The Fast More Four Do Over

With Yüv

Math Camp Two (Ten to the Cube) and Ten and Five—Week Five

1 How fast can we find $a \cdot b$?

Let's say I give you a real a and a real b, and you want to find the real $a \cdot b$. How fast can you do this, if a and b have n bits each? If we use the way to do this that we were told when we were kids, it does work like this:

a			5	6	9	
b			1	2	4	$\times$
		2	2	7	6	
	1	1	3	8		
	5	6	9			$+$
$a \cdot b$	7	0	5	5	6	

How many acts do we have to do when we do this? Note that for each bit in our real b, we do n acts of the dot map, one on each bit (in base ten) of the real a. So we do n^2 acts of the dot map on bits all in all. We then have many a real (n^2 of them), and we need to add them up. So we also need to do not far from n^2 sums on bits. To do all of it, then, uses not much off from n^2 acts, all said.

So that we can talk in a nice way on how fast we do acts, we will use the big-o sign:

Term 1. For maps $f, g : \mathbb{R} \to \mathbb{R}$, we say that $f(x) = O(g(x))$ if we can find some $C, x_0 \in \mathbb{R}$ such that for all $x > x_0$, we have that

$$|f(x)| \leq C\,|g(x)|$$

So what this does mean is that $f(x) = O(g(x))$ if f and g grow at much the same rate, as long as we don't care for many a real that don't vary.

With this idea in mind, we can see that the way to find $a \cdot b$ that we all know and love does work in time $O(n^2)$ if both a and b have n bits. This is not bad! But it's also not too good. We need to do more well if we want to make our CPUs do a heap of dot acts in the wink of an eye.

To see how to do this, let's do the dot map we did in the past, but this time we won't move the new tens bit to the next line that is not a row:

a			5	6	9	
b			1	2	4	$\times$
			20	24	36	
		10	12	18		
	5	6	9			$+$
$a * b$	5	16	41	42	36	

The list that we get is not $a \cdot b$, but it is not far from $a \cdot b$. We call it $a * b$, and we can get $a \cdot b$ from it if we just move the new tens bit to the next line that is not a row each time. To be more true, we say this:

Term 2. If $\underline{a} = (a_0, \ldots, a_{n-1})$ and $\underline{b} = (b_0, \ldots, b_{n-1})$, then $a * b$ is the line of size $2n + 1$ with kth term

$$(a * b)_k = \sum_{j=0}^{k} a_j b_{k-j}$$

If we make two many-a-term

$$A(x) = \sum_{k=0}^{n-1} a_k x^k \qquad\qquad B(x) = \sum_{k=0}^{n-1} b_k x^k$$

then we see that

$$A(x)B(x) = \sum_{k=0}^{n-1} (a * b)_k x^k$$

As we saw in the past, we love the star map! But even more good than the star map the More Four Do Over, a way to turn the star map into the dot map.

2 The More Four Do Over

The key idea is that if we want to find the dot map $a \cdot b$, step one is to find the star map $a * b$, and we then move the tens bit as we saw. To do this, we use the many-a-term fact that we saw in the past; from the real a we get a line $(a_0, \ldots, a_{n-1})$ and from b we get $(b_0, \ldots, b_{n-1})$, and from them we get the many-a-term $A(x)$ and the many-a-term $B(x)$. Then if we can find each term of the many-a-term $A(x)B(x)$, then we'll know $a * b$, as we saw. So the plan is:

1. Find what are $A(x)$ and $B(x)$ at some x's, and do this fast.
2. For the x's in Step One, find $A(x) \cdot B(x)$ fast.
3. Once we know what is $A(x)B(x)$ at some x's, use this to find out what the many-a-term AB is, and do this fast.
4. Read off the term $(a * b)_k$ for each k, as this is the term by x^k in $AB(x)$.

So it does seem that one way to help go fast is to pick our x's in a sly way so that Step One and Step Two and One both go fast. You may also be a bit wary of Step Two, as it does look like we can't do it if we don't know how to find the dot map fast, but we will deal with that in a bit. But the key idea in the More Four Do Over is to pick the x's in a very wise way; in fact, the x's are just nth many a root of one.

