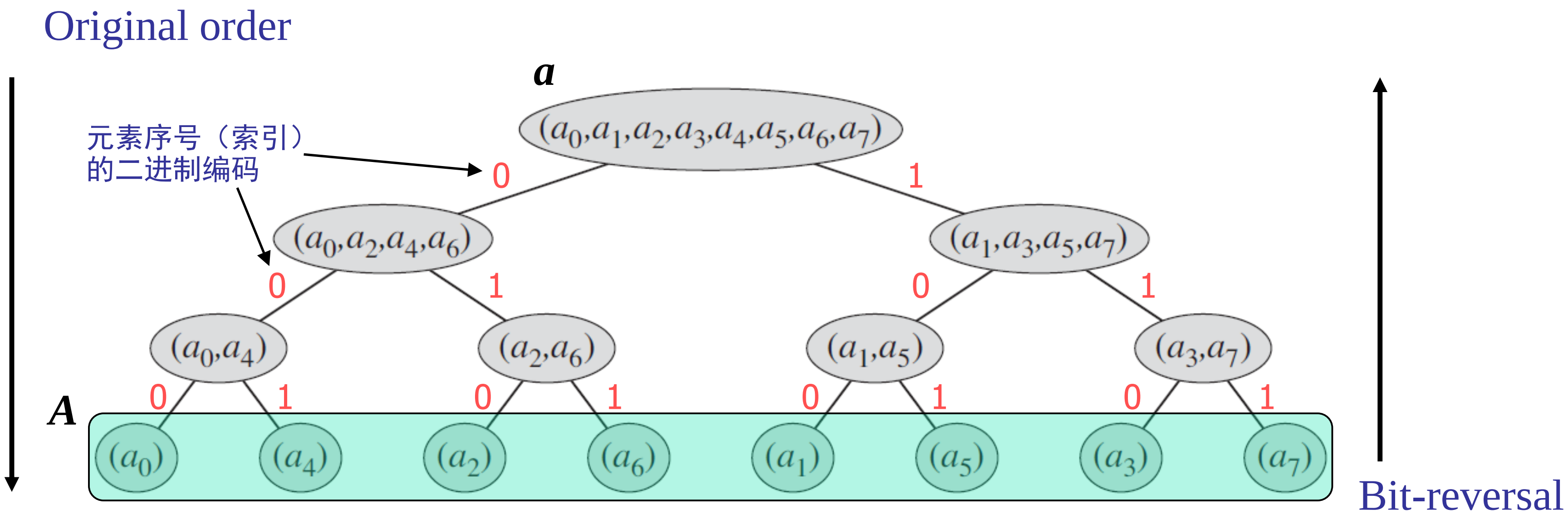


$s=1$	$s=2$	$s=3$
$m=2^s=2$	$m=4$	$m=8$
$k=0, 2,..., n\text{-}1$	$k=0, 4,..., n\text{-}1$	$k=0, 8,..., n\text{-}1$
$j=0$	$j=0, 1$	$j=0, 1, 2, 3$
$A[0]$ $A[1]$	$A[0]$ $A[2]$	$A[0]$ $A[4]$
$A[2]$ $A[3]$	$A[1]$ $A[3]$	$A[1]$ $A[5]$
…	…	…

30.3.1 An iterative FFT implementation

- 递归到最底层时，按原始输入的什么顺序开始计算?
- 元素序号的按位逆置换 (Bit-reversal permutation)

ITERATIVE-FFT (*a*)
1 BIT-REVERSE-COPY (*a*, *A*)
…



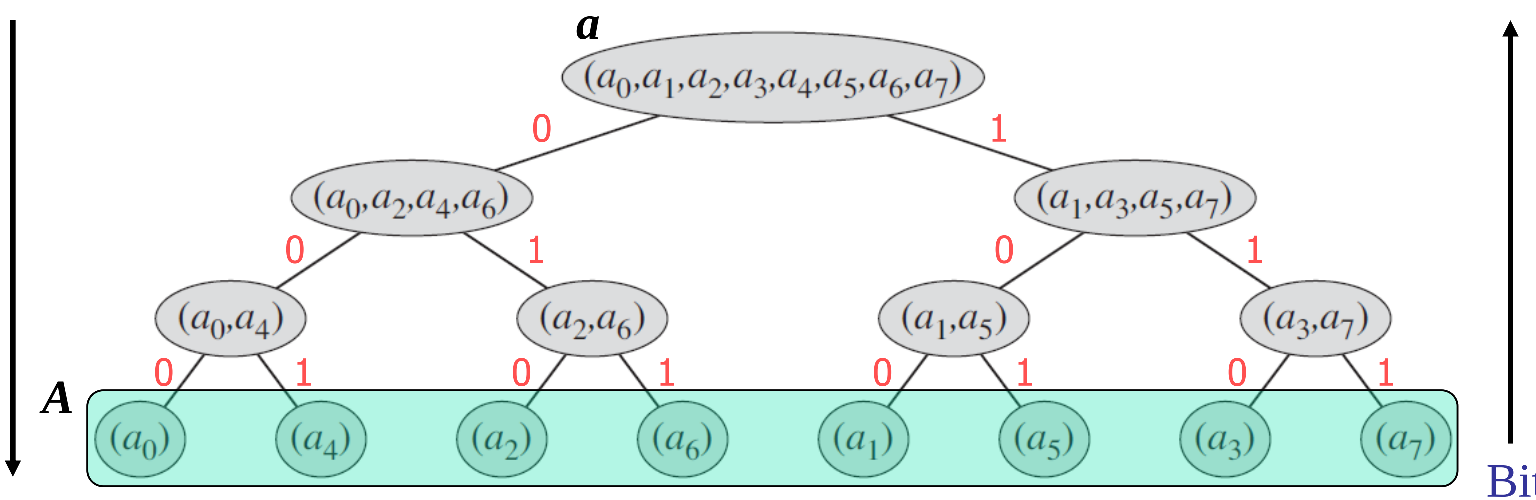
0	1	2	3	4	5	6	7
000	001	010	011	100	101	110	111
000	100	010	110	001	101	011	111
0	4	2	6	1	5	3	7

