COMPSCI 330 Lecture 3 Divide and Conquer (continued)

Sunday, September 4, 2016 8:26 PM

- Integer multiplication
- More recursions
- Master Theorem

- Integer Multiplication
  - Given two integers a, b with $\leq n$ digits, compute $a \times b$
  - basic algorithm: elementary school

$$\begin{array}{r} 12 \\ \times\ 34 \\ \hline 48 \\ 36 \\ \hline 408 \end{array}$$

    $\Theta(n^2)$ time
  - can we do better.
  - divide and conquer

preprocess: make sure a, b have same length by adding 0's to the left of shorter one.

$$a = [a_L][a_R] = a_L \cdot 10^{\frac{n}{2}} + a_R$$

$$b = [b_L][b_R] = b_L \cdot 10^{\frac{n}{2}} + b_R$$

$$a \times b = (a_L \cdot 10^{\frac{n}{2}} + a_R)(b_L \cdot 10^{\frac{n}{2}} + b_R)$$

$$= a_L b_L 10^n + (a_L b_R + a_R b_L) 10^{\frac{n}{2}} + a_R b_R$$

```
multiply(a, b)
  if len(a), len(b) ≤ 1 then return a×b.
  partion a, b into (a_L, a_R) (b_L, b_R)
  return multiply(a_L, b_L) · 10^n + (multiply(a_L, b_R) + multiply(a_R, b_L)) 10^(n/2) + multiply(a_R, b_R)
```

  - running time?

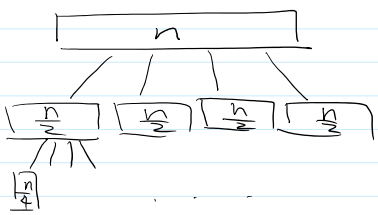
$$T(n) \leq 4 \cdot T\left(\frac{n}{2}\right) + O(n)$$

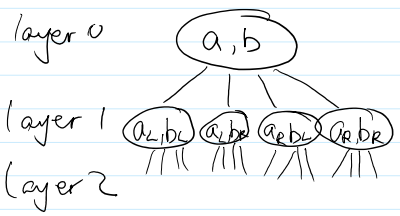
running time at layer $k = 4^k \cdot C\frac{n}{2^k} = C \cdot n 2^k$

$C \cdot n$

$4 \cdot C \cdot \frac{n}{2}$

$4^2 C \cdot \frac{n}{4}$





$$T(n) = \sum_{k=0}^{\log_2 n} C \cdot n 2^k$$

$$= C \cdot n \cdot \sum_{k=0}^{\log_2 n} 2^k$$

$$= C \cdot n \left(2^{\lceil \log_2 n + 1 \rceil} - 1\right) \leq 2C \cdot n^2 = O(n^2)$$

  - idea: reduce the number of subproblems
    - need: $a_L b_L$, $a_L b_R + a_R b_L$, $a_R b_R$

- notice: $a_L b_R + a_R b_L = (a_L + a_R)(b_L + b_R) - a_L b_L - a_R b_R$

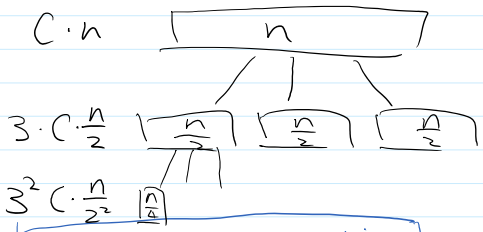
  if let $c_1 = a_L b_L \quad c_2 = a_R b_R \quad c_3 = (a_L + a_R)(b_L + b_R)$

  then can compute all three using +, −

```
multiply(a, b)
    if len(a), len(b) ≤ 1 return a×b
    partition a, b into (a_L, a_R), (b_L, b_R)
    C_1 = multiply(a_L, b_L)
    C_2 = multiply(a_R, b_R)
    C_3 = multiply(a_L + a_R, b_L + b_R)
    return C_1 · 10^n + (C_3 − C_1 − C_2) 10^(n/2) + C_2
```