Term 3. The main root of one is $\omega_n = e^{2\pi i/n}$. If we take ω_n to the k for some k, we get the next nth many a root of one, $\omega_n^k = e^{2\pi ik/n}$

Term 4. For a line $\underline{a} = (a_0, \ldots, a_{n-1})$, the More Four Do Over (DFT) is the line $\mathscr{F}\underline{a} = ((\mathscr{F}\underline{a})_0, \ldots, (\mathscr{F}\underline{a})_{n-1})$ with ℓth term

$$(\mathscr{F}\underline{a})_\ell = \sum_{k=0}^{n-1} a_k e^{-2\pi ik\ell/n} = \sum_{k=0}^{n-1} a_k \omega_n^{-k\ell} = A(\omega^{-\ell})$$

for

$$A(x) = \sum_{k=0}^{n-1} a_k x^k$$

If you know some not-the-same idea with the name "More Four Do Over", it may be nice to see what is the same and what is not in this one. We love each More Four Do Over, for each one can turn the star map into the dot map, so all are cool.

In our case, the DFT is just our Step One with the x's just the many a root of one. If you call back the past, you see that we want to both find the DFT fast and to get back $\underline{a}$ from $\mathscr{F}\underline{a}$ fast. We are in luck, as the DFT is the best:

Mini Fact 1. *We can undo the DFT in a way that is much like the DFT:*

$$a_k = \frac{1}{n} \sum_{\ell=0}^{n-1} (\mathscr{F}\underline{a})_\ell e^{2\pi ik\ell/n} = \frac{1}{n} \sum_{\ell=0}^{n-1} (\mathscr{F}\underline{a})_\ell \omega_n^{k\ell} = \frac{1}{n} (\mathscr{F}\underline{A})(\omega^k)$$

when $(\mathscr{F}\underline{A})(x)$ is the many-a-term from the line $\mathscr{F}\underline{a}$. We say

$$a_k = \mathscr{F}^{-1}(\mathscr{F}\underline{a})$$

and this back-DFT has the name IDFT.

How to See. We just bash it:

$$\begin{aligned}
\sum_{\ell=0}^{n-1}(\mathscr{F}\underline{a})_\ell e^{2\pi ik\ell/n} &= \sum_{\ell=0}^{n-1}\left(\sum_{j=0}^{n-1} a_j e^{-2\pi ij\ell/n}\right)e^{2\pi ik\ell/n} \\
&= \sum_{j=0}^{n-1} a_j \left(\sum_{\ell=0}^{n-1} e^{2\pi i(k-j)\ell/n}\right) \\
(*) \qquad &= \sum_{j=0}^{n-1} a_j \quad \begin{cases} n & \text{if } k=j \\ 0 & \text{if } k\neq j \end{cases} \\
&= na_k
\end{aligned}$$

In $(*)$, we used that if $k \neq j$, then $\omega_n^{(k-j)\ell}$ will hit in an even way each root of one as ℓ does vary, so the sum will be zero; if $k = j$, then each term is one, and we have n of them. □

We now know why we like the DFT so much—it does turn the star map into the dot map, and it acts very nice in many ways. So it is time to see that we can do it *fast*.

Big Fact 1 (Like Cool–Two Key (and also Gauß))**.** *Let $n = 2^m$ and $\underline{a} = (a_0, \ldots, a_{n-1})$. Then we can find $\mathscr{F}\underline{a}$ in time $O(n \log n)$.*

We want to make one big note and two less big ones:

- If we try to find the DFT in the dumb way, as

  $$(\mathscr{F}\underline{a})_\ell = \sum_{k=0}^{n-1} a_k \omega_n^{-k\ell}$$

  then just to find $(\mathscr{F}\underline{a})_\ell$ will take $O(n)$ time, as we need to do n dot maps and n sums. So to find all of $\mathscr{F}\underline{a}$ will take $O(n^2)$ time, and that is too slow. So the Like Cool–Two Key idea is very good.
- This Big Fact has us say that $n = 2^m$. This is no big deal, for if not, we can sub in $2^{\lceil \log_2 n \rceil}$ for n and not lose too much time.
- This Big Fact has the name Like Cool–Two Key, but Gauß knew it many a year in the past of them. When they did find it, they did not know that Gauß knew it.

How to See. We just show the way to do the DFT. This is a cut-up-and-win way to do it, and goes like this: we call back that

$$(\mathscr{F}\underline{a})_\ell = \sum_{j=0}^{n-1} a_j \omega_n^{-j\ell}$$

We cut this up into the even and the odd case; let $\underline{e} = (a_0, a_2, \ldots, a_{n-2})$ and $\underline{o} = (a_1, a_3, \ldots, a_{n-1})$. Then

$$\begin{aligned}
(\mathscr{F}\underline{a})_\ell &= \sum_{k=0}^{n/2-1} a_{2k}\omega_n^{-2k\ell} + \sum_{k=0}^{n/2-1} a_{2k+1}\omega_n^{-(2k+1)\ell} \\
&= \sum_{k=0}^{n/2-1} a_{2k}\omega_{n/2}^{-k\ell} + \omega_n^{-\ell}\sum_{k=0}^{n/2-1} a_{2k+1}\omega_{n/2}^{-k\ell} \\
&= (\mathscr{F}\underline{e})_\ell + \omega_n^{-\ell}(\mathscr{F}\underline{o})_\ell
\end{aligned}$$

So we can turn a DFT of size n into two of size $n/2$. So we now do this over and over! Thus, our way to find the DFT is this:

0. If $n = 1$, then the DFT is just $\underline{a}$, so we are done.

1. If not, then $n = 2^m$ for some $m > 0$. Cut up $\underline{a}$ into the even part $\underline{e}$ and the odd part $\underline{o}$.