30.3.1 An iterative FFT implementation

Bit-reversal permutation

ITERATIVE-FFT (*a*)
1 BIT-REVERSE-COPY (*a*, *A*)
…

Original order



Bit-reversal

0	1	2	3	4	5	6	7
000	001	010	011	100	101	110	111
000	100	010	110	001	101	011	111
0	4	2	6	1	5	3	7

BIT-REVERSE-COPY(*a*, *A*) // $\Theta(n\lg n)$
1 $n \leftarrow length[a]$
2 **for** $k \leftarrow 0$ to $n-1$ // $\Theta(n)$
3 **do** $A[k] \leftarrow a_{\mathrm{rev}(k)}$ // $\Theta(\lg n)$
// rev(k)表示对 k 的二进制进行按位逆置换

30.3.1 An iterative FFT implementation

iterative FFT algorithm，FFT的非递归算法

```
ITERATIVE-FFT (a)
  BIT-REVERSE-COPY (a, A)
  n←length[a]  // n is a power of 2.
  for s←1 to lg n
      m←2^s
      ω_m←e^{2πi/m}
      for k←0 to n-1 by m
          ω←1
          for j←0 to m/2 -1
              t←ωA[k+j+m/2]
              u←A[k+j]
              A[k+j]←u+t
              A[k+j+m/2]←u-t
              ω←ω · ω_m
```

running time?

```
BIT-REVERSE-COPY(a, A)  // Θ(nlgn)
  n←length[a]
  for k←0 to n–1            // Θ(n)
      do A[k]←a_rev(k)      // Θ(lgn)
      // rev(k)表示对 k 的二进制进行按位逆置换
```

```
    …
     for(k=0; k<n; k++)
    {   // a是输入的n个数，A是a的按位逆置换
        A[k] = a[rev(k)];
    }
    …
unsigned rev_x(unsigned x) // rev(k)的实现
{
    unsigned i;
    for(i=0; i<=n_len; i++)
        rev[i] = 1&(x>>i);
    unsigned y=0;
    for(i=0; i<=n_len; i++)
        y |= rev[i]<<(n_len-i);
    return y;
}
```

作业：请完整实现 ITERATIVE-FFT

30.3.1 An iterative FFT implementation

iterative FFT algorithm

```
ITERATIVE-FFT (a)
  BIT-REVERSE-COPY (a, A)          //Θ(nlgn)
  n←length[a]  // n is a power of 2.
  for s←1 to lg n                   // lgn
      m←2^s
      ω_m←e^(2πi/m)
      for k←0 to n-1 by m           // n/m=n/2^s
          ω←1
          for j←0 to m/2 -1         // m/2=2^s/2=2^(s-1)
              t←ωA[k+j+m/2]
              u←A[k+j]
              A[k+j]←u+t
              A[k+j+m/2]←u-t
              ω←ω · ω_m
```

