Algorithms 04

CS201

Kaustuv Nag

Multiplication of Two Integers

- Problem Statement: Given two n digit long integers x and y in base r, find $x \times y$.
- We usually assume that it takes a constant time to perform the multiplication of two integers.
  - Makes life simpler.
  - Numbers are usually relatively small.
  - We can do multiplications relatively fast.
- The naive approach takes $O(n^2)$ running time.

Multiplication of Two Integers: Divide and Conquer

- Divide each number into two halves:

$$
\begin{aligned}
x &= x_H \times r^{n/2} + x_L \\
y &= y_H \times r^{n/2} + y_L
\end{aligned}
$$

- Conmbine:

$$
\begin{aligned}
xy &= (x_H \times r^{n/2} + x_L) \times (y_H \times r^{n/2} + y_L) \\
&= x_H y_H r^n + (x_H y_L + x_L y_H) r^{n/2} + x_L y_L
\end{aligned}
$$

- Runnting Time:

$$
\begin{aligned}
T(n) &= 4T(n/2) + O(n) \\
&= O(n^2)
\end{aligned}
$$

Multiplication of Two Integers: Karatsuba's Algorithm

- Instead of four subproblems, we can have only three subproblems:

$$
\begin{aligned}
a &= x_H y_H \\
d &= x_L y_L \\
e &= (x_H + x_L)(y_H + y_L) - a - d
\end{aligned}
$$

- Then:

$$xy = ar^n + er^{n/2} + d$$

- Runnting Time:

$$
\begin{aligned}
T(n) &= 3T(n/2) + O(n) \\
&= O(n^{\lg 3}) \approx O(n^{1.584})
\end{aligned}
$$

Multiplication of Two Integers: Karatsuba's Algorithm

An Example

- Compute 1234×4321.
- Subproblems:

$$\begin{aligned} a &= 12 \times 43 \\ d &= 34 \times 21 \\ e &= (12+34) \times (43+21) - a - d \end{aligned}$$

- Recrusively solve $a = 12 \times 43$:

$$\begin{aligned} a_a &= 1 \times 4 = 4 \\ d_a &= 2 \times 3 = 6 \\ e_a &= (1+2) \times (4+3) - a_a - d_a = 11 \\ a &= 4 \times 10^2 + 11 \times 10 + 6 = 516 \end{aligned}$$

Multiplication of Two Integers: Karatsuba's Algorithm

An Example

- Recrusively solve $d = 34 \times 21$:

$$\begin{aligned} a_d &= 3 \times 2 = 6 \\ d_d &= 4 \times 1 = 4 \\ e_d &= (3+4) \times (2+1) - a_d - d_d = 11 \\ d &= 6 \times 10^2 + 11 \times 10 + 4 = 714 \end{aligned}$$

- Solve subproblem $e = 46 \times 64 - a - d = 2944 - a - d = 1714$
  - Recrusively solve $e' = 46 \times 64$:

$$\begin{aligned} a_{e'} &= 4 \times 6 = 24 \\ d_{e'} &= 6 \times 4 = 24 \\ e_{e'} &= (4+6) \times (6+4) - a_{e'} - d_{e'} = 52 \\ e' &= 24 \times 10^2 + 52 \times 10 + 24 = 2944 \end{aligned}$$

Multiplication of Two Integers: Karatsuba's Algorithm

An Example

- Combine

$$\begin{aligned} 1234 \times 4321 &= 516 \times 10^4 + 1714 \times 10^2 + 714 \\ &= 5332114 \end{aligned}$$

The Master Theorem

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

The Intuition behind the Master Theorem

- We compare the function $f(n)$ with the function $n^{\log_b a}$ The larger of the two functions determines the solution to the recurrence.
- Case 1: If the function $n^{\log_b a}$ is *polynomially* larger, then the solution is $T(n) = \Theta(n^{\log_b a})$.
- Case 3: If the function $f(n)$ is *polynomially* larger, then the solution is $T(n) = \Theta(f(n))$.
- Case 2: If the two functions are the same size, we multiply by a logarithmic factor, and the solution is $T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(f(n) \lg n)$.

The Master Theorem: Examples

- $T(n) = 9T(n/3) + n$
  - $a = 9$, $b = 3, f(n) = n$.
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$.
  - $f(n) = O(n^{\log_3 9-\varepsilon})$, where $\varepsilon = 1$.
  - Case 1: $T(n) = \Theta(n^2)$.
- $T(n) = T(2n/3) + 1$
  - $a = 1$, $b = 3/2, f(n) = 1$.
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$.
  - $f(n) = n^{\log_{3/2} 1} = \Theta(1)$.
  - Case 2: $T(n) = \Theta(\lg n)$.