- running time

$$T(n) \le 3T\left(\frac{n}{2}\right) + O(n)$$



layer k: $3^k \cdot C \cdot \frac{n}{2^k} = C \cdot n \cdot \left(\frac{3}{2}\right)^k$

$$T(n) = \sum_{k=0}^{\log_2 n} C \cdot n \left(\frac{3}{2}\right)^k = C \cdot n \sum_{k=0}^{\log_2 n} \left(\frac{3}{2}\right)^k = C \cdot n \cdot O\left(\left(\frac{3}{2}\right)^{\log_2 n}\right) = O\left(n \cdot n^{\log_2 \frac{3}{2}}\right) = O\left(n^{\log_2 3}\right)$$

$\log_2 3 \approx 1.585$

formula for geometric sum

$$\left(\sum_{i=0}^{t} p^i = \frac{p^{t+1} - 1}{p - 1}\right)$$

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

$$\left(\frac{3}{2}\right)^{\log_2 n} = \left(2^{\log_2 \frac{3}{2}}\right)^{\log_2 n} = 2^{\log_2 \frac{3}{2} \cdot \log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 \frac{3}{2}} = n^{\log_2 \frac{3}{2}} = n^{\log_2 3 - 1}$$

- not the fastest algorithm, Fast Fourier Transform $O(n \log n \log\log n)$
- same idea can be applied to matrix multiplication [Strassen]

- Master Theorem

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \qquad (a, b: \text{constants})$$

- case 1 if $f(n) = O(n^c)$, $c < \log_b a$

  $T(n) = \Theta(n^{\log_b a})$

  bottom layer of recursion tree dominates

- case 2 if $f(n) = \Theta(n^c \log^t n)$, $c = \log_b a$

  $T(n) = \Theta(n^c \log^{t+1} n)$

  all layers are roughly equal

  additional log n factor from depth of the tree

- case 3 if $f(n) = \Omega(n^c)$, $c > \log_b a$

  $T(n) = \Theta(f(n)) = \Theta(n^c)$

  top layer of recursion tree dominates

- Examples: ① merge sort, running time $a = b = 2$, $f(n) = n$ case 2

  $T(n) = \Theta(n \log n)$

  ② mergesort, memory $S(n) = S\left(\frac{n}{2}\right) + O(n)$ $a = 1$, $b = 2$, $f(n) = n$

  $S(n) = \Theta(n)$ case 3

  ③ integer multiplication $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$ $a = 3$, $b = 2$, $c = 1 < \log_b a$

$T(n) = \Theta(n^{\log_2 3})$ case 1

- Proof (sketch) use recursion tree

$\log_b n$ layers

layer k $a^k f\left(\frac{n}{b^k}\right)$

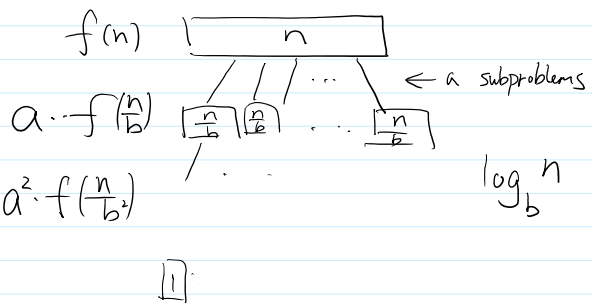
if $f(n) = n^c$

$\rightarrow a^k \left(\frac{n}{b^k}\right)^c$

$= n^c \left(\frac{a^k}{b^c}\right)^k$

$T(n) = \sum_{k=0}^{} n^c \left(\frac{a}{b^c}\right)^k$

case 1 $c < \log_b a$ $\quad b^c < b^{\log_b a} = a$



- rely on recursion tree / induction

$T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$

layer k $C \cdot n \left(\frac{9}{10}\right)^k$

$T(n) = \sum_{k=0}^{\infty} C \cdot n \left(\frac{9}{10}\right)^k$

$\le \sum_{k=0}^{\infty} C \cdot n \left(\frac{9}{10}\right)^k$

$= O(n)$