Análise de Algoritmos - T1

Renato Rodrigues Vandré Leal

Universidade Federal de Uberlândia

17/09/2018

Multiplicação de N bits

O problema

$$
\begin{array}{r}
12 \\
\times\, 13 \\
\hline
36 \\
12 \\
\hline
156 \\
\end{array}
\qquad\qquad
\begin{array}{r}
1100 \\
\times\, 1101 \\
\hline
1100 \\
0000 \\
1100 \\
1100 \\
\hline
10011100 \\
\end{array}
$$

(a) **(b)**

Figure 5.8 The elementary-school algorithm for multiplying two integers, in (a) decimal and (b) binary representation.

```
               X1          X0
          x    Y1          Y0
          -----------------
              X1Y0      X0Y0
+    Y1X1     Y1X0
--------------------------
     Y1X1 (X1Y0+Y1X0) X0Y0
```

A derivação formal desta ideia se dá da seguinte maneira:

$$XY = (X1\ 10^{n/2} + X0)\ (Y1\ 10^{n/2} + Y0)$$
$$XY = X1Y1\ 10^{n} + X1Y0\ 10^{n/2} + X0Y1\ 10^{n/2} + X0Y0$$
$$XY = X1Y1\ 10^{n} + (X1Y0 + X0Y1)\ 10^{n/2} + X0Y0$$

Divide and Conquer

```python
def divideConquer(x, y, verbose=False):
    x = str(x)
    y = str(y)
    if len(x) == 1 and len(y) == 1:
        return int(x) * int(y)

    if len(x) < len(y):
        x = zeroPad(x, len(y) - len(x))
    elif len(y) < len(x):
        y = zeroPad(y, len(x) - len(y))

    n = len(x)
    m = int(math.ceil(float(n) / 2))
    X1 = int(x[:m])
    X0 = int(x[m:])
    Y1 = int(y[:m])
    Y0 = int(y[m:])

    X0Y0 = divideConquer(X0, Y0)
    X0Y1 = divideConquer(X0, Y1)
    X1Y0 = divideConquer(X1, Y0)
    X1Y1 = divideConquer(X1, Y1)

    BZeroPad = n - m
    AZeroPad = BZeroPad * 2
    A = int(zeroPad(str(X1Y1), AZeroPad, False))
    B = int(zeroPad(str(X1Y0 + X0Y1), BZeroPad, False))
    C = X0Y0

    return A + B + C
```

$\Theta(n)$

$4\,T(\lceil n/2 \rceil)$

$\Theta(n)$

Divide and Conquer: Análise

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 4T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$

Karatsuba

```python
def karatsuba(x, y, verbose=False):
    x = str(x)
    y = str(y)
    if len(x) == 1 and len(y) == 1:
        return int(x) * int(y)

    if len(x) < len(y):
        x = zeroPad(x, len(y) - len(x))
    elif len(y) < len(x):
        y = zeroPad(y, len(x) - len(y))

    n = len(x)
    m = int(math.ceil(float(n) / 2))
    X1 = int(x[:m])
    X0 = int(x[m:])
    Y1 = int(y[:m])
    Y0 = int(y[m:])

    X1Y1 = karatsuba(X1, Y1)
    X0Y0 = karatsuba(X0, Y0)
    P = karatsuba(X1 + X0, Y1 + Y0)

    BZeroPad = n - m
    AZeroPad = BZeroPad * 2
    A = int(zeroPad(str(X1Y1), AZeroPad, False))
    B = int(zeroPad(str(P - X1Y1 - X0Y0), BZeroPad, False))
    C = X0Y0

    return A + B + C
```

$\Theta(n)$

$3\,T(\lceil n/2 \rceil)$

$\Theta(n)$

Karatsuba: Análise

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 3T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$

$$\implies \quad T(n) = \Theta(n^{\log_2 3}) = O(n^{1.585})$$

História da complexidade assintótica do problema

year	algorithm	bit operations
12xx	**grade school**	$O(n^2)$
1962	**Karatsuba–Ofman**	$O(n^{1.585})$
1963	**Toom-3, Toom-4**	$O(n^{1.465})$, $O(n^{1.404})$
1966	**Toom–Cook**	$O(n^{1+\varepsilon})$
1971	**Schönhage–Strassen**	$O(n \log n \cdot \log \log n)$
2007	**Fürer**	$n \log n \, 2^{O(\log^* n)}$
2018	**Harvey–van der Hoeven**	$O(n \log n \cdot 2^{2 \lg^* n})$
	???	$O(n)$

Fonte

Algorithm Design - Jon Kleinberg, Eva Tardos - Copyright © 2005 Pearson-Addison Wesley

Divide-and-conquer multiplication - Carl Burch - disponível em: http://www.cburch.com/csbsju/cs/160/notes/31/1.html