Locality of data access

1) Choose recursive FFT, not iterative FFT

\[ \text{DFT}_{kn} = \left( \text{DFT}_k \otimes \text{Im} \right) T_m^u \left( \text{Im} \otimes \text{DFT}_m \right) L'^u \]

\((\text{schema}) = \)

- \(\text{compute } m \text{ many } \text{DFT}_k \cdot D\)
- \(\uparrow \text{ part of } \text{twiddles}\)
- \(\text{writes to same locations as it reads from } \Rightarrow \text{in place}\)

\(\text{DFT+twiddle}(k, *x, *t, \text{stride})\)

\(\uparrow \uparrow \text{size input } \text{twiddles} \text{ output vector}\)

\(\text{cannot handle arbitrary recursion } \Rightarrow \text{in FFTW a base case}\)

Pseudocode:

\[
\text{DFT}(k, *x, *y) = \text{DFTrec}(k, *x, *y, 1, 1) \]

\[
\forall i = 0 : k-1 \text{ \text{DFTrec} } \in \text{base case} \]

\[
\forall j = 0 : m-1 \text{ \text{DFT+twiddle} } \in \text{stage 2 explained later}\]

- \(\text{stride as parameter}\)
- \(\text{writes to different locations than it reads from } \Rightarrow \text{out-of-place}\)

\(\text{DFTrec}(m, *x, *y, \text{inside}, \text{outside})\)

\(\uparrow \uparrow \text{size input output vector}\)

\(\text{the interface handles arbitrary recursions}\)
DAG example

formula: \[ \text{DFT}_2: \text{diag}(1, c) \]

DAG:

- \( x_0 \longrightarrow y_0 \)
- \( x_1 \longrightarrow y_1 \)
- \( \text{loads, mult, adds, stores} \)

or

Simplifications

\[ t_5 = 0 \cdot t_2 \rightarrow t_5 = 0 \]
\[ t_3 = \ldots \]
\[ t_4 = t_3 + t_3 \rightarrow t_5 = t_3 + 4t_1 \]
\[ t_5 = t_4 + t_1 \]
\[ t_3 = 2t_1 \]
\[ t_4 = -2t_2 \rightarrow t_4 = 2t_2 \]
\[ t_5 = t_0 + t_4 \]

Scheduling

\[ \text{DAG} \rightarrow \text{C code (sequential)} \]

Step 1: cut DAG in middle (how?)

Step 2: the two pieces decompose into independent DAGs

Step 3: schedule these recursively using the same method
Adaptivity

Space of algorithms considered by FFTW 2.x

$u = 2^k$

$u = u_1 + \ldots + u_{r + t}$

Best choice found by dynamic programming.