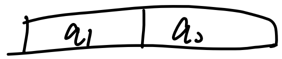
Integer multiplication using divide and conquer. in $O(n^2)$

Pf: a, b n-bit numbers.

$$a = a_1 \cdot 2^{\frac{n}{2}} + a_0$$

$$b = b_1 \cdot 2^{\frac{n}{2}} + b_0$$



shift in $O(n)$

$$a \cdot b = \left(a_1 \cdot 2^{\frac{n}{2}} + a_0\right)\left(b_1 \cdot 2^{\frac{n}{2}} + b_0\right)$$

$$= \underline{a_1 b_1}\, 2^{n} + \left(\underline{a_1 b_0} + \underline{a_0 b_1}\right) 2^{\frac{n}{2}} + \underline{a_0 b_0}$$

[conquer = $\left\{ \begin{array}{l} 3 \text{ addition} \\ 2 \text{ shifts} \end{array} \right\}$ in $O(n)$]

4 subproblems $T(\frac{n}{2})$ ["divide"]

Correctness : by induction ?

$$T(n) \le 4\,T\left(\frac{n}{2}\right) + O(n) \in O(n^2)$$

Do better: Karatsuba, in $O(n^{\log_2 3})$

Pf: Same set up.

$$a \cdot b = (a_1 \cdot 2^{\frac{n}{2}} + a_0)(b_1 \cdot 2^{\frac{n}{2}} + b_0)$$
$$= a_1 b_1 2^n + (a_1 b_0 + a_0 b_1) 2^{\frac{n}{2}} + a_0 b_0$$

idea: use 3 recursive calls.

Lemma: $(a_1 - a_0)(b_1 - b_0) = a_1 b_1 + a_0 b_0 - (a_0 b_1 + a_1 b_0) \longrightarrow$ three num we want

Note that they are $\frac{n}{2}$-bit numbers. $\Rightarrow T(\frac{n}{2})$

algo: – recursively compute:

$$\left.\begin{cases} a_1 b_1 \\ a_0 b_0 \\ (a_1 - a_0)(b_1 - b_0) \end{cases}\right\} \rightarrow 3T(\tfrac{n}{2})$$

– compute $(a_0 b_1 + a_1 b_0)$ by subtraction $\} \rightarrow O(n)$

– compute $a \cdot b$ $\} \rightarrow O(n)$

Complexity: $O(n^{\log_2 3})$