$$L(n)=\sum_{s=1}^{\lg n}\frac{n}{2^s}2^{s-1}$$

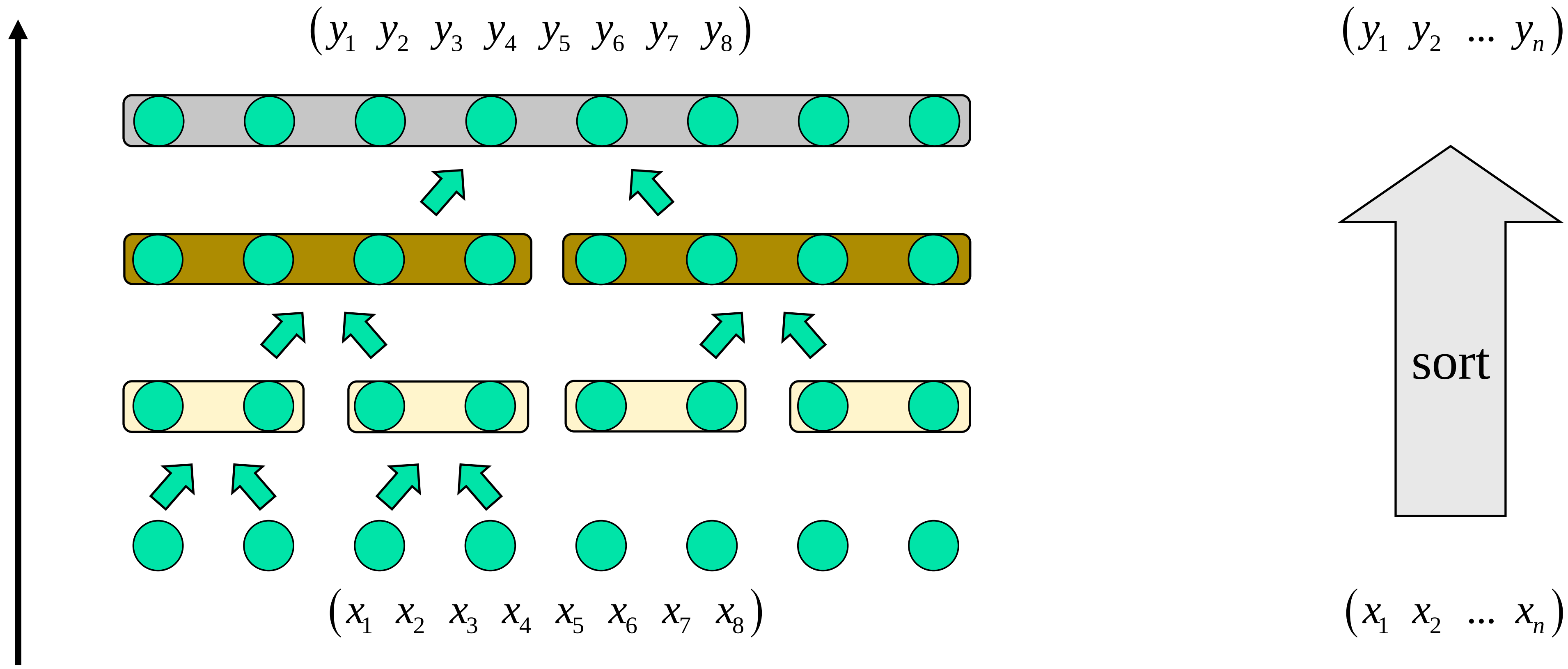
$$=\sum_{s=1}^{\lg n}\frac{n}{2}$$

$$=\Theta(n\lg n)$$

思考题(email to me)：

本算法是否能用于归并排序的非递归实现？如果可以，请写出C程序（输入n个数，迭代的归并排序，输出n个排序的数），并与递归版的归并排序比较（运行实际输入，对比运行时间）。

Iterative merge-sort



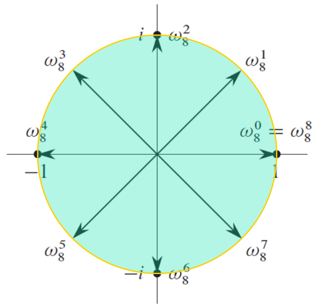
* 30.3.2 A parallel FFT circuit

FFT的并联图

- See book

30.3.3 A butterfly operation

DFT, $$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \cdots & \omega_n^{(n-1)\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}, \quad \Theta(n^2)$$



FFT, $\Theta(n \lg n)$?

FFT, $$\begin{pmatrix} y_0 \\ y_4 \\ y_2 \\ y_6 \\ y_1 \\ y_5 \\ y_3 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{4\cdot 1} & \omega_n^{4\cdot 2} & \cdots & \omega_n^{4\cdot(n-1)} \\ 1 & \omega_n^{2\cdot 1} & \omega_n^{2\cdot 2} & \cdots & \omega_n^{2\cdot(n-1)} \\ 1 & \omega_n^{6\cdot 1} & \omega_n^{6\cdot 2} & \cdots & \omega_n^{6\cdot(n-1)} \\ 1 & \omega_n^{1\cdot 1} & \omega_n^{1\cdot 2} & \cdots & \omega_n^{1\cdot(n-1)} \\ 1 & \omega_n^{5\cdot 1} & \omega_n^{5\cdot 2} & \cdots & \omega_n^{5\cdot(n-1)} \\ 1 & \omega_n^{3\cdot 1} & \omega_n^{3\cdot 2} & \cdots & \omega_n^{3\cdot(n-1)} \\ 1 & \omega_n^{7\cdot 1} & \omega_n^{7\cdot 2} & \cdots & \omega_n^{7\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = W_1 W_2 W_3 a, \quad \Theta(n \lg n)$$

稀疏矩阵?

DFT公式进行行变换后

30.3.3 A butterfly operation

$$\omega = e^{2\pi i/8}$$

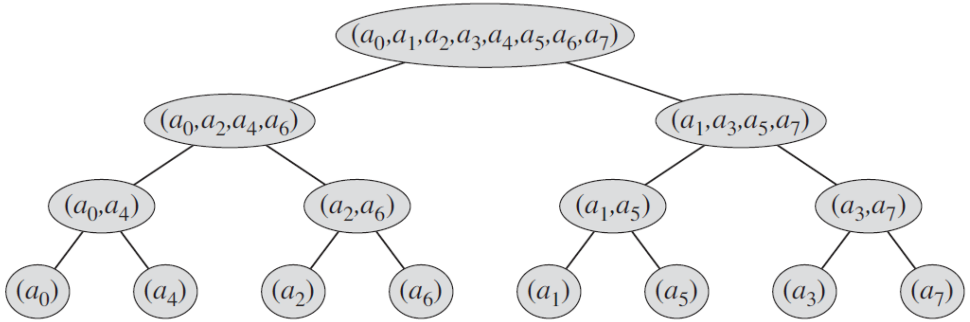
$$\begin{pmatrix} y_0 \\ y_4 \\ y_2 \\ y_6 \\ y_1 \\ y_5 \\ y_3 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{4\cdot 1} & \omega_n^{4\cdot 2} & \cdots & \omega_n^{4\cdot(n-1)} \\ 1 & \omega_n^{2\cdot 1} & \omega_n^{2\cdot 2} & \cdots & \omega_n^{2\cdot(n-1)} \\ 1 & \omega_n^{6\cdot 1} & \omega_n^{6\cdot 2} & \cdots & \omega_n^{6\cdot(n-1)} \\ 1 & \omega_n^{1\cdot 1} & \omega_n^{1\cdot 2} & \cdots & \omega_n^{1\cdot(n-1)} \\ 1 & \omega_n^{5\cdot 1} & \omega_n^{5\cdot 2} & \cdots & \omega_n^{5\cdot(n-1)} \\ 1 & \omega_n^{3\cdot 1} & \omega_n^{3\cdot 2} & \cdots & \omega_n^{3\cdot(n-1)} \\ 1 & \omega_n^{7\cdot 1} & \omega_n^{7\cdot 2} & \cdots & \omega_n^{7\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}$$

