Quiz!

X-tra: Show $x^2$ is not $O(n)$.

Announcements

Announcements:
- HW submitted through Canvas (pdf)
  - No late accepted
  - Submit early to test
  - New students

- Scanners
  - Type
Q: Create a Divide & Conquer Algorithm for Multiplication
(a, b are digit numbers, n is a power of 2)

Base Case: If 1 digit numbers, can do in $O(1)$ time

\[ a = (a_{n-1} \ldots a_0) \cdot 10^{n/2} + (a_{n-1} \ldots a_0) \]
\[ b = (b_{n-1} \ldots b_0) \cdot 10^{n/2} + (b_{n-1} \ldots b_0) \]

\[ a \cdot b = (a' \cdot 10^{n/2} + a'') \cdot (b' \cdot 10^{n/2} + b'') \]
\[ = a' \cdot b' \cdot 10^n + (a'' + a' b'') \cdot 10^{n/2} + a'' b'' \]

Conquer: Solve smaller problems: $a' b'$, $a'' b'$, $a' b''$, $a'' b''$

Combine: Add $a' b' \cdot 10^n + (a'' b' + a' b'') \cdot 10^{n/2} + a'' b''$

Use for 100 ops $O(n)$
Doing Better: "Gauss's Trick"

Before: \(a' b' \ 10^n + (a' b'' + a'' b') 10^n/2 + a'' b''\)

Notice \(a' b'' + a'' b' = (a' + a'')(b' + b'') - a' b' - a'' b''\)

2 multiplications 1 new multiplication of \(\frac{n}{2} + 1\) digit \#s

already perform these

If interested, shows \(\frac{n}{2} + 1\) digits as hard as \(\frac{n}{2}\)

Problem: Assume only multiplying \(2^k\)-digit numbers, but

if \(n = 2^k\), \(\frac{n+1}{2} \neq 2^k\)

Instead: If \(d, e\) \(\frac{n+1}{2}\) digit numbers, do

\[d \cdot e = \left(\begin{array}{c} d_{n/2} \\ \vdots \\ d_0 \end{array}\right) \left(\begin{array}{c} e_{n/2} \\ \vdots \\ e_0 \end{array}\right) + d_{n/2} \cdot e_{n/2} \cdot 10^n + d_{n/2} \cdot e_{n/2} \cdot 10^{n/2}\]

Multiplying \(2 \frac{n}{2}\)-digit numbers + \(O(n)\) multiplications

Base problem \(n \times n\)

\[\frac{n}{2} \times \frac{n}{2}\]
Q: How many base problems are implemented?

A: $n \log_3 3^2$  
B: $n \log_2 3$  
C: $n^2 \log_2 3$  
D: $n^2$

How many base problems?

$3 \cdot 3 \cdot 3 \ldots 3$ 

$\log_2 n$ times

$= 3 \log_2 n = \left(2^{\log_2 3}\right) \log_2 n = \left(2^{\log_2 n}\right) \log_2 3$

$= n \log_2 3$

$\leq n^{1.59}$

Given a proposed algorithm, always need to answer 2 questions:

1. Is the algorithm correct?
2. What is the asymptotic worst-case run-time?

Q. Which proof method should be used to prove correctness of Karatsuba Multiplication algorithm?

A) Proof by Contradiction  
B) Brute-force Search  
C) Proof by Contrapositive  
D) Proof by Induction

To analyze runtime, create "Recurrence Relation".

Let $T(n)$ be # of operations required to solve Karatsuba multiplication of n-digit #s.
Recurrence for Karatsuba

1. Base Case: $T(1) \leq O(1)$ (use lookup table)

2. For $n > 1$:

   $T(n) \leq \frac{3T(n/2)}{2} + \frac{11n + 3}{10}$

   - Divide
   - Recursive calls
   - Size of input to recursive call

   Everything else the algorithm does

   Recursive call initiation

   Dealing with $\frac{n}{2}$+1-digit multiplication

   All the additions/subtractions of $n$-digit numbers

Want $T(n)$ as a function of $n$, not as a function of $T(n/2)$.

2 Approaches

- Master Method
- Guess & Check