

FFTs in IAP — Homework Solutions

16th January 2006

Problem 1

part a

When we perform radix- \sqrt{N} Cooley-Tukey, we decompose a DFT of length $N = 2^{2^m}$ into \sqrt{N} DFTs of length \sqrt{N} , followed by multiplication by N twiddle factors, followed by another \sqrt{N} DFTs of length \sqrt{N} (just the regular C-T steps applied to the case of $N_1 = N_2 = \sqrt{N}$). Thus, the time (#operations) $T(N)$ must satisfy a recurrence $T(N) = 2\sqrt{N}T(\sqrt{N}) + O(N)$. We'll informally write the $O(N)$ term as $\# \cdot N$, where “#” is the asymptotic constant factor hidden in $O(N)$, just as in class. If we expand this recurrence recursively (noting $\sqrt{N} = 2^{2^{m-1}}$), we get something like:

$$\begin{aligned}
 T(N) &= 2N^{1/2}T(N^{1/2}) + O(N) \approx 2N^{1/2}T(N^{1/2}) + \# \cdot N \\
 &= 2N^{1/2} \left(2N^{1/4}T(N^{1/4}) + \# \cdot N^{1/2} \right) + \# \cdot N = 4N^{3/4}T(N^{1/4}) + 2N^{1/2} \cdot \# \cdot N^{1/2} + \# \cdot N \\
 &= 4N^{7/8}T(N^{1/8}) + 4N^{3/4} \cdot \# \cdot N^{1/4} + 2N^{1/2} \cdot \# \cdot N^{1/2} + \# \cdot N \\
 &= \dots = 2^m N \cdot \# \cdot 1 + \dots (m \text{ terms}) \dots + 4N^{3/4} \cdot \# \cdot N^{1/4} + 2N^{1/2} \cdot \# \cdot N^{1/2} + \# \cdot N \\
 &= \# \cdot (1 + 2 + 4 + \dots + 2^m) N \approx \# \cdot 2^m \cdot 2N = O(N \log N).
 \end{aligned}$$

where in the last term we simply summed the geometric series and noted that $2^m = \log_2 N$. The key here is that the recursion went on for m steps (the number of times we can take \sqrt{N} before getting 1), and that each term was (roughly) $\# \cdot N$ times a power of 2 (e.g. $N^{3/4} \cdot \# \cdot N^{1/4}$ is $\# \cdot N$).

part b

Here, we expand out the recurrence in exactly the way we expanded the recurrence in class for the number of operations in the radix-2 case, except that here we *stop* recurring when $N = C$ instead of when $N = 1$. That is, for large $N > C$:

$$\begin{aligned}
 M(N) &= 2M(N/2) + \# \cdot N \\
 &= 4M(N/4) + 2 \cdot \# \cdot N/2 + \# \cdot N \\
 &= 8M(N/8) + 4 \cdot \# \cdot N/4 + 2 \cdot \# \cdot N/2 + \# \cdot N \\
 &= \dots = \frac{N}{C} \cdot \# \cdot C + \frac{N}{2C} \cdot \# \cdot 2C + \dots (\log_2 \frac{N}{C} \text{ terms}) \dots + 4 \cdot \# \cdot N/4 + 2 \cdot \# \cdot N/2 + \# \cdot N \\
 &= O(N \log \frac{N}{C}),
 \end{aligned}$$

where we have used the fact that the number of times you can divide N by 2 until you get C is $\ell = \log_2 \frac{N}{C}$, because this solves $N/2^\ell = C$. Thus, there are $O(\log \frac{N}{C})$ stages of the recursion until we fit in the cache, each of which requires $O(N)$ cache misses, leading to the desired result.

part c

The recurrence $M'(N)$ for cache misses in the radix- \sqrt{N} case is much like the recurrence for the number of operations, except that if $N \leq C$ only $O(N)$ cache misses are required as for the radix-2 case.

$$M'(\sqrt{N}) = \begin{cases} 2\sqrt{N}M'(\sqrt{N}) + O(N) & \text{if } N > C \\ O(N) & \text{if } N \leq C \end{cases} .$$

The solution of this recurrence follows exactly the same process as in part (a), except that we stop taking \sqrt{N} when $N \leq C$ instead of when $N = 1$. Thus, the last term of the recurrence is $2^\ell \frac{N}{C} \cdot \# \cdot C$ instead of $2^m N \cdot \# \cdot 1$, where ℓ is the number of times we take \sqrt{N} until we get C . That is, ℓ satisfies $C = N^{\frac{1}{2^\ell}}$ and thus (taking the log of both sides), we find $2^\ell = \frac{\log N}{\log C}$. It follows that $M'(N) = O(N^{\frac{\log N}{\log C}})$ for large N .

To take the $\frac{N}{C} \rightarrow \infty$ limit, simply let $N = \alpha C$ and let $\alpha \rightarrow \infty$. In this limit, the ratio $M(N)/M'(N) \rightarrow O(\log C)$, showing that the radix-2 algorithm (and any bounded radix) has asymptotically more cache misses for large N/C and large C .