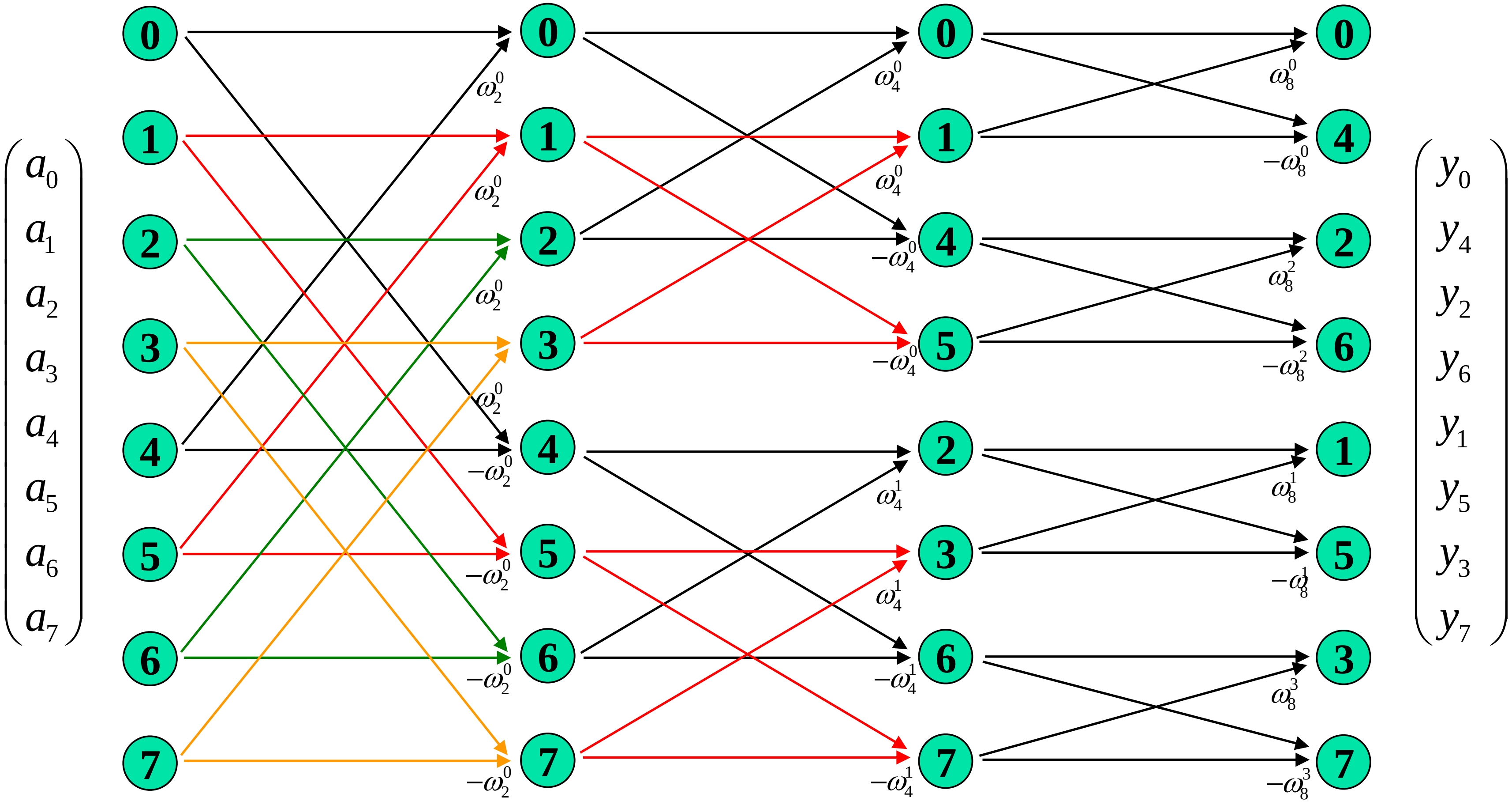
$$= \begin{pmatrix} 1 & \omega^0 & & & & & & \\ 1 & \omega^4 & & & & & & \\ & & 1 & \omega^2 & & & & \\ & & 1 & \omega^6 & & & & \\ & & & & 1 & \omega^1 & & \\ & & & & 1 & \omega^5 & & \\ & & & & & & 1 & \omega^3 \\ & & & & & & 1 & \omega^7 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & \omega^0 & 0 & & & & \\ 0 & 1 & 0 & \omega^0 & & & & \\ 1 & 0 & \omega^4 & 0 & & & & \\ 0 & 1 & 0 & \omega^4 & & & & \\ & & & & 1 & 0 & \omega^2 & 0 \\ & & & & 0 & 1 & 0 & \omega^2 \\ & & & & 1 & 0 & \omega^6 & 0 \\ & & & & 0 & 1 & 0 & \omega^6 \end{pmatrix} \cdot \begin{pmatrix} 1 & & & & \omega^0 & & & \\ & 1 & & & & \omega^0 & & \\ & & 1 & & & & \omega^0 & \\ & & & 1 & & & & \omega^0 \\ 1 & & & & \omega^4 & & & \\ & 1 & & & & \omega^4 & & \\ & & 1 & & & & \omega^4 & \\ & & & 1 & & & & \omega^4 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}$$

