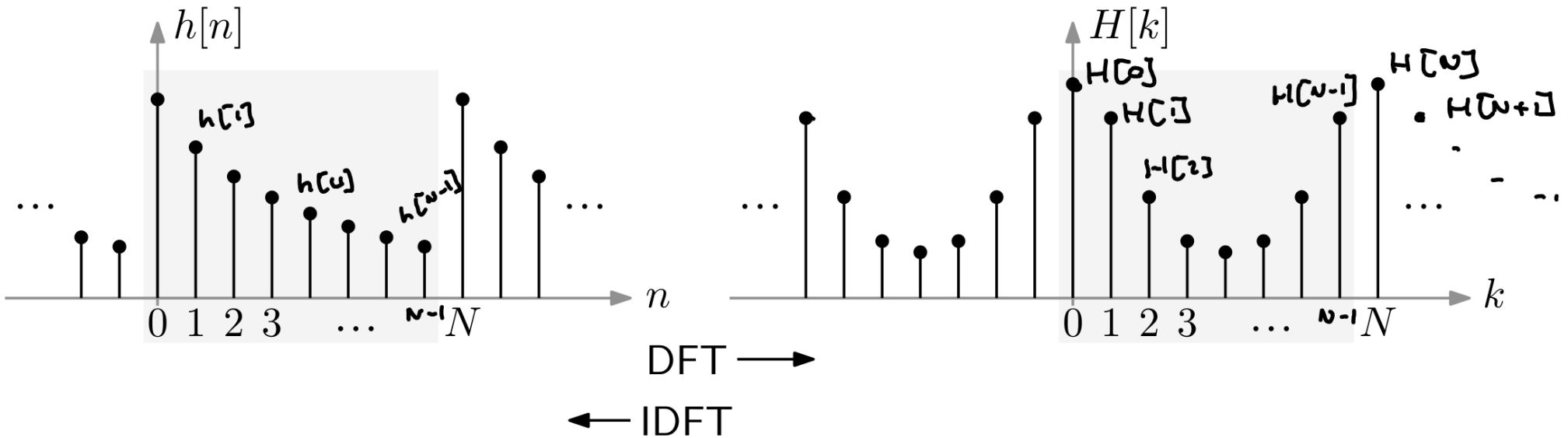
Fast Fourier transform (FFT)

And examples of how to compute things quickly

Herman Kamper


h[n]
h[1]
h[2]
h[N-1]
0 1 2 3 ... N-1 N
n
H[k]
H[0]
H[1]
H[2]
H[N-1]
H[N]
H[N+1]
0 1 2 3 ... N-1 N
k
DFT
IDFT

Fast Fourier transform (FFT)

N-point DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

$X[0] = \ldots$
$X[1] = \ldots$
$X[2] = \ldots$
$X[3] = x[0] \cdot e^{-j2\pi 3 \cdot 0/N} + x[1] \cdot e^{-j2\pi 3 \cdot 1/N} + \ldots + x[N-1] \cdot e^{-j2\pi 3 \cdot (N-1)/N}$
$X[4] = \ldots$
$\vdots$
$X[N-1] = \ldots$

Complex multiplications:

- for each k in $X[k]$: N complex mults
- for all N values of $X[k]$: N^2 complex mults

radix-2 FFT: Require N to be power of 2

The FFT and DFT lead to precisely the same result. They differ only in the way that they are calculated.

