$$
\begin{array}{r}
1100 \\
\times\,1101 \\
\hline
1100 \\
0000 \\
1100 \\
1100 \\
\hline
10011100
\end{array}
$$

(b)

even as an algorithmic question. But, in fact, elementary schoolers are taught a concrete (and quite efficient) algorithm to multiply two n-digit numbers x and y. You first compute a "partial product" by multiplying each digit of y separately by x, and then you add up all the partial products. (Figure 5.8 should help you recall this algorithm. In elementary school we always see this done in base-10, but it works exactly the same way in base-2 as well.) Counting a single operation on a pair of bits as one primitive step in this computation, it takes $O(n)$ time to compute each partial product, and $O(n)$ time to combine it in with the running sum of all partial products so far. Since there are n partial products, this is a total running time of $O(n^2)$.

First Idea

$$\begin{aligned} xy &= (x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0) \\ &= \underline{x_1 y_1} \cdot 2^n + (\underline{x_1 y_0} + \underline{x_0 y_1}) \cdot 2^{n/2} + \underline{x_0 y_0}. \end{aligned} \tag{5.1}$$

$$T(n) \leq 4T(n/2) + cn$$

$$T(n) \leq = O(n^2)$$

Second Idea

$$\underline{(x_1 + x_0)(y_1 + y_0)} = \underline{x_1 y_1} + \underline{x_1 y_0 + x_0 y_1} + \underline{x_0 y_0}.$$

$$T(n) \leq 3T(n/2) + cn$$

$$O(n^{\log_2 3}) = O(n^{1.59}).$$

```
Recursive-Multiply(x,y):
  Write x = x_1 · 2^{n/2} + x_0
        y = y_1 · 2^{n/2} + y_0
  Compute x_1 + x_0 and y_1 + y_0
  p = Recursive-Multiply(x_1 + x_0,  y_1 + y_0)
  x_1y_1 = Recursive-Multiply(x_1, y_1)
  x_0y_0 = Recursive-Multiply(x_0, y_0)
  Return x_1y_1 · 2^n + (p − x_1y_1 − x_0y_0) · 2^{n/2} + x_0y_0
```