A butterfly operation

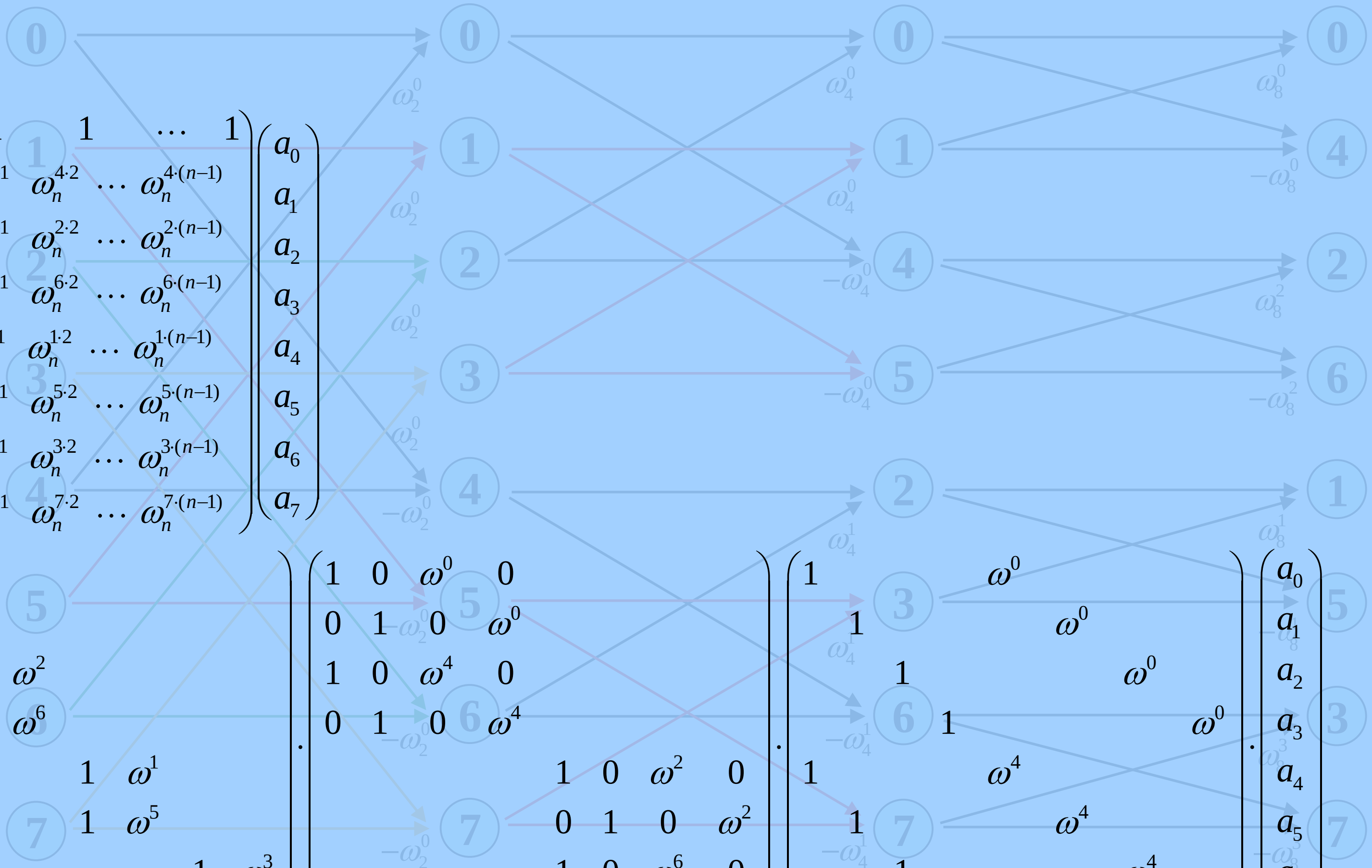
for $k = 0$ **to** $n/2 - 1$

$$t = \omega\, y_k^{[1]}$$
$$y_k = y_k^{[0]} + t$$
$$y_{k+(n/2)} = y_k^{[0]} - t$$
$$\omega = \omega\, \omega_n$$