Decimation in time

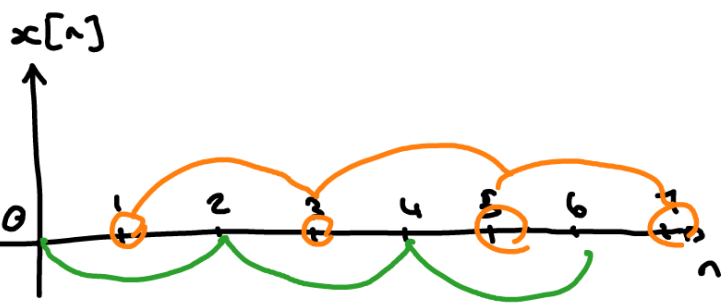
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

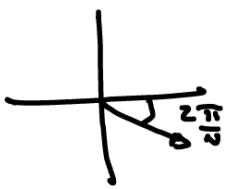
$$= \sum_{n=0}^{N/2-1} x[2n] \cdot e^{-j\frac{2\pi k(2n)}{N}} + \sum_{n=0}^{N/2-1} x[2n+1] \cdot e^{-j\frac{2\pi k(2n+1)}{N}}$$

$$= \sum_{n=0}^{N/2-1} x[2n] \cdot e^{-j\frac{2\pi k\, n}{N/2}} + e^{-j\frac{2\pi k}{N}} \cdot \sum_{n=0}^{N/2-1} x[2n+1] \cdot e^{-j\frac{2\pi k\, n}{N/2}}$$

$$= X_1[k] + e^{-j\frac{2\pi k}{N}} \cdot X_2[k]$$

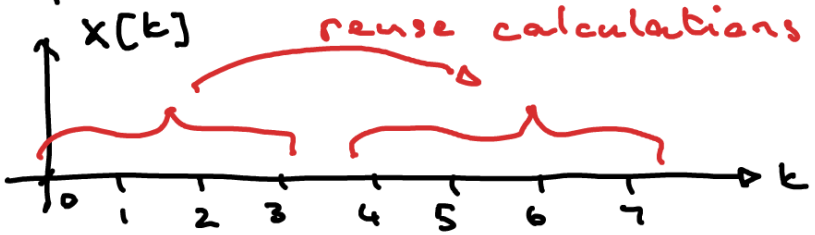
$$= X_1[k] + W_N^k \cdot X_2[k]$$

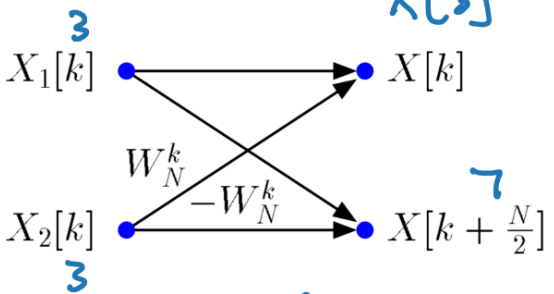




$$W_N = e^{-j\frac{2\pi}{N}}$$

Split $X[k]$ calculation in two:





$\frac{N}{2}$-point DFTs

$$X[k] = X_1[k] + W_N^k X_2[k] \quad \text{for } k = 0, 1, \ldots, \tfrac{N}{2}-1$$

$$X[k+\tfrac{N}{2}] = X_1[k+\tfrac{N}{2}] + W_N^{k+\frac{N}{2}} X_2[k+\tfrac{N}{2}] \quad \text{for } k = 0, 1, \ldots, \tfrac{N}{2}-1$$

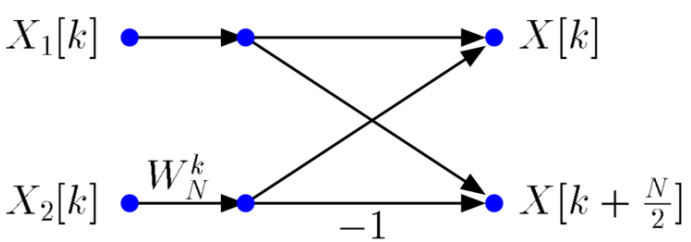
$$= X_1[k] + W_N^{k+\frac{N}{2}} X_2[k]$$

