T1: Friday :C
A1: Thurs?

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

vexing complexity

mergesort

Divide-and-conquer

Notes

Upper bound on $T(n)$

trouble!

last time + tutorial

$T(n) \leq c \lg n$, if $c \geq 1$

try prove this in general.

$$T(n) = \begin{cases} 1 & n = 1 \\ 1 + T(\lceil n/2 \rceil) \end{cases}$$

Assume $n \in \mathbb{N}^{\geq 1}$ and $\forall i \ 1 \leq i < n$, $T(i) \leq c \lg i$.

Then $T(n) = 1 + T(\lceil n/2 \rceil)$

$\leq 1 + c \lg(\lceil n/2 \rceil)$

$\leq 1 + c \lg\left(\frac{n+1}{2}\right)$ or

$1 + c(\lg(n+1) - 1)$

(try pursuing).

$1 \leq \lceil n/2 \rceil < n$

try odd even

$\leq 1 + c \lg\left(\frac{n}{2} + 1\right)$

$1 + c$?

try

$T(n) \leq c \lg(n-1)$

since $\lceil n/2 \rceil - 1 \leq \frac{n+1}{2} - 1 = \frac{n-1}{2}$

recurrence for MergeSort

$T(n)$.

$m - b + 1 = \lceil n/2 \rceil$

$e - m = \lfloor n/2 \rfloor$

```
MergeSort(A,b,e):
    if b == e:  return
    m = (b + e) / 2
    MergeSort(A,b,m)
    MergeSort(A,m+1,e)
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
    for i = b,...,e:  B[c] = A[c]
    c = b
    d = m+1
    for i = b,...,e:
        if d > e or (c <= m and B[c] < B[d]):
            A[i] = B[c]
            c = c + 1
        else:  # d <= e and (c > m or B[c] >= B[d])
            A[i] = B[d]
            d = d + 1
```

constant, constant

$T(\lceil n/2 \rceil)$

$T(\lfloor n/2 \rfloor)$

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n + 1 \end{cases}$$

const $\cdot$ n —

constant.

n times constant —

constant

Unwind (repeated substitution)

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

rewrite $T(2^k) = 2T\left(\left\lceil \frac{2^k}{2} \right\rceil\right) + 2^k + 1$

$= 2T(2^{k-1}) + 2^k + 1$

$= 2(2T(2^{k-2}) + 2^{k-1} + 1) + 2^k + 1$

$= 2^2 T(2^{k-2}) + 2^k + 2 + 2^k + 1$

$= 2^2 T(2^{k-2}) + 2 \cdot 2^k + 3$

$= 2^2(2T(2^{k-3}) + 2^{k-2} + 1) + 2 \cdot 2^k + 3$

$= 2^3 T(2^{k-3}) + 3 \cdot 2^k + 7$

$\vdots$

$= 2^i T(2^{k-i}) + i2^k + 2^i - 1$

$\vdots$

$= 2^k T(2^{k-k}) + k2^k + 2^k - 1$

$= n + n\lg n + n - 1 = n\lg n + 2n - 1$

$\Theta(n \lg n)$

Prove $\Omega(n \lg n)$ for general case

$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n + 1$

Prove $\Omega(n \lg n)$ for general case

$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n + 1$

General case

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

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

where $B, k > 0$, $a_1, a_2 \geq 0$, and $a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.

Master Theorem

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

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

Notes

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

Prove $T(n) \le c \lg(n-1)$. **Want** $T(i) \le c \lg(i-1), \forall i \; 3 \le i < n$

Induction

$$\begin{aligned} T(n) &= 1 + T(\lceil n/2 \rceil) && \# \text{ IH} \\ &\le 1 + c \lg(\lceil n/2 \rceil - 1) \\ &\le 1 + c \lg\left(\tfrac{n+1}{2} - 1\right) && \# \lg \uparrow \\ &= 1 + c \lg\left(\tfrac{n-1}{2}\right) \\ &= 1 + c\left(\lg(n-1) - \lg 2\right) \\ &= 1 - c + c \lg(n-1) && \# c \ge 1 \end{aligned}$$

if $n > 3$, $\lceil \frac{n}{2} \rceil < n$ is $\lceil \frac{n}{2} \rceil \ge 3$? true $n \ge 5$

Base cases — Break point > 0

$T(1) = 1 \le c \lg(1-1)$

$T(2) = 1 + T(\lceil 2/2 \rceil) = 2 \overset{?}{\le} c \lg(2-1)$

$T(3) = 1 + T(\lceil 3/2 \rceil) = 1 + 2 = 3 \overset{?}{\le} c \lg(3-1)$

leave as exercise

try $3 = c$