Design and Analysis of Algorithms I

Master Method

Motivation

Integer Multiplication Revisited

<u>Motivation</u>: potentially useful algorithmic ideas often need mathematical analysis to evaluate.

<u>Recall</u>: grade-school multiplication algorithm uses $\Theta(n^2)$ operations to multiply two n-digit numbers.

A Recursive Algorithm

Recursive approach

Write $x = 10^{n/2} a + b$ $\quad y = 10^{n/2} c + d$

[where a, b, c, d are $\frac{n}{2}$-digit numbers]

So:

$$x \cdot y = \boxed{10^n ac + 10^{\frac{n}{2}}(ad + bc) + bd} \quad (*)$$

Algorithm #1: recursively compute ac, ad, bc, bd, then compute $(*)$ in obvious way.

A Recursive Algorithm

$T(n)$ = maximum number of operations this algorithm needs to multiply two n-digit numbers.

<u>Recurrence</u>: express $T(n)$ in terms of running time of recursive calls.

<u>Base case</u>: $T(1) \le$ a constant.

<u>For all $n>1$</u>: $T(n) \le \underbrace{4T\left(\frac{n}{2}\right)}_{\text{work done by recursive calls}} + \overbrace{O(n)}^{\text{work done here}}$

A Better Recursive Algorithm

Algorithm #2 (Gauss): recursively compute

$ac^{(1)}$, $bd^{(2)}$, $(a+b)(c+d)^{(3)}$ [recall $ad+bc = (3)-(1)-(2)$]

New Recurrence:

Base case: $T(1) \le$ a constant.

Which recurrence best describes the running time of Gauss's algorithm for integer multiplication?

- ○ $T(n) \leq 2T(n/2) + O(n^2)$
- ○ $3T(n/2) + O(n)$
- ○ $4T(n/2) + O(n)$
- ○ $4T(n/2) + O(n^2)$

A Better Recursive Algorithm

Algorithm #2 (Gauss): recursively compute $ac^{①}$, $bd^{②}$, $(a+b)(c+d)^{③}$ [recall $ad+bc$ = ③ − ① − ②]

New Recurrence:

Base case: $T(1) \le$ a constant.

For all $n > 1$: $T(n) \le \underbrace{3T\left(\frac{n}{2}\right)}_{\text{work done in recursive calls}} + \overbrace{O(n)}^{\text{work done here}}$

Tim Roughgarden