$$= X_1[k] + W_N^k \cdot W_N^{N/2} X_2[k]$$

$$= X_1[k] - W_N^k X_2[k]$$

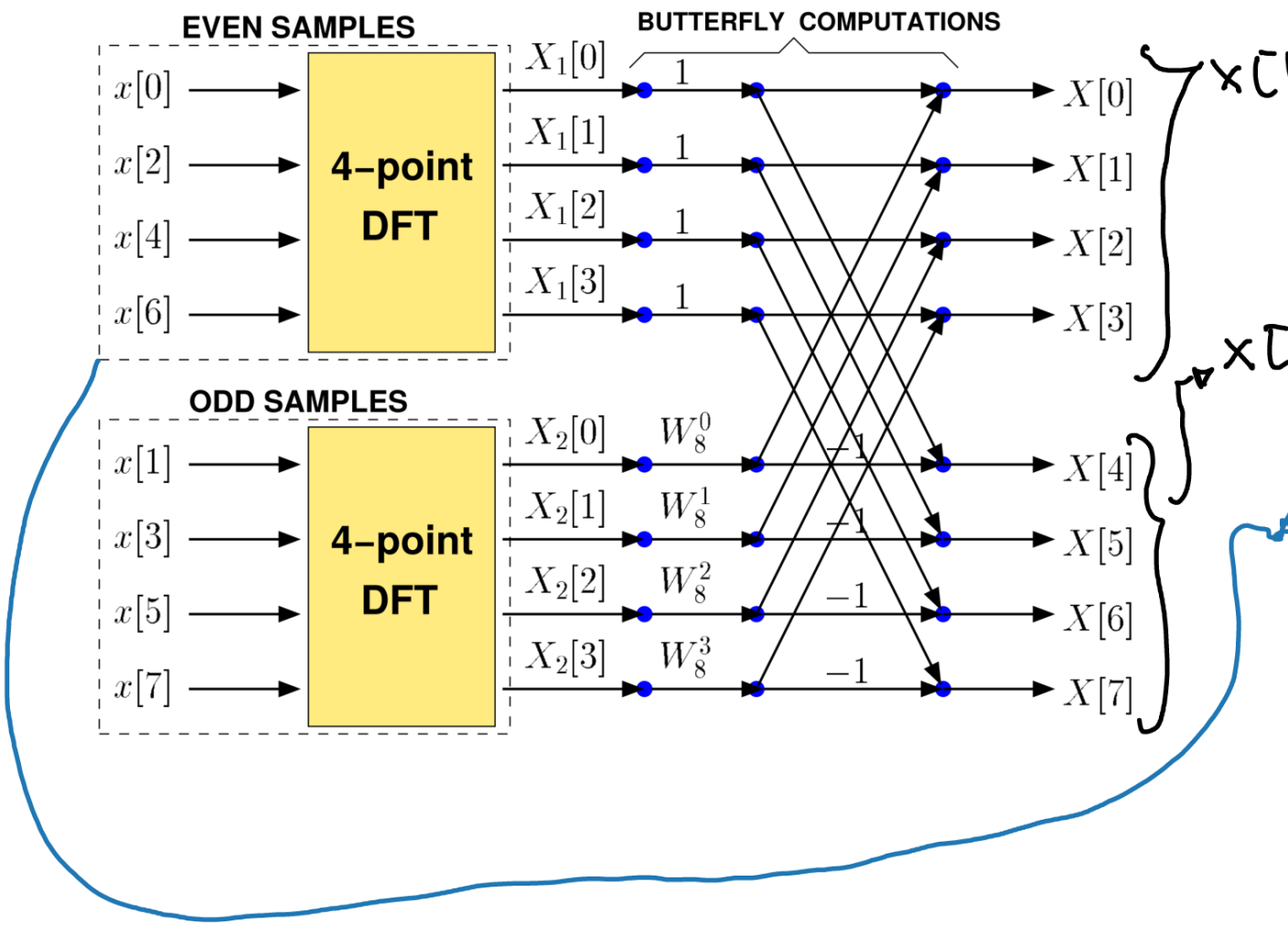
$$\left(e^{-j\frac{2\pi}{N}}\right)^{N/2} = e^{-j\pi} = -1$$

