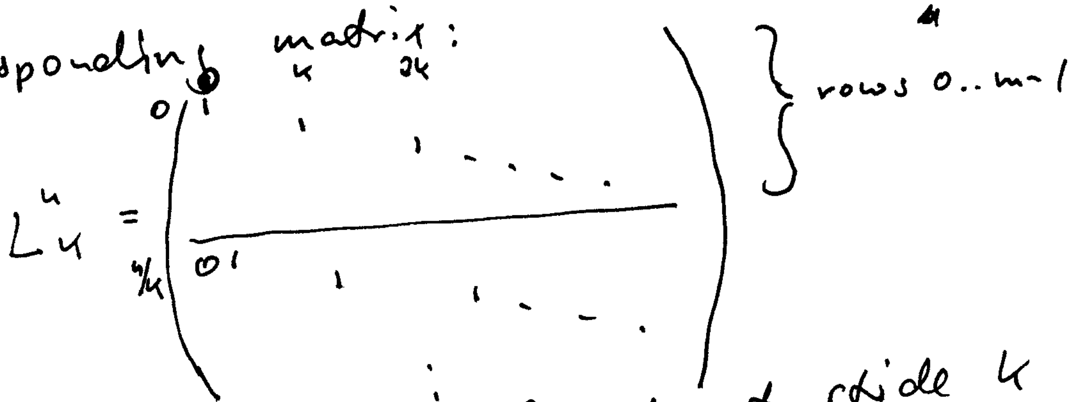
Stride Permutations

$L^n_k: i \mapsto ki \bmod n-1, \quad i = 0 .. n-2$

$\quad n-1 \mapsto n-1$

or $L^n_k: im+j \mapsto jk+i, \quad (i,j) = ((0,0),(0,1),\dots)$ $\quad i = 0..k-1$, $j = 0..m-1$

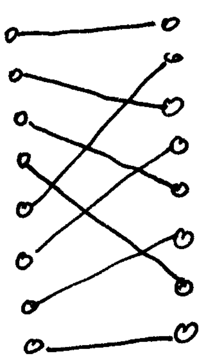
Corresponding matrix:



Interpretation: read elements at stride k
or write elements at stride $m = n/k$

Example:

L^8_4: in

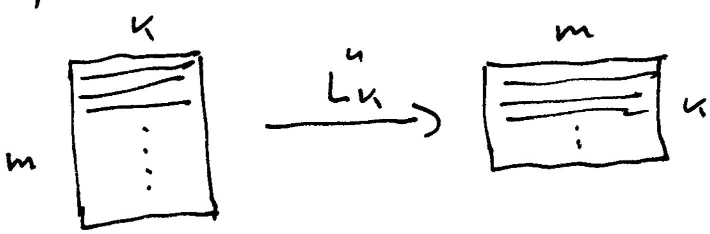


out

read at stride 4
or write at stride 2

in general: $L^n_{n/2}$ is called "perfect shuffle"

Interpretation as matrix transposition:



row-major order

also called cornerturn

L^n_k: read at stride k

$n = km: \quad (L^n_k)^{-1} = (L^n_k)^T = L^n_m$

Stride Permutations and Tensor Products

A $n \times n$, B $m \times m$

$$\Rightarrow \quad A \otimes B = L^{nm}_{n} (B \otimes A) L^{nm}_{m}$$

Example:

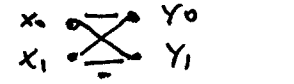
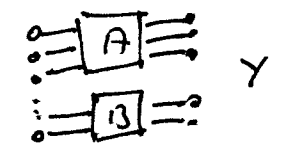
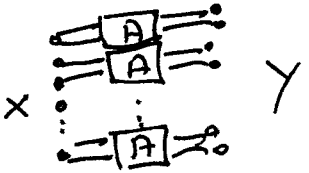
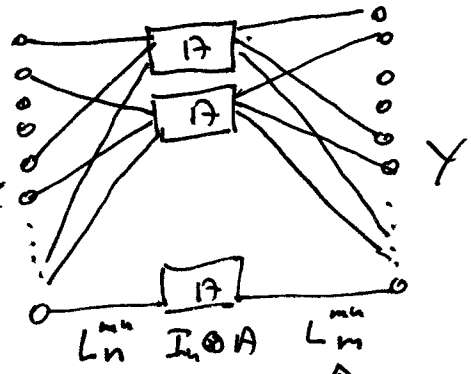
$$A \otimes I_m = L^{nm}_{n} (I_m \otimes A) L^{nm}_{m}$$

- $A \otimes I_m$: m A's at stride m
- L^{nm}_{n}: write at stride m
- $I_m \otimes A$: m A's at stride 1
- L^{nm}_{m}: read at stride m

Structured matrices are useful for describing fast algorithms

- concise
- visual
- easy to manipulate
- makes parallelism, vector structure explicit
- can be translated into code (even automatically)

Different views

structure	matrix	dataflow	pseudo code	
DFT_2	$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$		$y_0 = x_0 + x_1$ $y_1 = x_0 - x_1$	butterfly
$\mathrm{diag}(a_0, \ldots, a_{n-1})$	$\begin{pmatrix} a_0 & & \\ & a_1 & \\ & & \ddots & a_{n-1} \end{pmatrix}$		for $i = (0:1:n-1)$ $y_i = a_i x_i$	scaling
$m \times m$ ← $A \otimes I_n$ ← $n \times n$	$\begin{pmatrix} A & & \\ & B & \\ & & \ddots \end{pmatrix}$		$y(0:1:m-1) = A \cdot x(0:1:m-1)$ $y(n:1:m+n-1) = A \cdot x(m:1:m+n-1)$	parallel
$m \times m$ ← $I_n \otimes A$	$\begin{pmatrix} A & & \\ & A & \\ & & \ddots \end{pmatrix}$		for $i = 0:1:n-1$ $y(im:1:im+m-1) = A x(im:1:im+m-1)$ loopable, n A's in parallel	
$m \times m$ ← $A \otimes I_n$	$\begin{pmatrix} a_{00} I_n & a_{01} I_n & \cdots \\ a_{10} I_n & \ddots & \\ \vdots & & \end{pmatrix}$		for $i = 0:1:n-1$ $y(i:n:i+(m-1)n) = A x(i:n:i+(m-1)n)$ loopable, n A's in parallel at stride n vectorizable	