30.3.3 A butterfly operation



$$\omega = e^{2\pi i/8}$$

$$\begin{pmatrix} y_0 \\ y_4 \\ y_2 \\ y_6 \\ y_1 \\ y_5 \\ y_3 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{4\cdot 1} & \omega_n^{4\cdot 2} & \cdots & \omega_n^{4\cdot(n-1)} \\ 1 & \omega_n^{2\cdot 1} & \omega_n^{2\cdot 2} & \cdots & \omega_n^{2\cdot(n-1)} \\ 1 & \omega_n^{6\cdot 1} & \omega_n^{6\cdot 2} & \cdots & \omega_n^{6\cdot(n-1)} \\ 1 & \omega_n^{1\cdot 1} & \omega_n^{1\cdot 2} & \cdots & \omega_n^{1\cdot(n-1)} \\ 1 & \omega_n^{5\cdot 1} & \omega_n^{5\cdot 2} & \cdots & \omega_n^{5\cdot(n-1)} \\ 1 & \omega_n^{3\cdot 1} & \omega_n^{3\cdot 2} & \cdots & \omega_n^{3\cdot(n-1)} \\ 1 & \omega_n^{7\cdot 1} & \omega_n^{7\cdot 2} & \cdots & \omega_n^{7\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \omega^0 & & & & & & \\ 1 & \omega^4 & & & & & & \\ & & 1 & \omega^2 & & & & \\ & & 1 & \omega^6 & & & & \\ & & & & 1 & \omega^1 & & \\ & & & & 1 & \omega^5 & & \\ & & & & & & 1 & \omega^3 \\ & & & & & & 1 & \omega^7 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & \omega^0 & 0 & & & & \\ 0 & 1 & 0 & \omega^0 & & & & \\ 1 & 0 & \omega^4 & 0 & & & & \\ 0 & 1 & 0 & \omega^4 & & & & \\ & & & & 1 & 0 & \omega^2 & 0 \\ & & & & 0 & 1 & 0 & \omega^2 \\ & & & & 1 & 0 & \omega^6 & 0 \\ & & & & 0 & 1 & 0 & \omega^6 \end{pmatrix} \cdot \begin{pmatrix} 1 & & & & \omega^0 & & & \\ & 1 & & & & \omega^0 & & \\ & & 1 & & & & \omega^0 & \\ & & & 1 & & & & \omega^0 \\ 1 & & & & \omega^4 & & & \\ & 1 & & & & \omega^4 & & \\ & & 1 & & & & \omega^4 & \\ & & & 1 & & & & \omega^4 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}$$