$(\frac{N}{2})^2$ $\quad$ $(\frac{N}{2})^2$



Complex mults: $2 \times \left(\frac{N}{2}\right)^2 = \frac{N^2}{2}$

Eight-point DFT

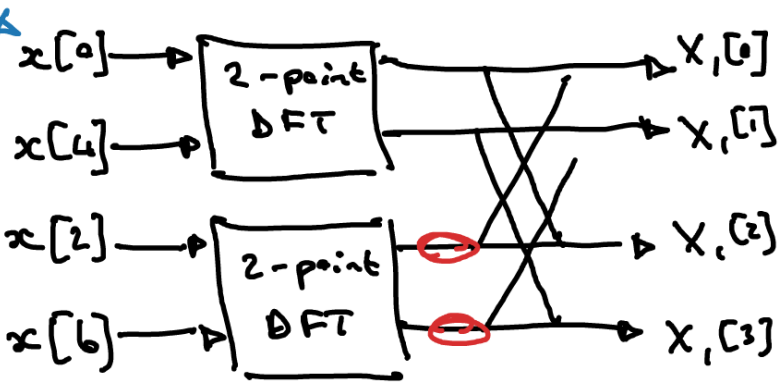


$$X[k] = X_1[k] + W_N^k X_2[k]$$

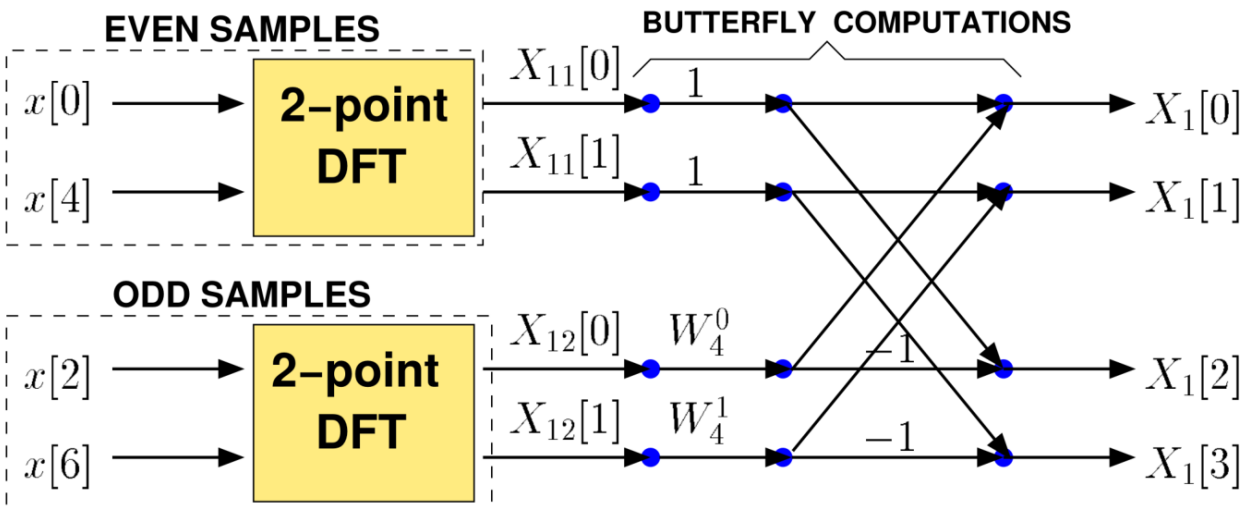
for $k = 0, 1, \ldots, \frac{N}{2} - 1$

$$X\left[k + \frac{N}{2}\right] = X_1[k] - W_N^k X_2[k]$$

