CSE 101 Class Notes

Divide & Conquer: Integer Multiplication, Master Theorem

October 25, 2006

Multiplication: First Try

We want the product of two numbers $x = \sum_{i=0}^{n-1} x_i 10^i, y = \sum_j y_{j=0}^{n-1} 10^j$. The grade-school method is $O(n^2)$, but we can do a faster version using divide-and-conquer.

As an example, let $x = 1234, y = 5678$. Then

$$\begin{aligned}
1234 \times 5678 &= (12 \times 100 + 34) \times (56 \times 100 + 78) \\
&= (12 \times 56)10^4 + (56 \times 34 + 12 \times 78)10^2 + 34 \times 78) \\
&= ... = 7066652
\end{aligned}$$

In general,

$$xy = 10^n x_h y_h + 10^{n/2}(x_h y_l + x_l y_h) + x_l y_l$$

or in pseudocode:

```
1     mult(x[n-1..0],y[n-1..0])
2         if n = 1
3             return lookup(x,y)
4         else
5             xh <- x[n-1..n/2]; xl <- x[n/2..0]
6             yh <- y[n-1..n/2]; yl <- y[n/2..0]
7             a  <- xh * yh
8             b  <- xh * yl
9             c  <- xl * yh
10            d  <- xl * yl
11            bc <- add(b, c)
12            a  <- append_zeros(a, 2*floor(n/2))
13            bc <- append_zeros(bc, floor(n/2))
14            return add(a, bc, d)
```

Lines 5-6 take $O(n)$; lines 7-10 take $T(n/2)$ each; lines 11-13 take $O(n)$; so the total running time is

$$T(n) = 4T(n/2) + O(n)$$

Running Time

Depth	Subproblems	Size	Subprobem Time	Total time
0	1	n	cn	cn
1	4	$n/2$	$cn/2$	$4cn/2 = 2cn$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
i	4^i	$n/2^i$	$cn/2^i$	$4^i cn/2^i = 2^i cn$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\log n$	$4^{\log n} = n^2$	1	c	cn^2
TOTAL				$\sum_{i=0}^{\log n} 2^i cn = cn \sum_i 2^i$ $= cn\left(2^{\log n+1} - 1\right)$ $= \Theta(n^2)$

So this is no better than grade-school multiplication.

Second (Clever) Try

As it turns out, we can do better than this by rearranging the work we do in lines 7-10 to reduce the number of subproblems to solve. We need to compute the following expression:

$$(x_l y_l)10^n + (x_l y_h + x_h y_l)10^{n/2} + (x_h y_h)$$

We can compute the three subterms using only *three* multiplications by recognizing that

$$(x_l y_h + x_h y_l) = (x_l + x_h)(y_l + y_h) - (x_l y_l) - (x_h y_h)$$

We already have to compute the last two terms, so by computing $(x_l + x_h)(y_l + y_h)$, we can find the $10^{n/2}$-term in one less (recursive) multiplication, by subtracting out the last two. Our table then becomes:

Depth	Subproblems	Size	Subprobem Time	Total time
0	1	n	cn	cn
1	3	$n/2$	$cn/2$	$4cn/2 = 2cn$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
i	3^i	$n/2^i$	$cn/2^i$	$3^i cn/2^i = 1.5^i cn$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\log n$	$3^{\log n} = n^{\log 3}$	1	c	$cn^{\log 3}$
TOTAL				$\sum_{i=0}^{\log n} 1.5^i cn = cn \sum_i 1.5^i$ $= cn\frac{1.5^{\log n+1}-1}{1.5-1} \approx cn\frac{3^{\log n}}{2^{\log n}}$ $= \Theta(3^{\log n}) = \Theta(n^{\log 3})$

To get an idea of how much an improvement that is, $\frac{n^2}{n^{\log 3}} = n^{2-\log 3}$ is 2.6 for $n = 10$, 17 for $n = 1000$, and 309 for $n = 10^6$.

Master Theorem

So far, to find the running time of divide-and-conquer algorithms, we have filled in a table like the one above. However, if we could fill in this table once for a "generic" divide-and-conquer algorithm, we could simply read off the answer in the future. Let's.

Depth	Subproblems	Size	Subprobem Time	Total time
0	1	n	cn^k	cn^k
1	a	n/b	$c(n/b)^k$	$ac(n/b)^k$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
i	a^i	n/b^i	$c(n/b^i)^k$	$a^i c(n/b^i)^k$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\log_b n$	$a^{\log_b n} = n^{\log_b a}$	1	c	$cn^{\log_b a}$

The total time falls into one of three cases:

- If the algorithm is top-heavy (i.e. $a/b^k < 1$), then the top line of the table dominates, and $T(n) = O(n^k)$.
- If the algorithm is balanced (i.e. $a/b^k = 1$), then the total time is the sum of the times at each level, or $T(n) = \log_b n O(n^k) = O(n^k \log_b n)$
- If the algorithm is bottom-heavy (i.e. $a/b^k > 1$), then the table's height (hence the number of base cases) dominates and $T(n) = O(n^{\log_b a})$.