to draw dataflow use

$A \otimes I_n = L^{mn}_m (I_n \otimes A) L^{mn}_n$

read at stride n — write at stride n

L^{kn}_k $\quad kn = km$

$$\begin{pmatrix} 1 & & & \cdots \\ & 1 & & \cdots \\ & & \ddots & \\ 1 & & & \cdots \\ & 1 & & \cdots \\ & & \ddots & \end{pmatrix}$$

does the same, but upper one without mod

```
for i = 0:1:n-1
    for j = 0:1:m-1
        y[i*m+j] = x[j*n+i]
```

or

```
for i = 0:1:n-2
    y[i] = x[n*i mod n-1]
y[n-1] = x[n-1]
```

} read at stride n

cost analysis: (counting adds and mults)

$C(u) = (A(u), M(u))$

careful: for complex matrices this counts complex ops which is different from real ops

$\text{cost}(DFT_2) = (2,0)$

$\text{cost}(I_n) = (0,0)$

$\text{cost}(L^{kn}_k) = (0,0)$

$\text{cost}(A \otimes B) = \text{cost}(A) + \text{cost}(B)$

$\text{cost}(\bigoplus_{i=0}^{n-1} (q_i)) \leq (0,n)$ (or: $= (0,k)$, $k = \#\, q_i\text{'s} \neq \pm 1$)

$\text{cost}(I_n \otimes A) = n \cdot \text{cost}(A)$

$A \otimes I_n$

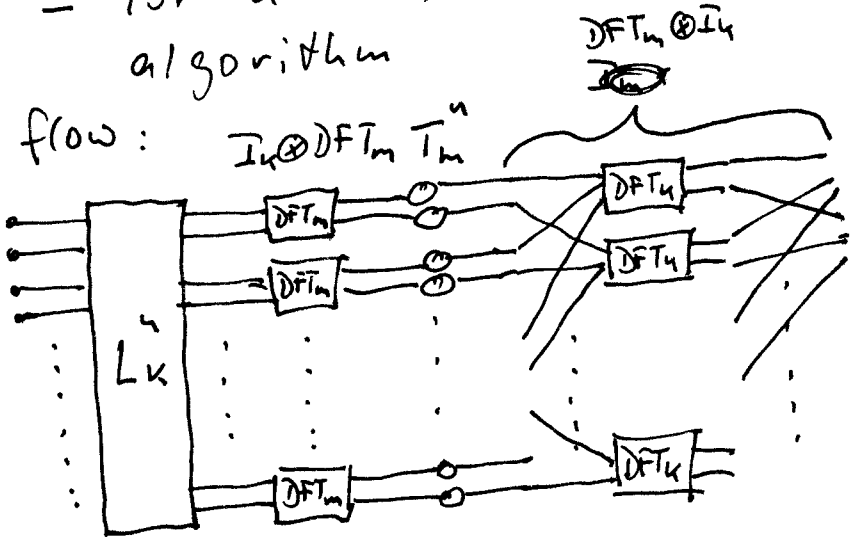
2.) Cooley-Tukey FFT ("the" FFT)

a.) $DFT_{km} = (DFT_k \otimes I_m)\, T^{mn}_m\, (I_k \otimes DFT_m)\, L^{mn}_k$

$T^{mn}_m = \bigoplus_{i=0}^{n-1} \text{diag}(1, \omega_{mn}^{i}, \ldots, \omega_{mn}^{(m-1)i})$ "twiddle factors"

Notes:
- the above version is called decimation-in-time (DIT)
- if recursively the same k is chosen it is called radix-k
- For a 2-power n, a.) yields an $O(n \log(n))$ algorithm

Data flow: $I_k \otimes DFT_m \; T^{mn}_m \; DFT_m \otimes I_k$



b.) Decimation in frequency: transpose of DIT

$$DFT_{km} = L^{km}_{m}\,(I_k \otimes DFT_m)\,T^{km}_{m}\,(DFT_k \otimes I_m)$$

Tensor: $(A \otimes B)^T = A^T \otimes B^T$

- again: radix-k means the same k is chosen recursively

c.) Iterative version; $n = 2^k$

- recursively expand radix-2 DIT
- eliminate parentheses
- collect all stride permutations L^n_k

$$DFT_{2^k} = \left[\prod_{j=1}^{k} \underbrace{(I_{2^{j-1}} \otimes DFT_2 \otimes I_{2^{k-j}})}_{\text{butterflies of varying stride } 2^{k-j}}\ \underbrace{(I_{2^{j-1}} \oplus T^{2^{k-j+1}}_{2^{k-j}})}_{\text{diagonal}}\right] \underbrace{R_{2^k}}_{\text{bit-reversal}}$$

- transposition yields another version with bit-reversal in the end

d.) Pease version

- manipulates c.) for constant geometry

$$DFT_{2^k} = \left(\prod_{j=1}^{k} \underbrace{L^{2^k}_{2}\,(I_{2^{k-1}} \otimes DFT_2)}_{\text{does not depend on } j}\ \underbrace{T^{\cdots}_{j}}_{\text{diagonal}}\right) R_{2^k}$$

- transposition yields another version