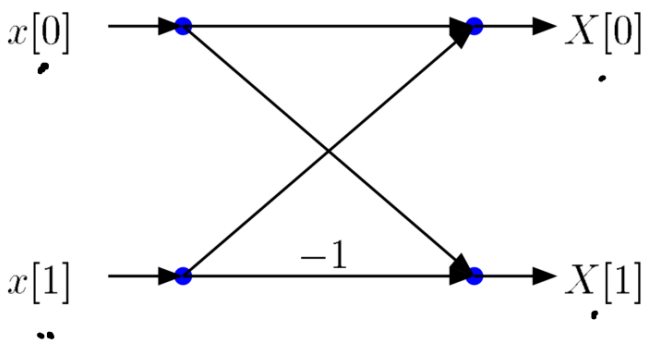
for $k = 0, 1, \ldots, \frac{N}{2} - 1$



Four-point DFT



Two-point DFT



$$X[k] = \sum_{n=0}^{1} x[n] \cdot e^{-j\pi kn}$$

$$= x[0] \cdot e^{0} + x[1] \cdot e^{-j\pi k}$$

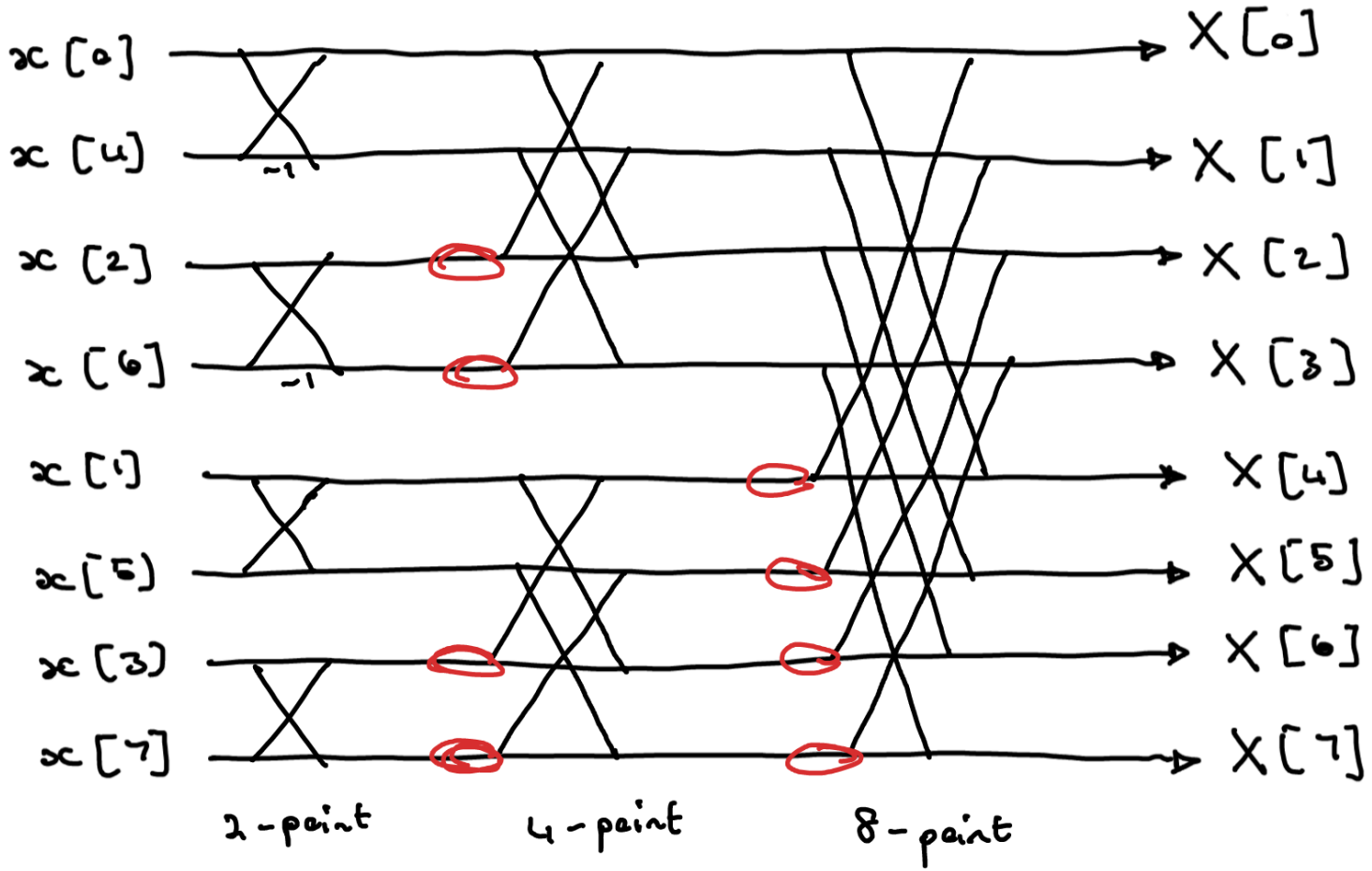
$$X[0] = x[0] + x[1]$$

$$X[1] = x[0] - x[1]$$

$2^{\text{layers}} = N$