The Master Theorem: Examples

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$, $b = 4$, $f(n) = n \lg n$.
  - $n^{\log_b a} = n^{\log_4 3} \approx O(n^{0.793})$.
  - $f(n) = \Omega(n^{\log_4 3 + \varepsilon})$, where $\varepsilon \approx 0.2$.
  - Case 3 may be applicable.
  - For sufficiently large n, $af(n/b) = 3(n/4)\lg(n/4) \leq (3/4) n \lg n = cf(n)$ for $c = 3/4$.
  - Case 3: $T(n) = \Theta(n \lg n)$
- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$, $b = 2$, $f(n) = n \lg n$.
  - $n^{\log_b a} = n^{\log_2 2} = n$.
  - $f(n) = n \lg n$ is *asympotically* larger than $n^{\log_b a} = n$ but it is not *polynomially* larger.
  - The recurrence falls into the gap between case 2 and case 3.
  - The master method does not apply to the recurrence.

The Master Theorem: Examples

- $T(n) = 2T(n/2) + \Theta(n)$
  - $a = 2$, $b = 2$, $f(n) = \Theta(n)$.
  - $n^{\log_b a} = n^{\log_2 2} = n = \Theta(n)$.
  - $f(n) = \Theta(n^{\log_2 2}) = \Theta(n)$.
  - Case 2: $T(n) = \Theta(n \lg n)$
- $T(n) = 8T(n/2) + \Theta(n^2)$
  - $a = 8$, $b = 2$, $f(n) = \Theta(n^2)$.
  - $n^{\log_b a} = n^{\log_2 8} = n^3$.
  - $f(n) = \Theta(n^{\log_2 8 - \varepsilon})$, where $\varepsilon = 1$.
  - Case 1: $T(n) = \Theta(n^3)$
- $T(n) = 7T(n/2) + \Theta(n^2)$
  - $a = 7$, $b = 2$, $f(n) = \Theta(n^2)$.
  - $n^{\log_b a} = n^{\log_2 7} \approx n^{2.8}$.
  - $f(n) = \Theta(n^{\log_2 7 - \varepsilon})$, where $\varepsilon \approx 0.8$.
  - Case 1: $T(n) = \Theta(n^{\lg 7})$

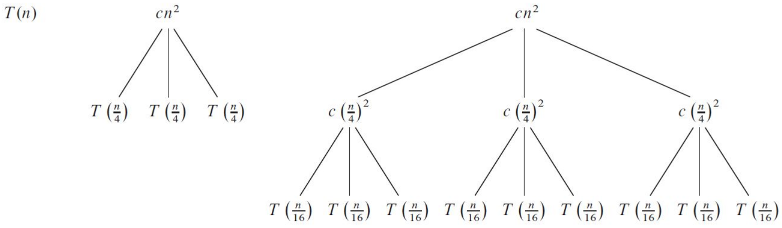
Recursion Tree

- In a *recursion tree*, each node represents the cost of a single subproblem somewhere in the set of recursive function invocations.
- A recursion tree is best used to generate a good guess, which we can then verify by the *substitution method*.
- When using a recursion tree to generate a good guess, we can often tolerate a small amount of "sloppiness," since we verify our guess later on.
- If we are very careful when drawing out a recursion tree and summing the costs, we can use a recursion tree as a direct proof of a solution to a recurrence.

Recursion Tree: Examples

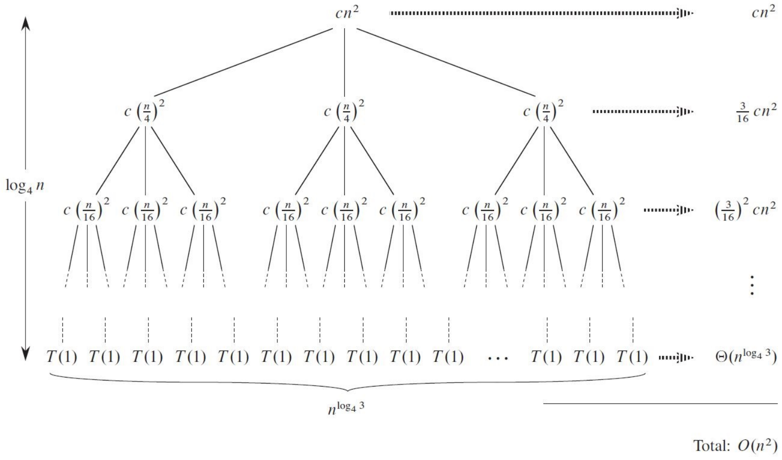
$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- To find an upper bound, we create a recurrence tree for $T(n) = 3T(n/4) + \Theta(n^2)$ (this is an example of sloppyness).
- For convenience, we assume that n is an exact power of 4 (another example of tolerable sloppiness) so that all subproblem sizes are integers.