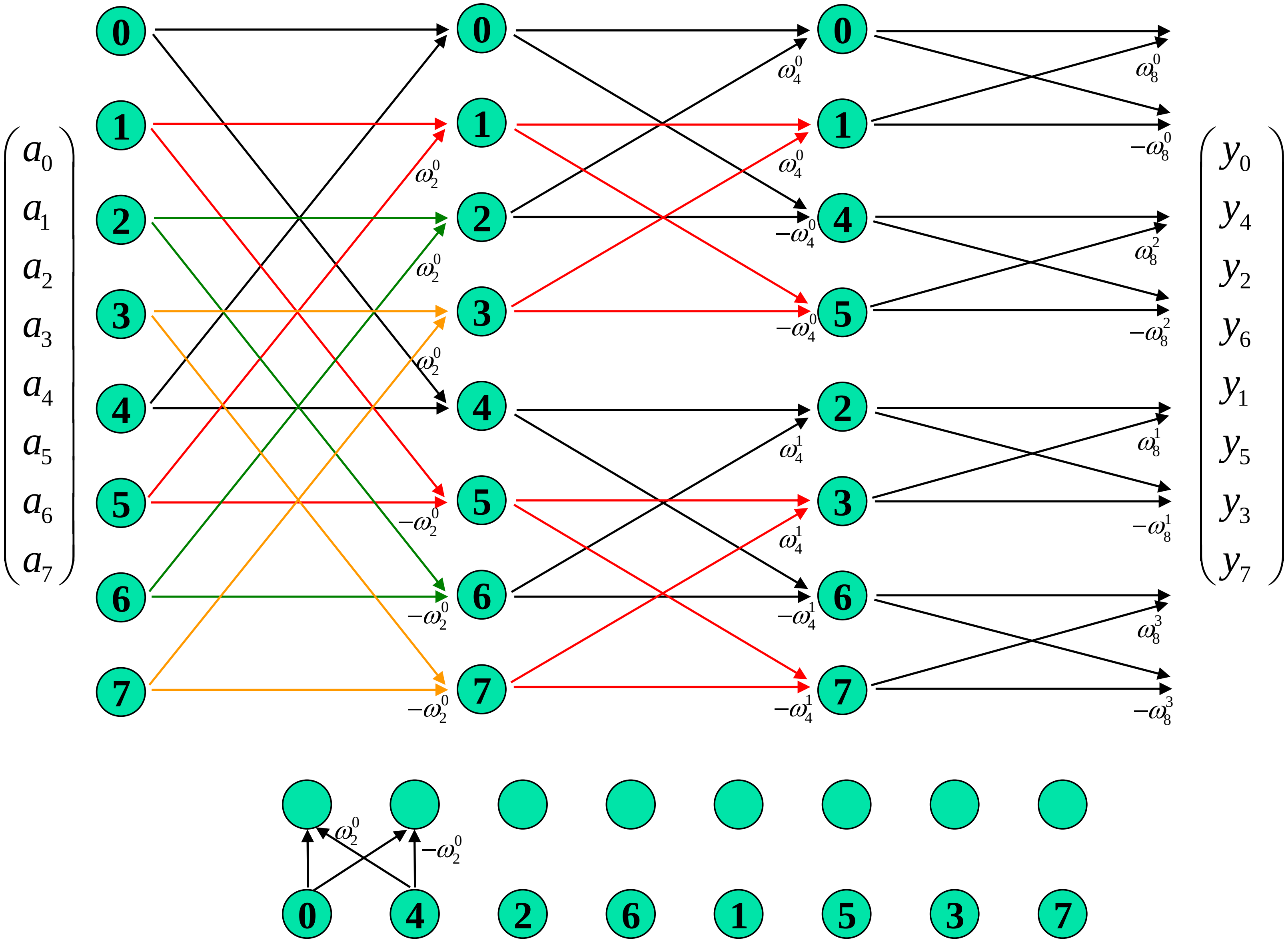
30.3.3 A butterfly operation

$$\omega = e^{2\pi i/8}$$

$$\begin{pmatrix} y_0 \\ y_4 \\ y_2 \\ y_6 \\ y_1 \\ y_5 \\ y_3 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{4\cdot 1} & \omega_n^{4\cdot 2} & \cdots & \omega_n^{4\cdot(n-1)} \\ 1 & \omega_n^{2\cdot 1} & \omega_n^{2\cdot 2} & \cdots & \omega_n^{2\cdot(n-1)} \\ 1 & \omega_n^{6\cdot 1} & \omega_n^{6\cdot 2} & \cdots & \omega_n^{6\cdot(n-1)} \\ 1 & \omega_n^{1\cdot 1} & \omega_n^{1\cdot 2} & \cdots & \omega_n^{1\cdot(n-1)} \\ 1 & \omega_n^{5\cdot 1} & \omega_n^{5\cdot 2} & \cdots & \omega_n^{5\cdot(n-1)} \\ 1 & \omega_n^{3\cdot 1} & \omega_n^{3\cdot 2} & \cdots & \omega_n^{3\cdot(n-1)} \\ 1 & \omega_n^{7\cdot 1} & \omega_n^{7\cdot 2} & \cdots & \omega_n^{7\cdot(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & \omega^0 & & & & & & \\ 1 & \omega^4 & & & & & & \\ & & 1 & \omega^2 & & & & \\ & & 1 & \omega^6 & & & & \\ & & & & 1 & \omega^1 & & \\ & & & & 1 & \omega^5 & & \\ & & & & & & 1 & \omega^3 \\ & & & & & & 1 & \omega^7 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & \omega^0 & 0 & & & & \\ 0 & 1 & 0 & \omega^0 & & & & \\ 1 & 0 & \omega^4 & 0 & & & & \\ 0 & 1 & 0 & \omega^4 & & & & \\ & & & & 1 & 0 & \omega^2 & 0 \\ & & & & 0 & 1 & 0 & \omega^2 \\ & & & & 1 & 0 & \omega^6 & 0 \\ & & & & 0 & 1 & 0 & \omega^6 \end{pmatrix} \cdot \begin{pmatrix} 1 & & & & \omega^0 & & & \\ & 1 & & & & \omega^0 & & \\ & & 1 & & & & \omega^0 & \\ & & & 1 & & & & \omega^0 \\ 1 & & & & \omega^4 & & & \\ & 1 & & & & \omega^4 & & \\ & & 1 & & & & \omega^4 & \\ & & & 1 & & & & \omega^4 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}$$

30.3.4 A new butterfly operation?



30.3.4 A new butterfly operation?

(1)

$$A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-1} x^{n-1}$$

even, $A^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \cdots + a_{n-2} x^{n/2-1}$

odd, $A^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \cdots + a_{n-1} x^{n/2-1}$

(2)

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

(3)

$\omega_n^k \qquad (\omega_n^k)^2 = \omega_{n/2}^k$

$$A(\omega_n^k) = A^{[0]}(\omega_{n/2}^k) + (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$

$$A(\omega_n^{n/2+k}) = A^{[0]}(\omega_{n/2}^k) - (\omega_n^k) \cdot A^{[1]}(\omega_{n/2}^k),$$

$k = 0, 1, \ldots, n/2-1$

(4)

$$\mathrm{DFT}(a_0, a_1, \cdots, a_{n-1}) = A(\omega_n^k),\ k = 0, 1, \cdots, n-1$$

$$\mathrm{DFT}(a_0, a_2, \cdots, a_{n-2}) = A^{[0]}(\omega_{n/2}^k),\ k = 0, 1, \cdots, n/2-1$$

$$\mathrm{DFT}(a_1, a_3, \cdots, a_{n-1}) = A^{[1]}(\omega_{n/2}^k),\ k = 0, 1, \cdots, n/2-1$$

