Recurrences

Divide & conquer

Given problem instance of size n

$\Theta(1)$ if n small – solve directly

if n large

$f(n)$ – divide into b subproblems of size n/a

– solve each

– combine to solve full problem.

$$T(n) = \begin{cases} \Theta(1) & n - \text{small} \\ bT(n/a) + f(n) & n \text{ large.} \end{cases}$$

eg T cubic, $a = 2$

$T(n/2) \leq \frac{1}{8} T(n)$

afford 7 subproblems ~~and~~ still be faster

[if $f(n)$ not bad]

$a = b = 2$

if T linear

$T(n/2) \approx \frac{1}{2} T(n)$

if T super linear

$T(n/2) \ll \frac{1}{2} T(n)$

Example : Merge Sort

Problem: Sort $A[1], A[2], \ldots, A[n]$

Method:

MSort (A, p, q) : Sort $A[p] \cdots A[q]$

if $(p \geq q)$ return;

MSORT$(A, p, \lfloor \frac{p+q}{2} \rfloor)$

MSORT$(A, \lfloor \frac{p+q}{2} \rfloor + 1, q)$

MERGE(A, p, q)

Analysis:

$n = 2^k$

count only comparisons

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T(n/2) + \underbrace{n-1}_{\text{merge}} & n > 1 \end{cases}$$

Solving Recurrences: Substitution

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T(n/2) + n - 1 & n > 1 \end{cases}$$

claim: $T(n) = n \lg n - n + 1$

Proof (induction on n):

basis ($n = 1$):

$1 \cdot \lg 1 + 1 - 1 = 0$

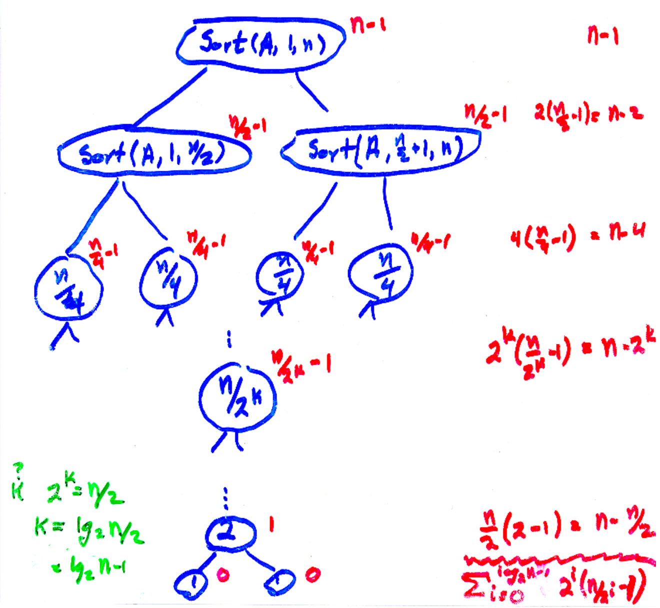
induction ($n > 1$):

Assume $T(j) = j \lg j - j + 1$ for $j < n$

$$\begin{aligned} T(n) &= 2T(n/2) + n - 1 \\ &= 2\left(\tfrac{n}{2} \lg \tfrac{n}{2} - \tfrac{n}{2} + 1\right) + n - 1 \\ &= \left(n \lg \tfrac{n}{2} - n + 2\right) + n - 1 \\ &= n \lg \tfrac{n}{2} + 1 \\ &= n(\lg n - 1) + 1 = n \lg n - n + 1 \end{aligned}$$

Solving Recurrences: Recursion Tree

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T(n/2) + n - 1 & n > 1 \end{cases}$$



Recursion Tree (cont.)

$$\sum_{i=0}^{\lg_2 n-1} 2^i (n/2^i - 1)$$

$$= \sum_{i=0}^{\lg n-1} (n - 2^i)$$

$$= n \lg n - \sum_{i=0}^{\lg n-1} 2^i$$

$$= n \lg n - \frac{2^{\lg n} - 1}{2-1}$$

$$= n \lg n - n + 1$$

$2^{\lg n-1} \ldots 2^2\, 2^1\, 2^0$

1 1 1 1 1

1 0 0 0 0 0 0

$2^{\lg n} = n$

$$\sum_{i=0}^{k} x^i = \frac{x^{k+1}-1}{x-1}$$

Iteration

$$T(n) = \begin{cases} 1 & n=1 \\ 7T(n/2) + cn^2 & n>1 \end{cases}$$

$$T(n) = cn^2 + 7T(n/2)$$

$$= cn^2 + 7\left(c\left(\tfrac{n}{2}\right)^2 + 7T\left(\tfrac{n}{4}\right)\right)$$

$$= cn^2\left(1 + \tfrac{7}{4}\right) + 7^2 T\left(\tfrac{n}{2^2}\right)$$

$$= cn^2\left(1 + \tfrac{7}{4}\right) + 7^2\left(c\left(\tfrac{n}{2^2}\right)^2 + 7T\left(\tfrac{n}{2^3}\right)\right)$$

$$= cn^2\left(1 + \tfrac{7}{4} + \left(\tfrac{7}{4}\right)^2\right) + 7^3 T\left(\tfrac{n}{2^3}\right)$$

$$= cn^2 \sum_{i=0}^{k-1} \left(\tfrac{7}{4}\right)^i + 7^k T\left(\tfrac{n}{2^k}\right)$$

$$= cn^2 \sum_{i=0}^{\lg n-1} \left(\tfrac{7}{4}\right)^i + 7^{\lg_2 n} T(1)$$

$7^{\lg n} = n^{\lg 7}$

$a^{\log_b n} = n^{\log_b a}$

$$O(n^{\lg_2 7}) = O(n^{2.81})$$

Theorem 4.1 (Master theorem)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

A Pitfall

Claim: $\sum_{k=1}^{n} k = O(n)$

"Proof":

Basis: $\sum_{k=1}^{1} k = 1 = O(1)$

Ind!: $\sum_{k=1}^{n+1} k = \sum_{k=1}^{n} k + (n+1)$
$= O(n) + (n+1)$
$= O(n+1)$

FALSE!

Right approach:
prove $\exists$ a single constant c s.t.
$\sum_{k=1}^{n} k \leq cn$
(will fail:
$\sum_{k=1}^{n+1} k = \sum_{k=1}^{n} k + (n+1)$
$\leq cn + n+1 = (c+1)n + 1$