26 Feb 2018 Divide & Conquer I: Multiplication

Announcements: ① ACSU serving cookies (sponsor: Stevens Capital Management) outside Gates G01 after prelim.
② Need extra time for a medical reason? Come see me.

$$
\begin{array}{r}
4820 \\
\times\ 2018 \\
\hline
38560 \\
4820 \\
0000 \\
9640 \\
\hline
9726760
\end{array}
$$

Fürer (2007): Multiply two n-bit integers in time $O(n \log n \, 2^{\log_*(n)})$.

A faster algorithm based on divide & conquer.
Start with multiplying degree $n-1$ univariate polynomials.

$$
\begin{array}{r}
4x^3 + 8x^2 + 2x + 0 \\
2x^3 + \phantom{8x^2 +{}} x + 8 \\
\hline
32x^3 + 64x^2 + 16x \\
4x^4 + 8x^3 + 2x^2 \\
8x^6 + 16x^5 + 4x^4 \\
\hline
8x^6 + 16x^5 + 8x^4 + 40x^3 + 68x^2 + 16x
\end{array}
$$

Game plan: ① Develop faster alg for polynomial multiplication.

② Integer in binary = polynomial with $\{0,1\}$ coeffs evaluated at $x=2$.
So use the polyn mult method and substitute $x=2$ in final step

A tricky identity for multiplying degree 1 polynomials.

NOT TRICKY!

$$(a_1x + a_0)(b_1x + b_0) = a_1b_1x^2 + (a_1b_0 + a_0b_1)x + a_0b_0$$

This is not adding numbers. This is outputting the character '+'.

4 multiplications, 1 addition

$$c_0 := a_0b_0 \qquad c_\infty := a_1b_1 \qquad c_1 := (a_1 + a_0)(b_1 + b_0) = a_1b_1 + a_1b_0 + a_0b_1 + a_0b_0$$

$$(a_1x + a_0)(b_1x + b_0) = c_\infty x^2 + (c_1 - c_0 - c_\infty)x + c_0$$

3 multiplications 2 additions 2 subtractions

$$c_0 = a_0 b_0 \qquad c_\infty = a_1 b_1 \qquad c_1 = (a_1 + a_0)(b_1 + b_0)$$

$$(a_1 x + a_0)(b_1 x + b_0) = c_\infty x^2 + (c_1 - c_0 - c_\infty) x + c_0$$

Now suppose we have two polynomials of degree n-1.
For convenience suppose n is a power of 2, $n = 2^k$.
(If not, pre-pend some zero coefficients, e.g.

$$\underbrace{4x^5 + 2x^2 - 1}_{\text{degree } 5} = \underbrace{\cancel{0}x^7 + \cancel{0}x^6 + 4x^5 + \cancel{0}x^4 + \cancel{0}x^3 + 2x^2 + \cancel{0}x - 1}_{\text{degree } 7.}$$

Say we're multiplying $A(x) \cdot B(x)$ where $A(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$
$B(x) = b_0 + b_1 x + b_2 x^2 + \dots + b_{n-1} x^{n-1}$

$$A(x) = A_0(x) + A_1(x)\, x^{n/2} \qquad \deg(A_0), \deg(A_1) < \tfrac{n}{2}.$$
$$B(x) = B_0(x) + B_1(x)\, x^{n/2}$$

E.g. if $A(x) = \boxed{\cancel{0}x^7 + \cancel{0}x^6 + 4x^5 + \cancel{0}x^4} + \boxed{\cancel{0}x^3 + 2x^2 + \cancel{0}x - 1}$

then $A_0(x) = 2x^2 - 1 \qquad A_1(x) = 4x$

$A_1(x) \cdot x^4 = 4x^5$
$\therefore A_1(x) = 4x$

$$A(x) \cdot B(x) = \left[A_0(x) + A_1(x) \cdot x^{n/2} \right] * \left[B_0(x) + B_1(x) \cdot x^{n/2} \right]$$

$$C_0(x) := A_0(x) \cdot B_0(x) \qquad C_\infty(x) = A_1(x) \cdot B_1(x)$$
$$C_1(x) = [A_0(x) + A_1(x)] \cdot [B_0(x) + B_1(x)].$$

As before...

$$A(x) \cdot B(x) = C_0(x) + \left[C_1(x) - C_0(x) - C_\infty(x) \right] \cdot x^{n/2} + C_1(x)\, x^n.$$

<u>Conclusion.</u> To multiply two degree n-1 polynomials, we need to:

- add 2 pairs of degree $\frac{n}{2} - 1$ polynomials.
- multiply 3 pairs of degree $\frac{n}{2} - 1$ polynomials,
- subtract 2 pairs of degree $n-2$ polynomials.
- add 2 pairs of degree $2n-2$ polynomials.