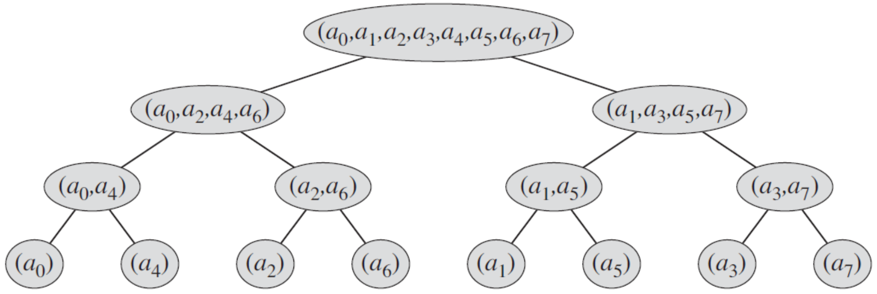
for $k = 0$ **to** $n/2 - 1$

$\quad t = \omega\, y_k^{[1]}$

$\quad y_k = y_k^{[0]} + t$

$\quad y_{k+(n/2)} = y_k^{[0]} - t$

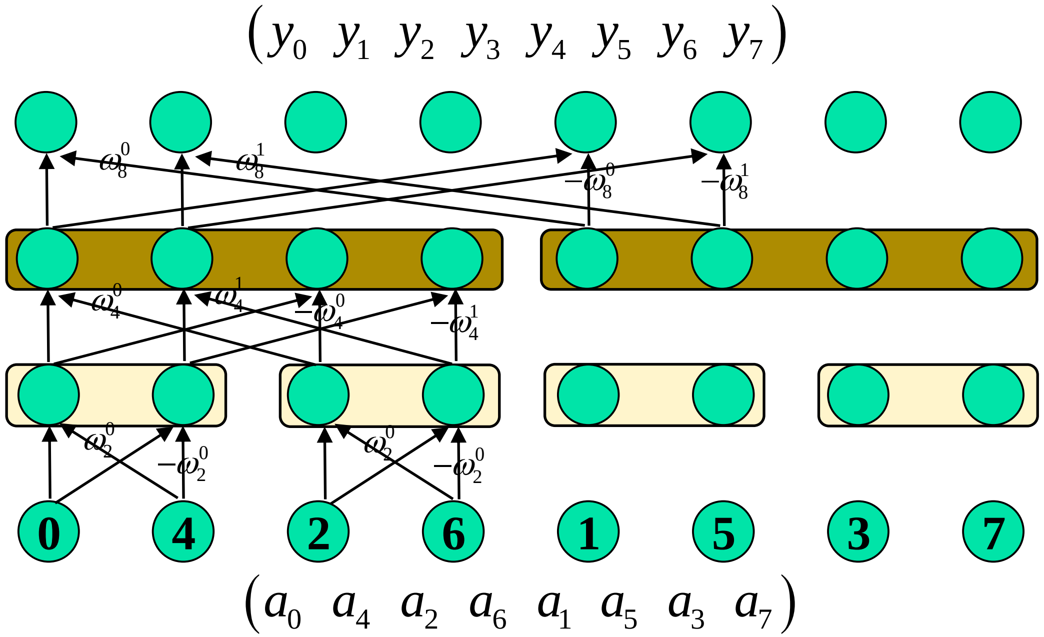
$\quad \omega = \omega\, \omega_n$

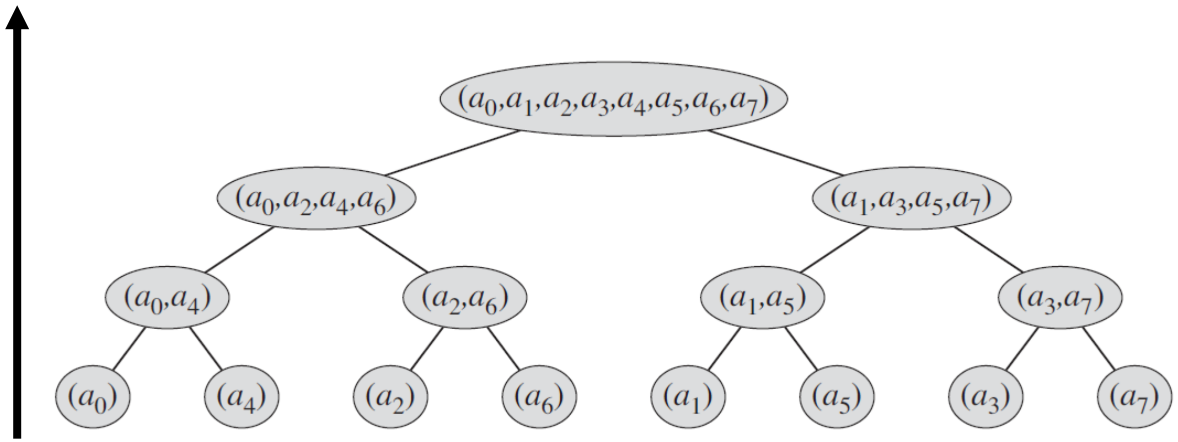


30.3.4 A new butterfly operation?

for $k = 0$ **to** $n/2 - 1$

$$t = \omega\, y_k^{[1]}$$
$$y_k = y_k^{[0]} + t$$
$$y_{k+(n/2)} = y_k^{[0]} - t$$
$$\omega = \omega\, \omega_n$$





Some Applications of FFT

- Signal processing (phonic, image, video, $\cdots$)
- Polynomials operation $\quad C(x) = A(x)*B(x) = \sum_{j=0}^{2n-2} c_j x^j$
- Multiplication of two big integers

$$y = A(x) = \sum_{j=0}^{n-1} a_j x^j \qquad 4567 = A(x=10) = \sum_{j=0}^{n-1} a_j x^j = 4*10^3 + 5*10^2 + 6*10^1 + 7*10^0$$

- $\cdots$

two n-digit numbers X and Y, Complexity($X \times Y$) = ?

Divide and Conquer (Karatsuba's algorithm)

Let $\quad X = ab$, $\qquad Y = cd$

then $\quad XY = (10^m a+b)(10^m c+d) = 10^{2m}ac+10^m(bc+ad)+bd$

Note that $bc + ad = ac + bd - (a - b)(c - d)$. So, we have

Complexity analysis:

$T(1) = 1$,

$T(n) = 3T(\lceil n/2 \rceil)+O(n)$.

Applying Master Theorem,

$$T(n) = O(n^{\log_2 3}) = O(n^{1.585})$$