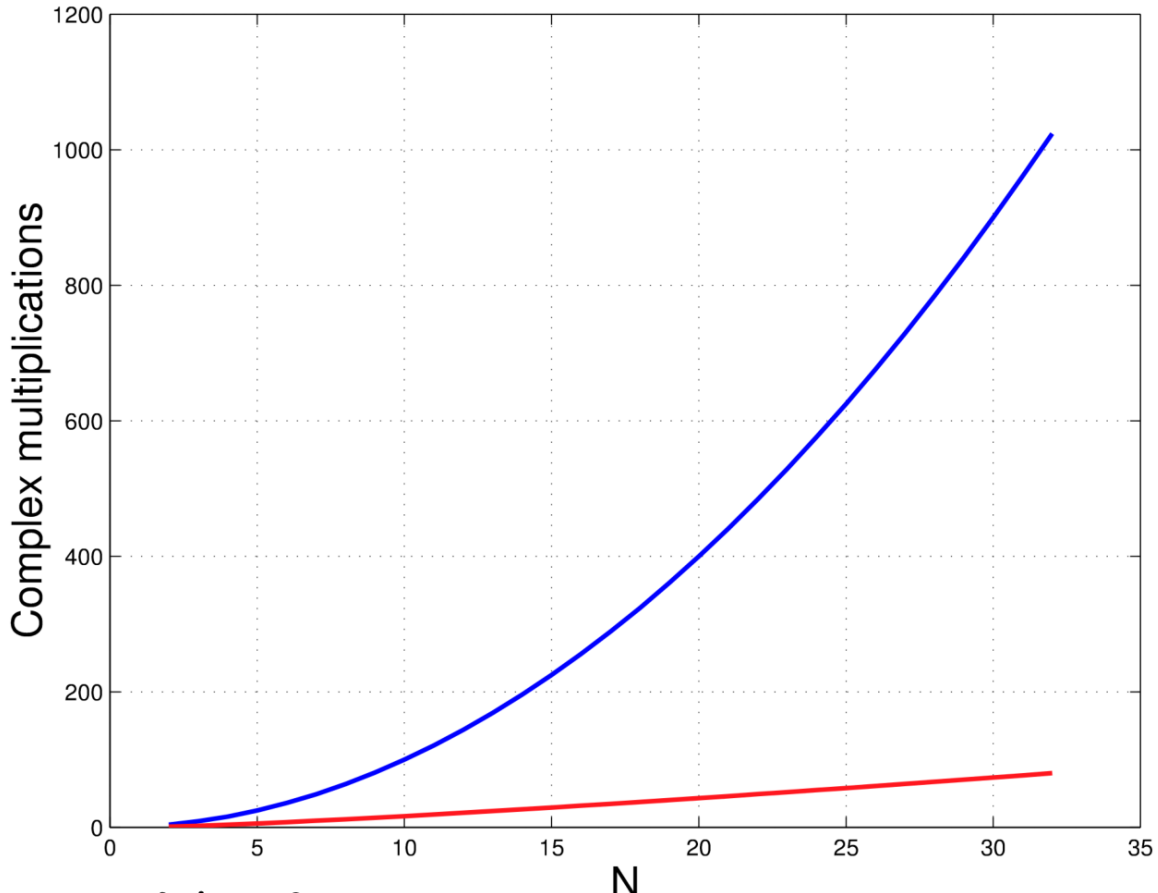
$\text{layers} = \log_2 N$

$\frac{N}{2}$ complex mults per layer



x[217]

Computational complexity



N - point DFT:

- $\log_2 N$ layers
- $\frac{N}{2}$ complex mult per layer

$\therefore \frac{N}{2} \log_2 N$ complex mults for FFT

vs

N^2 for direct DFT

1024-point DFT: 1 048 576 complex mults vs. 5120 for FFT

