Karatsuba algorithm

Anatoly Karatsuba, 1960.

Let \(a\) and \(b\) be two \(n\)-bit integers, whose binary representation is \(a = (a_{n-1}, a_{n-2}, \ldots, a_1, a_0)\) and \(a = (b_{n-1}, b_{n-2}, \ldots, b_1, b_0)\), respectively.

**Goal:** fast algorithm for finding a binary representation of the product \(a \cdot b\).

**Note:** a straightforward approach for multiplication requires \(O(n^2)\) bit operations. For comparison, addition requires only \(O(n)\) bit operations.

**Main step**

**Idea:** we trade a multiplication for a constant number of additions, which is a “cheaper” operation.

We write the numbers \(a\) and \(b\) as follows:

\[
\begin{align*}
a & : \quad a_{n-1}, a_{n-2}, \ldots, a_{n/2}, \quad a_{n/2-1}, a_{n/2-2}, \ldots, a_0 \\
& \quad \underbrace{a_L}_{a_R} \\

b & : \quad b_{n-1}, b_{n-2}, \ldots, b_{n/2}, \quad b_{n/2-1}, b_{n/2-2}, \ldots, b_0 \\
& \quad \underbrace{b_L}_{b_R}
\end{align*}
\]

We obtain that

\[
\begin{align*}
a & = a_L \cdot 2^{n/2} + a_R \\
b & = b_L \cdot 2^{n/2} + b_R
\end{align*}
\]

Then,

\[
a \cdot b = (a_L \cdot 2^{n/2} + a_R) (b_L \cdot 2^{n/2} + b_R)
\]

\[
= a_L b_L \cdot 2^n + (a_L b_R + a_R b_L) \cdot 2^{n/2} + a_R b_R .
\]

Does this approach help to reduce the time complexity?

\[T(n) = 4T(n/2) + O(n) = \cdots = O(n^2) .\]

---

No gain.

Instead, consider the following trick. Compute $a_L b_L$, $a_R b_R$ and $(a_L + a_R)(b_L + b_R)$ (three multiplications in total). Then,

$$a_L b_R + a_R b_L = (a_L + a_R)(b_L + b_R) - a_L b_L - a_R b_R.$$

**Recursive step and time complexity**

In order to compute the three products $a_L b_L$, $a_R b_R$ and $(a_L + a_R)(b_L + b_R)$ we apply recursion.

The time complexity is given by a recursive relation:

$$T(n) = 3 \cdot T(n/2) + O(n).$$

By solving this recursion we obtain

$$T(n) = \frac{3 \cdot T(n/2) + O(n)}{3} = \frac{3^2 \cdot T(n/4) + O(n)}{3} = \cdots$$

The number of levels of recursion is $\log_2 n$. Thus, we obtain

$$T(n) = O\left(3^{\log_2 n} + n \cdot \left(\frac{3}{2}\right)^{\log_2 n}\right) = O\left(n^{\log_3 3} + n^{1+\log_2 \frac{3}{2}}\right) = O(n^{\log_2 3}) \approx O(n^{1.59}).$$

Note that the second transition holds because

$$\log_3 \left(3^{\log_2 n}\right) = \log_2 n = \log_2 3 \cdot \log_3 n = \log_3 \left(n^{\log_2 3}\right),$$

and therefore

$$3^{\log_2 n} = n^{\log_2 3}.$$

**Conclusion:** the running time of the Karatsuba algorithm is approximately $O(n^{1.59})$.

**Strassen algorithm\(^2\)**

Discovered by Volker Strassen, 1969.

Let $X$ and $Y$ be two $n \times n$ real-valued matrices.

Goal: compute the matrix \( Z = X \cdot Y \), where

\[ Z_{i,j} = \sum_{k=1}^{n} X_{i,k} Y_{k,j} \text{ (standard matrix multiplication)}. \]

In a straightforward approach, computing each \( Z_{i,j} \) takes \( O(n) \) real-valued multiplications. In total this approach requires \( O(n^3) \).

Matrix addition requires only \( O(n^2) \) real-valued operations. Therefore, the main idea will be to trade a multiplication for several additions.

Denote

\[
X = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}.
\]

Then,

\[
X \cdot Y = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}.
\]

Does this approach help to reduce time complexity?

\[
T(n) = 8 \cdot T(n/2) + O(n^2) = \cdots = O(n^3).
\]

No gain.

Idea of Strassen. Write:

\[
X \cdot Y = \begin{pmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{pmatrix},
\]

where

\[
\begin{align*}
P_1 &= A(F - H) & P_5 &= (A + D)(E + H) \\
P_2 &= (A + B)H & P_6 &= (B - D)(G + H) \\
P_3 &= (C + D)E & P_7 &= (A - C)(E + F) \\
P_4 &= D(G - E)
\end{align*}
\]

Complexity analysis

The total number of operations is given by

\[
T(n) = 7 \cdot T\left(\frac{n}{2}\right) + O(n^2)
\]

\[
= 7 \cdot \left( 7 \cdot T\left(\frac{n}{2^2}\right) + O\left(\frac{n^2}{2^2}\right) \right) + O(n^2)
\]

\[
= 7^2 \cdot T\left(\frac{n}{2^2}\right) + \left(1 + \frac{7}{2^2}\right) \cdot O(n^2)
\]

\[
= 7^3 \cdot T\left(\frac{n}{2^3}\right) + \left(1 + \frac{7}{2^2} + \frac{7^2}{2^4}\right) \cdot O(n^2)
\]

\[
= \ldots
\]
Number of levels of recursion is $\log_2 n$. The resulting time complexity is

$$T(n) = O\left(7^{\log_2 n} + n^2 \cdot \left(\frac{7}{4}\right)^{\log_2 n}\right) = O\left(n^{\log_2 7} + n^2 + \log_2 \frac{7}{4}\right) = O\left(n^{\log_2 7}\right) \approx O\left(n^{2.81}\right).$$

Note: the fastest known algorithm for matrix multiplication is due to Francois Le Gall (2014) with $O\left(n^{2.37}\right)$. The known lower bound on the complexity is $O\left(n^2\right)$. 