Pseudo code for Cooley-Tukey:

for \( i = 0 \) to \( k-1 \)
for \( j = 0 \) to \( m-1 \)
\[
\text{DFT}(k, x+x+j) \rightarrow x, m)
\]
\[
\text{DFT}(m, xx+i, xy+im, k, l)
\]
\[
\text{recursed until codelet}
\]
\[
\text{codelets}
\]

Optimizations:

1. Fusing iterative steps (for locality)
2. Precomputing constants (twiddles)
3. Optimized basic blocks (codelets, generated)
4. Adaptivity through search over relevant algorithm space
Algorithm space: \( n = 2^k \)

\[ \begin{align*}
2^k \\
2^k \\
2^k \\
\text{not wide} \\
\text{codelet} \\
2^k \\
2^t \end{align*} \]

The "A \& In" Problem

Assume:
- \( 8k \leq 2^{15} \)
- \( B = 2^{10} \) doubles cache
- \( 64k = 2^{12} \) doubles cache-line
- 4-way set associative

- \( A_m \) is \( m \times m \)
- \( n = 2^k \), \( k > 5 \)

\[ A_m \otimes I_n \Leftrightarrow \text{for } i = 0: n-1 \]

\[ y [C_i: n: m] = A_x [C_i: n: m] \]
problems:
in each iteration, only 4 (associativity) cache location are usable for in- and output of A, respectively.
no cache line reuse

Solution 1:
\[
P_m \otimes I_n = L_m \ (I_n \otimes A_m) \ L_n
\]

- requires efficient implementation of \( L \), e.g.,

Solution 2:
Copy stacked in/output to consecutive memory locations.

Solution 3:
Interleave loop bodies for cache line reuse

\[
P_m \otimes I_n \rightarrow (P_m \otimes I_c) \otimes I_{nc}
\]

\[
\uparrow  
\text{loop body}
\]

\[
\downarrow  
\text{loop body}
\]

\[
Y \quad \text{and} \quad X
\]
Interleaving $A \otimes I_n$

instead of

1. op. on $A_i$'s
2. op. on $A_i$'s
.
last op. on $A_i$'s
1. op. on $B_i$'s
.
last op. on $B_i$'s

do

1. op. on $A_i$'s
2. op. on $A_i$'s
.
last op. on $A_i$'s
1. op. on $B_i$'s
.
last op. on $B_i$'s

- $c$ can be a search parameter, or
- choose $c = $ cache line size (in doubles)

The "$A \otimes I_n$" problem occurs in

- Cooley-Tukey FFT for 2-power size
- WHT (see assignment)
- any 2D transform (2-power size)

\[ T_{nxn} = T_n \otimes T_n = (T_n \otimes I_n) (I_n \otimes T_n) \]