CS202 Fall 2012
Lecture 16 – 10/25

Divide-and-conquer

Prof. Tanya Berger-Wolf
http://www.cs.uic.edu/bin/view/CS202/WebHome

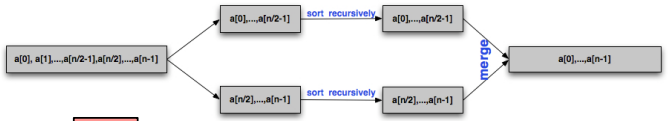
Recurrences

Solve $a_n = 2a_{n/2} + n-1$, $a_1 = 0$.

Mergesort
# of comparisons

Motivating Example: MergeSort

Sorting numbers a[0], **a[1], a[2], …, a[n-2], a[n-1]:**



T(n)

Running time on n numbers =
Split time + 2*(Running time on n/2 numbers) + Merge time of 2*(n/2)

Constant O(1)

T(n/2)

Exactly (we can prove) n

$T(n) = O(1) + 2T(n/2) + n = 2T(n/2) + n$

Divide and Conquer Recurrences

General form: T(n) = aT(n/b) + f(n)

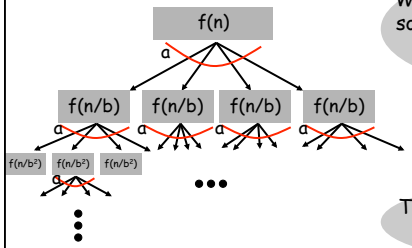
What do the algorithms look like?
Divide the problem into a subproblems of size n/b.
Solve those subproblems (recursively).
Conquer the solution in time f(n).

We understand how abstract this is. It gets better

Examples: mergesort and binary search.

Divide and Conquer Recurrences

General form: T(n) = aT(n/b) + f(n)
To solve a problem of size n, we require time f(n), plus the time it takes to solve a subproblems of size n/b.

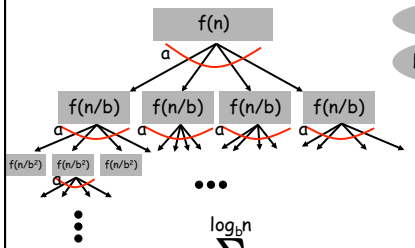


We don't have simple recipes for solving these in all cases, though sometimes we do…

Total running time is sum of the values in the grey rectangles.

Divide and Conquer Recurrences

General form: T(n) = aT(n/b) + f(n)



Sum over levels…

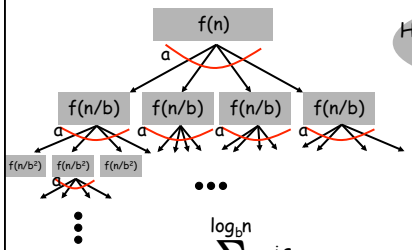
How many?

$\log_b n$

$$\sum_{i=0}^{\log_b n}$$

Divide and Conquer Recurrences

General form: T(n) = aT(n/b) + f(n)



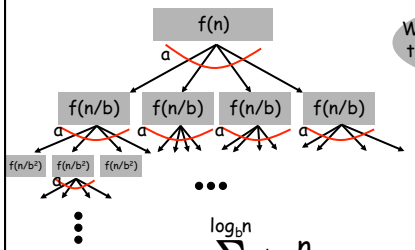
How many blocks at level i?

a^i

$$\sum_{i=0}^{\log_b n} a^i f(?)$$

Divide and Conquer Recurrences

General form: T(n) = aT(n/b) + f(n)



What is the work for the recursive call at level i?

n/b^i

$$\sum_{i=0}^{\log_b n} a^i f\left(\frac{n}{b^i}\right)$$

Announcements

Advising signup:
- Sign up this week (Oct 18-22)
- Advising next week (Oct 25-29)

Project 2 is due in a week

HW 3 is out, due Oct 29

Illinois Technology Challenge, Oct 27, 4-6:30pm, SES RM 130
Bring resume. Algorithm and a quiz. Top students invited for the final round ($5000 prize and a job)
Dorothy Radke at dradke@illinoistech.org

Divide and Conquer Recurrences

General form: $T(n) = aT(n/b) + f(n)$

$$\sum_{i=0}^{\log_b n} a^i f(\frac{n}{b^i})$$

We no longer have recursive terms, but we do have a sum to deal with.

How about our old favorite merge sort?
$T(n) = 2T(n/2) + n$

$a = 2, b = 2, f(n) = n.$

$$\sum_{i=0}^{\log_2 n} 2^i \cdot (n/2^i) = n \log_2 n + n$$

Divide and Conquer Recurrences

General form: $T(n) = aT(n/b) + f(n)$

$$\sum_{i=0}^{\log_b n} a^i f(\frac{n}{b^i})$$

We no longer have recursive terms, but we do have a sum to deal with.

Consider binary search, and write a recurrence for the # of comparisons:
$T(n) = T(n/2) + 1$

$a = 1, b = 2, f(n) = 1.$

$$\sum_{i=0}^{\log_2 n} 1^i \cdot 1 = \log_2 n + 1$$

CS202 Fall 2012
Lecture 17 – 10/30

Divide-and-conquer

Prof. Tanya Berger-Wolf
http://www.cs.uic.edu/bin/view/CS202/WebHome

Example: Exponentiation

Iterative: $x^n = x \cdot x \cdot x \cdot x \cdot \ldots \cdot x$
n-1 operations

Recursive:

$$x^n = \begin{cases} x^{\frac{n}{2}} \cdot x^{\frac{n}{2}} & \text{if } n \text{ even} \\ x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x & \text{if } n \text{ odd} \end{cases}$$

return pow(x, n/2) * pow (x, n/2);

T(n) = 2T(n/2) + 2
a = b = 2, f(n) = 2

$$\sum_{i=0}^{\log_2 n} a^i f(\frac{n}{b^i}) = \sum_{i=0}^{\log_2 n} 2^i f(\frac{n}{2^i}) = 2 \sum_{i=0}^{\log_2 n} 2^i = 2(2^{\log_2 n + 1} - 1) = 4n - 2$$

a = pow(x, n/2);
return (a*a);

T(n) = T(n/2) + 2
a = 1, b = 2, f(n) = 2 then T(n) = O(log n)

13

Master Theorem

The recurrence T(n) = aT(n/b) + f(n) can be solved as follows:
If $a \cdot f(n/b) = kf(n)$ for some constant $k < 1$, then $T(n) = \Theta(f(n))$
If $a \cdot f(n/b) = Kf(n)$ for some constant $K > 1$, then $T(n) = \Theta(n^{\log_b a})$
If $a \cdot f(n/b) = f(n)$, then $T(n) = \Theta(f(n)\log_b n)$

You should check that this works for the recurrences we've seen here.