In-place computation of the FFT

Decimal index	Binary index	Bit-reversed index
0: $x[0]$	$x[000]$	000
1: $x[4]$	$x[100]$	001
2: $x[2]$	$x[010]$	010
3: $x[6]$	$x[110]$	011
4: $x[1]$	$x[001]$	100
5: $x[5]$	$x[101]$	101
6: $x[3]$	$x[011]$	110
$x[7]$	$x[111]$	111

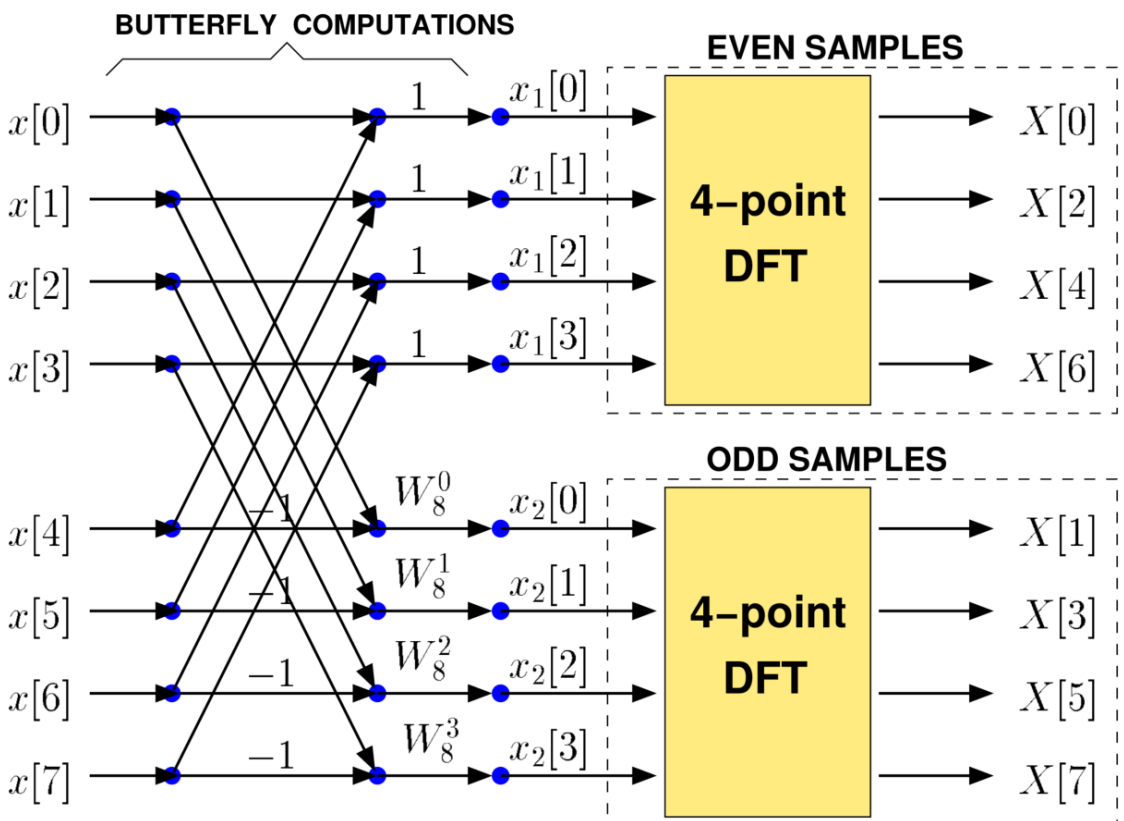
x[217] = ?

