A1 - RSN
T1 - 70s

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

Upper bound on $T(n) =$

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

trouble! $T(n) \geq c \lg n \leq c \lg(n-1)$

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

rough work $T(n) \leq c \lg n$

assume $n \in \mathbb{N}$, $n > 1$, and assume $T(i) \leq c \lg i$ $1 \leq i < n$

Then $T(n) = 1 + T(\lceil n/2 \rceil)$ # by def.

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

$\overset{?}{\leq} 1 + c \lg(\frac{n+1}{2})$ — or $1 + c \lg(\frac{n}{2} + 1)$

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

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

won't work (easily)

recurrence for MergeSort

$T(n)$ $\quad n = e - b + 1$

```
MergeSort(A,b,e):
    if b == e:  return                          constant
    m = (b + e) / 2                             constant
    MergeSort(A,b,m)                            T(⌈n/2⌉)
    MergeSort(A,m+1,e)                          T(⌊n/2⌋)
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
    for i = b,...,e:  B[i] = A[i]               n c
    c = b                                       constant
    d = m+1                                     constant
    for i = b,...,e:                            n c2
        if d > e or (c <= m and B[c] < B[d]):   constant
            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
```

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

Unwind (repeated substitution)

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

$n = 2^k$ → rewrite as $T(2^k) = 2T(2^{k-1}) + 2^k + 1$

Valid if $\lceil n/2 \rceil = \lfloor n/2 \rfloor$

$2^k = n$

$k = \lg n$

$$
\begin{aligned}
T(2^k) &= 2^1 T(2^{k-1}) + 2^k + 1 \\
&= 2\left(2T(2^{k-2}) + 2^{k-1} + 1\right) + 2^k + 1 \\
&= 2^2 T(2^{k-2}) + 2^k + 2 + 2^k + 1 \\
&= 2^2 T(2^{k-2}) + 2 \cdot 2^k + 3 \\
&\vdots \\
&= 2^i T(2^{k-i}) + i2^k + 2^i - 1 \\
&\vdots \\
&= 2^k T(2^{k-k}) + k2^k + 2^k - 1 \\
&= n \cdot 1 + n \lg n + n - 1 \\
&= n \lg n + 2n - 1
\end{aligned}
$$

Prove that T is non-decreasing

Lemma in ch 3

$T(n) \leq cn \lg n$?

See Course Notes, Lemma 3.6 Exercise: Prove the recurrence for binary search is non-decreasing

$\hat{n} = 2^{\lceil \lg n \rceil}$

$\frac{\hat{n}}{2} < n \leq \hat{n}$

$2n \geq \hat{n}$

$T(n) \leq Bg\uparrow$

$T(\hat{n}) = \hat{n} \lg \hat{n} + 2\hat{n} - 1$

$\leq 2n \lg(2n) + 4n - 1$

$6n - 1$

$cn \lg n \geq 6n - 1$

Prove $T \in O(n \lg n)$ for general case

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

given $T(n) = n \lg n + 2n - 1$ if $n = 2^k$, $k \in \mathbb{N}$.

get $\hat{n} = 2^{\lceil \lg n \rceil}$, and note that $\hat{n} \le 2n$

$T(n) \le T(\hat{n})$ # $T \uparrow$

$= \hat{n} \lg \hat{n} + 2\hat{n} - 1$ # by proof unwinding

$\le 2n \lg 2n + 4n - 1$ # by $\hat{n} \le 2n$.

$\le 2n(\lg n + 1) + 4n - 1$

$= 2n \lg n + 6n - 1$

$\le 8n \lg n$

$cn \lg n \ge 6n$ (-1)

$c \lg n \ge 6$

$c \ge 6$ works if $n \ge 2$

Thus $T \in O(n \lg n)$ with $c = 8$

General case

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

break apart, then 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

What about merge sort? What is $a = 2$, $b = 2$, $d = 1$? $2 = a = 2^1 = b^d$, $\theta(n^1 \log n)$

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

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

$n^{\log_b a}$

Proof sketch

1. Unwind the recurrence, and prove a result for $n = b^k$
2. Prove that T is non-decreasing
3. Extend to all n, similar to MergeSort

Notes

$T(n) \le c \lg(n-1)$. # we figure out # c on the way!

Scratch

Assume $n \in \mathbb{N}$, $n > 1$, and that $T(i) \le c \lg(i-1)$ $\forall i$ $3 \le i < n$.

Then
$$\begin{aligned} T(n) &= 1 + T(\lceil n/2 \rceil) \\ &\le 1 + c \lg(\lceil n/2 \rceil - 1) && \# \text{ by IH} \\ &\le 1 + c \lg\left(\tfrac{n+1}{2} - 1\right) && \# \ 3 \le \lceil \tfrac{n}{2} \rceil < n \\ &= 1 + c \lg\left(\tfrac{n-1}{2}\right) \\ &= 1 + c(\lg(n-1) - 1) \\ &= c \lg(n-1) + 1 - c \\ &\le c \lg(n-1) && \# \ 1 - c \le 0 \\ & && \# \ 1 \le c \end{aligned}$$

Base case 3/4

Base case

P(0) → false

P(2) → false

P(3)

$1 + T(\lceil 3/2 \rceil)$ → suggest $c \ge 3$

Notes