Polynomial addition / subtraction of degree n polynomials takes $O(n)$ arithmetic operations. (Adding, subtracting coefficients.)

Multiplying degree n-1 polynomials using this algorithm takes $T(n)$ where

$$T(n) = 3 \cdot T(\tfrac{n}{2}) + O(n).$$

treat this as the definition of $T(n)$.

How to solve $T(n) = 3 \cdot T(\frac{n}{2}) + O(n)$?

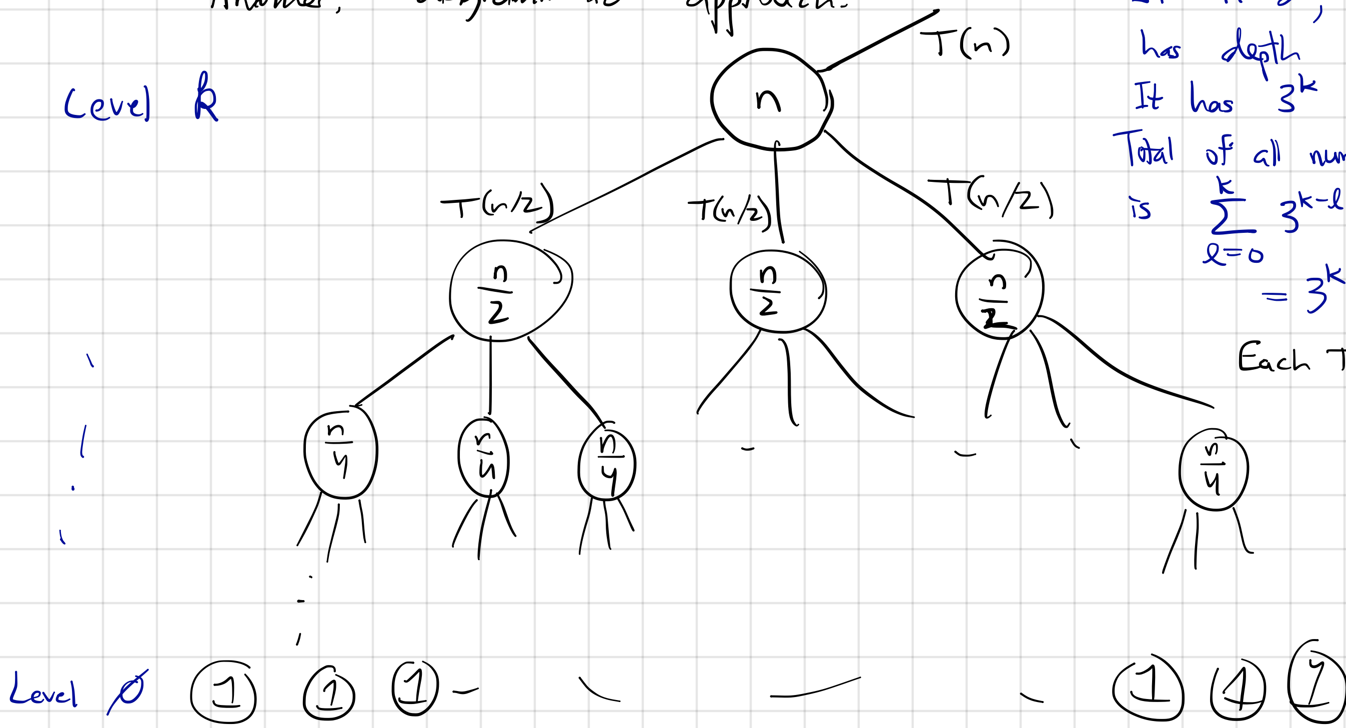
One approach that is good for people who like memorization: use the "Master Theorem." (In K.&T. ~~textbook~~, Chapter 2.)

Another, diagrammatic approach.

If $n = 2^k$, the tree has depth k.
It has 3^k leaves.
Total of all numbers inside nodes is $\sum_{\ell=0}^{k} 3^{k-\ell} \cdot 2^{\ell}$

$= 3^k \cdot \sum_{\ell=0}^{k} \left(\frac{2}{3}\right)^{\ell}$.

Each $T(\frac{n}{4})$



$$3^k \cdot \sum_{\ell=0}^{k} \left(\frac{2}{3}\right)^{\ell} < 3^{k+1} = 3^{1 + \log_2(n)}$$

$$= 3 \cdot 3^{\log_2(n)}$$

$$= 3 \cdot 2^{\log_2(3) \cdot \log_2(n)}$$

$$= 3 \cdot \left(2^{\log_2(n)}\right)^{\log_2(3)}$$

$$= 3 \cdot n^{\log_2(3)} \approx 3 \cdot n^{1.58}$$