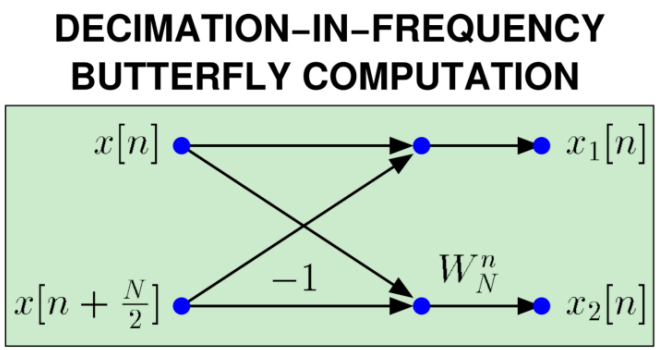
Decimation in frequency

Split $X[k]$ into even and odd sequences:

$X[2k]$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$

$X[2k+1]$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$





Summary

Decimation in time:

- Split $x[n]$ into even and odd parts
- Helpers: $X_1[k]$ and $X_2[k]$
- Input: Bit-reversed ; Output: Nice

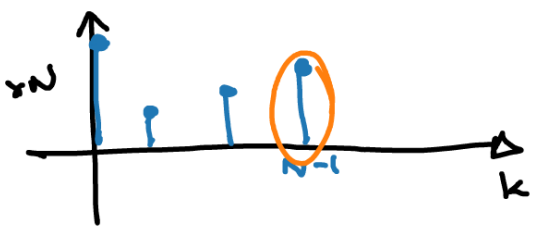
Decimation in frequency:

- Split $X[k]$ into even and odd parts
- Helpers: $x_1[n]$ and $x_2[n]$
- Input: Nice ; Output: Bit-reversed

Calculating IFFTs using FFTs

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}$$

- $\mathrm{FFT}\{X[n]\} = N \cdot x[N-k]$



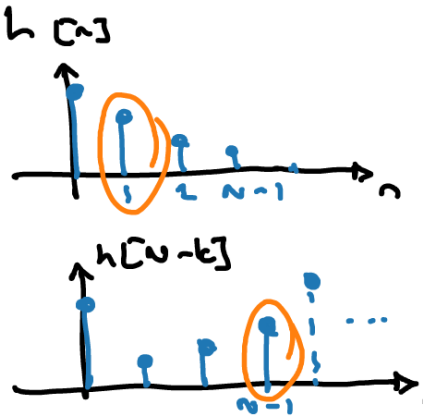
Symmetry for DFT:

if $\mathrm{DFT}\{h[n]\} = H[k]$

then

$\mathrm{DFT}\{H[n]\} = N \cdot h[-k]$

$= N \cdot h[N-k]$



- Swap the indices: what happens at N-k now happens at k

- k becomes n
- Divide by N

What to take away

- Some technical tricks
- Divide and conquer: Store intermediate results and use them later
- You need to understand implications of the tricks:
  - When can you (not) use in-place computations?
  - Why radix-2?
  - Why are things slower with `np.fft.fft` when the input is not radix-2?