Office hour feedback on test 2-3 in Help Centre BA 2230

CSC236 fall 2012

more complexity: mergesort

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/236/F12/
416-978-5899

Using Introduction to the Theory of Computation, Chapter 3

Outline

divide and conquer (recombine)

using the Master Theorem

Notes

General case

revisit...

Class of algorithms: partition problem into b *roughly* equal subproblems, solve, and recombine:

threshold

$$T(n) = \begin{cases} k & \text{if } n \leq B \\ a_1\,T(\lceil n/b \rceil) + a_2\,T(\lfloor n/b \rfloor) + \underline{f(n)} & \text{if } n > B \end{cases}$$

where $B, k > 0$, $a_1, a_2 \geq 0$, and $\underbrace{a_1 + a_2}_{a=2} > 0$. $f(n)$ is the cost of splitting and recombining.

$a = 2$

b

$\theta(n^d)$

Master Theorem

(for divide-and-conquer recurrences)

$$\sum \frac{a}{b^d}$$

$\frac{a}{b^d} < 1$

$\frac{a}{b^d} = 1$

$\frac{a}{b^d} > 1$

If f from the previous slide has $f \in \theta(n^d)$, then

$$T(n) = \begin{cases} \theta(n^d) & \text{if } a < b^d \\ \theta(n^d \log n) & \text{if } a = b^d \\ \theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

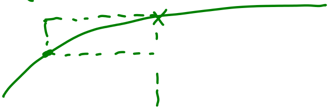
$$\theta\left(n^{\log_2 4}\right) = \theta\left(n^2\right)$$

Proof sketch

done twice in tutorial

1. Unwind the recurrence, and prove a result for $n = b^k$

2. Prove that T is non-decreasing (exercise or 2).



3. Extend to all n, similar to MergeSort

multiply lots of bits

what if they don't fit into a machine instruction?

multiplication of n-bit numbers.

try this in java, another language.

fixed length (e.g. 32-bits, 64-bits).

$$
\begin{array}{r}
1101 \\
\times 1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
10001111
\end{array}
$$

n (4) copies of n-bit numbers

n sums of n-bit numbers.

$\sim n^2$

$\Theta(n^2)$

divide and recombine

recursively...

$n = 4$ (!)

$x_1 \times 2^2 = 1100$

$11 = 3$

$110 = 6$

$1100 = 12$

n-bit x and y.

x_1	x_0
11	01
$\times$10	11
y_1	y_0

$$xy = 2^n x_1 y_1 + 2^{n/2}(x_1 y_0 + y_1 x_0) + x_0 y_0$$

$$= 2^n(10 \times 11) +$$

compare costs

$\Theta(n^2)$

$T(n) = \begin{cases} k & n < B \\ a_1 T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + a_2 \left(\left\lceil \frac{n}{2} \right\rceil\right) + \Theta \end{cases}$

n n-bit additions versus:

1. divide each factor (roughly) in half
2. multiply the halves (recursively, if they're too big)
3. combine the products with shifts and adds

$\Theta(n)$

$a = 4$ $\qquad a > b^d$

$b = 2$

$d = 1$ $\qquad \Theta(n^{\log_b a})$

Gauss's trick

$$xy = 2^n x_1 y_1 + 2^{n/2}(x_1 y_0 + x_0 y_1)$$

$x_0 y_0$

$$xy = 2^n x_1 y_1 + x_0 y_0 + 2^{n/2}\left((x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0\right)$$

reduce a from $4 \to 3$.

Gauss's payoff

lose one multiplication

$\Theta(n^2)$

$$T(n) = \begin{cases} k & n < B \\ a_1 T(\lceil n/2 \rceil) + a_2 T(\lfloor n/2 \rfloor) + \Theta(n). \end{cases}$$

do better with FFT

1. divide each factor (roughly) in half
2. sum the halves
3. multiply the sum and the halves Gauss-wise
4. combine the products with shifts and adds

$\lfloor n/2 \rfloor + 1$ $(x_1 + x_0)$ $\lceil n/2 \rceil + 1$ $(y_1 + y_0)$

$\Theta(n)$

$b = 2$

$d = 1$

$a = 3$

$a > b^d$

$\Theta(n^{\log_2 3}) \approx \Theta(n^{1.5\ldots})$

Computer Science
UNIVERSITY OF TORONTO

Notes

n dots

- find closest pair,

$\sim \underline{\underline{n^2}}$

$\binom{n}{2}$ pairs to examine