Recursion Tree: Examples

Recurrence Tree for $T(n) = 3T(n/4) + \Theta(n^2)$



Recursion Tree: Examples

$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- How far from the root do we reach one?
  - The subproblem size for a node at depth i is $n/4^i$.
  - The subproblem size hits $n = 1$ when $n/4^i = 1$, i.e., $i = \log_4 n$.
  - The tree has $\log_4 n + 1$ levels (at depths $0, 1, 2, \ldots, \log_4 n$).
- We determine the cost at each level of the tree.
  - The number of nodes at depth i is 3^i.
  - Each node at depth i has a cost of $c(n/4^i)^2$.
  - The total cost over all nodes at depth i is $3^i c(n/4^i)^2 = (3/16)^i cn^2$.
  - The bottom level, at depth $\log_4 n$, has $3^{\log_4 n} = n^{\log_4 3}$ nodes, each contributing cost $T(1)$ for a total cost of $n^{\log_4 3} T(1) = \Theta(n^{\log_4 3})$.

Recursion Tree: Examples

$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- The cost of the entire tree:

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})
\end{aligned}
$$

Recursion Tree: Examples

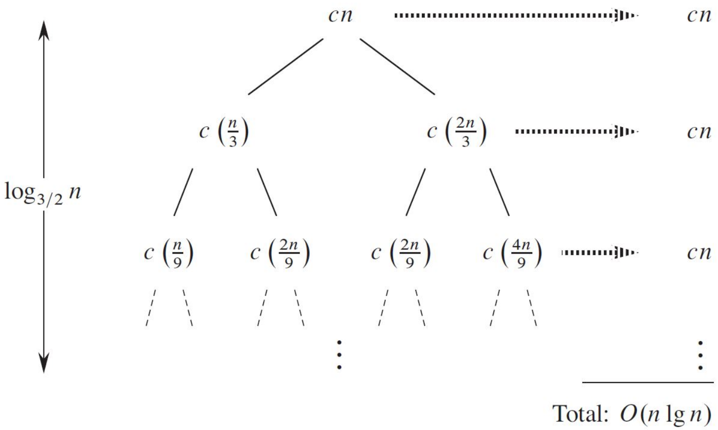
$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- We take advantage of small amounts of "sloppiness" and use an infinite decreasing geometric series as an upper bound.

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2) .
\end{aligned}
$$

Recursion Tree: Examples

Recurrence Tree for $T(n) = T(n/3) + T(2n/3) + O(n)$



Recursion Tree: Examples

Recurrence Tree for $T(n) = T(n/3) + T(2n/3) + O(n)$

- The longest simple path from the root to a leaf is $n \to (2/3)n \to (2/3)^2 n \to \ldots \to 1$.
- Since $(2/3)^k n = 1$ when $k = \log_{3/2} n$, the height of the tree is $\log_{3/2} n$.
- The solution to the recurrence is at most the number of levels times the cost of each level, or $O(cn \log_{3/2} n) = O(n \lg n)$.

Substitution Method

Steps of Substitution Method

- Guess the form of the solution (better to use a recursion tree for this).
- Use mathematical induction to find the constants and show that the solution works.

Substitution Method: Examples

Revisiting $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

- We have guessed (using recursion tree) $T(n) = O(n^2)$.
- We want to show that $T(n) \le dn^2$ for some constant $d > 0$. Using the same constant $c > 0$ as before, we have

$$\begin{aligned}
T(n) &\le 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\le 3d\,\lfloor n/4 \rfloor^2 + cn^2 \\
&\le 3d(n/4)^2 + cn^2 \\
&= \frac{3}{16}dn^2 + cn^2 \\
&\le dn^2\,,
\end{aligned}$$

where the last step holds as long as $d \ge (16/13)c$.

Substitution Method: Examples

Revisiting $T(n) = T(n/3) + T(2n/3) + O(n)$

- We have guessed (using recursion tree) $T(n) = O(n \lg n)$.
- We want to show that $T(n) \leq dn \lg n$ for some constant $d > 0$. Using the same constant $c > 0$ as before, we have

$$
\begin{aligned}
T(n) &\leq T(n/3) + T(2n/3) + cn \\
&\leq d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) \\
&\qquad + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\leq dn\lg n\,,
\end{aligned}
$$

where the last step holds as long as $d \geq c/(\lg 3 - (2/3))$.

White Board

White Board

White Board