2. Go to Step Zero to find the DFTs of $\underline{e}$ and $\underline{o}$.

3. Spit out $(\mathscr{F}\underline{e})_\ell + \omega_n^{-\ell}(\mathscr{F}\underline{o})_\ell$.

How fast does this take? Let $T(n)$ be the time it does take to find the DFT on $\underline{a}$ of size n if we use this way to do it. Then we need to do two DFTs, each of size $n/2$, and then to add them back via the code $(\mathscr{F}\underline{a})_\ell = (\mathscr{F}\underline{e})_\ell + \omega_n^{-\ell}(\mathscr{F}\underline{o})_\ell$. This does take time 2, as we need to do one dot act and one sum. So to do this for each ℓ does take time $O(n)$, as we have n such ℓ's. So all in all, we get that

$$T(n) = 2T(n/2) + O(n)$$

At the tth step of this, we have to find 2^t DFTs each of size $n/2^t$. So at the tth step, we do 2^t acts of add-back, and each does take time $O(n/2^t)$. So we take $O(n)$ at each step; also, we have $\log_2 n$ many a step in all. So all told, we take time $O(n \log n)$ in the Like Cool–Two Key way to find the DFT. □

As we saw in the past, the IDFT is near to the same as the DFT, so it is not hard to see that we can also do it in $O(n \log n)$ time, and this is true. So in our big plan to find $a \cdot b$, we now know how to do Step One and Step Two and One in $O(n \log n)$ time.

3 Back to $a \cdot b$

The very big fact as this: we can find $a \cdot b$ fast!

Big Fact 2 (Nice Bush–Road)**.** *If we have a real a and a real b and both have n bits, then we can find $a \cdot b$ in $O(n \log n \log\log n)$ time.*

We won't have time to see the full idea, as it's a bit hard. But the main idea is to not do the DFT, but a Do Over that is near to the same. We work in $\mathbb{C}$ no more; we now work mod $2^N + 1$ for some big N. We have many a root of one here as well, as $2^N \equiv -1$, so $2^{2N/n}$ is an nth root of one. We then note that the Like Cool–Two Key way to find the DFT did not use the fact that we work over $\mathbb{C}$, but just the fact that $(\omega_n)^n = 1$. So we can do the DFT also mod $2^N + 1$, in the same way.

But we need to take care! When we did the DFT in the past, to find the time it does take to find it, we did sum up how many a time we did add or dot some item in $\mathbb{C}$. But now, we can't do this, as a sum and a dot mod $2^N + 1$ may take a lot of time—in fact, we do this just so that we can find dots fast! But we are in luck; it does turn out that we can once more use the cut-up-and-win idea! To find dots mod $2^N + 1$, we use this same way to find dots, but now use a much less big N. If we do this over and over, we can make sure to have all of our sums and dots be very fast, as they act on a real that is not at all big.

Now, if we can find the DFT mod $2^N + 1$ very fast, we can find $a * b$ very fast mod $2^N + 1$, as the big idea was that

$$a * b = \mathscr{F}^{-1}(\mathscr{F}\underline{a} \cdot \mathscr{F}\underline{b})$$

Now, call back that from $a * b$ we can also find $a \cdot b$ mod $2^N + 1$ very fast. Then, as the last step, we make sure that we pick N so big that $2^N + 1 > a \cdot b$, so $a \cdot b$ mod $2^N + 1$ is the same as $a \cdot b$. So we did find $a \cdot b$! And if you do all the hard work, you see that it does take time $O(n \log n \log\log n)$.

Key that lets you know what a word does mean

cut-up-and-win	Divide-and-conquer
DFT, More Four Do Over	Discrete Fourier Transform
dot map, $a \cdot b$	multiplication
Gauß	Gauss
IDFT	Inverse DFT
Like Cool	Cooley
many-a-term	polynomial
More Four	Fourier
Nice Bush	Schönhage
Road	Strassen
star map, $a * b$	convolution
Two Key	Tukey