Analyzing Insertion Sort as a Recursive Algorithm

- Basic idea: divide and conquer
  - » Divide into 2 (or more) subproblems.
  - » Solve each subproblem recursively.
  - » Combine the results.
- Insertion sort is just a bad divide & conquer !
  - » Subproblems: (a) last element
    (b) all the rest
  - » Combine: find where to put the last element

Lecture 2, April 5, 2001

19

Recursion for Insertion Sort

- We get a recursion for the running time T(n):

$$T(n) = \begin{cases} T(n-1)+n & \text{for } n>1 \\ 1 & \text{for } n=1 \end{cases}$$

$$\begin{aligned} T(n) &= T(n-1)+n \\ &= T(n-2)+(n-1)+n \\ &= T(n-3)+(n-2)+(n-1)+n \\ &= \ldots \\ &= \sum_{i=1}^{n} i \\ &= \Theta(n^2) \end{aligned}$$

- Formal proof: by induction.
- Another way of looking: split into n subproblems, merge one by one.

20

Improving the insertion sort

- Simple insertion sort is good only for small n.
- Balance sorting vs. merging: Merge equal size chunks.
- How to merge:

```
i=1, j=1
for k=1 to 2n
   if A(i)<B(j)
        then
          C(k)=A(i)
          i++
        else
          C(k)=B(j)
          j++
end
```

- O(n) time !!

21

Analysis

- Iterative approach:
  - » Merge size-1 chunks into size-2 chunks
  - » Merge size-2 chunks into size-4 chunks
  - » etc.

$$\frac{n}{2}merge(1)+\frac{n}{4}merge(2)+\frac{n}{8}merge(4)+\cdots$$

Overall: $\Theta(n\log n)$

- Intuitively right, but needs proof !

22

Analyzing Recursive Merge-Sort

- **Another approach: recursive.**
  - » **Divide into 2 equal size parts.**
  - » **Sort each part recursively.**
  - » **Merge.**

•Recursion is a way of thinking.

•Easy to design recursive algorithms.

- **We directly get the following recurrence:**

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

- **How to formally solve recurrence ?**
  - » **For example, does it matter that we have Q(n) instead of an exact expression ??**
  - » **Does it matter that we sometimes have n not divisible by 2 ??**

Summations

- **Before dealing with recurrencies, need to read Chapter 3, in particular summations:**

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Harmonic function: $H(n) = \sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1)$

Telescoping series: $\sum_{k=1}^{n-1} \frac{1}{k(k+1)} \quad = \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right)$

$$= \sum_{k=1}^{n-1} \left(\frac{1}{k} \right) - \sum_{k=1}^{n-1} \left(\frac{1}{k+1} \right)$$

$$= \sum_{k=1}^{n-1} \left(\frac{1}{k} \right) - \sum_{k=2}^{n} \left(\frac{1}{k} \right)$$

$$= 1 - \frac{1}{n}$$

More summations

- **Another useful trick:**

$$\sum_{k=0}^{\infty} kx^k = x\frac{d}{dx}\sum_{k=0}^{\infty} x^k = x\frac{d}{dx}\frac{1}{1-x} = \frac{x}{(1-x)^2}$$

- **Summary:**
  - » **Learn to recognize standard simplifications**
  - » **Try going opposite direction**
  - » **If all fails -apply tricks one by one...**

25

Recurrencies

- **Chapter 4 in the textbook.**
- **Algorithm "calls itself" - recursive.**

$$T(n) = \begin{cases} 1 & n=1 \\ T\left(\left\lceil \frac{n}{2} \right\rceil\right)+1 & otherwise \end{cases}$$

- **First, solve for** $n = 2^k$
  - » **Claim:** $T(n) = \lg n + 1$
  - » **Proof by induction:**

$$\begin{aligned} T(1) &= 1 \\ T(2^{k+1}) &= T(2^k)+1 \\ &= \lg(2^k)+1+1 \\ &= k+2 \\ &= \lg(2^{k+1})+1 \quad \text{QED} \end{aligned}$$

26

What if n not a power of 2 ?

- **Easy to prove by induction that** $T(n) \ge T(n-1)$
- **Now we can say:** $T(n) \le T(2^{\lceil \lg n \rceil}) = \lceil \lg n \rceil + 1 = \Theta(\log n)$
- **Observe that we did not prove Theta, only big-Oh !**
- **Technically, we should be careful about floor/ceiling, but usually we can safely concentrate on *n=power of 2*.**

Guessing the solution

- **Instead of adding sequentially, lets divide into 2 parts, add each one recursively, and add the result:**

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

Note that we omit the n=1 case for simplicity

Guess: $T(n) < cn$ for some constant c

$$\text{Then: } T(n) = T\left(\lfloor n/2 \rfloor\right) + T\left(\lceil n/2 \rceil\right) + 1$$
$$< c\left(\lfloor n/2 \rfloor\right) + c\left(\lceil n/2 \rceil\right) + 1$$
$$= cn + 1 \quad \textit{Oopssss....}$$

- **Need a stronger induction hypothesis !**
  Assume: $T(n) < cn - b$ for some constants c,b
  Then: $T(n) = \cdots = cn - 2b + 1 < cn - b$ for $b > 1$

Another example

- **Consider recursion:** $T(n) = 4T\left(\frac{n}{2}\right) + n$
- **First guess:** $T(n) \le cn^3$
- **We omit base case.**
  Induction step: $4T\left(\frac{n}{2}\right) + n \le c\frac{n^3}{2} + n = cn^3 + \underbrace{(n - \frac{c}{2}n^3)}_{rest}$

  for $c \le 2, n \ge 1 \Rightarrow "rest" \le 0$ QED
- **But we can do better !** First try: $T(n) \le cn^2$ is too weak !

  Assume: $T(n) \le c_1 n^2 - c_2 n$

  Then: $T(n) = 4T\left(\frac{n}{2}\right) + n \le 4\left(c_1\left(\frac{n}{2}\right)^2 - c_2\frac{n}{2}\right) + n = c_1 n^2 - 2c_2 n + n$

  $= c_1 n^2 - c_2 n + \underbrace{(n - c_2 n)}_{REST}$

29

Initial Conditions

- **Can initial conditions affect the solution ?** → **YES !**

$$T(n) = \left[T(n/2)^2\right]$$

$$T(1) = 2 \Rightarrow T(n) = 2^n$$
$$T(1) = 3 \Rightarrow T(n) = 3^n$$
$$T(1) = 1 \Rightarrow T(n) = 1$$

- **n was assumed to be a power of 2.**

30

Iterating recurrencies

- **Example:** $T(n) = 4T(n/2) + n$

$$= n + 4(n/2 + 4T(n/4)) = n + 2n + 16T(n/4)$$
$$= n + 2n + 16[n/4 + 4T(n/8)] = n + 2n + 4n + 4T(n/8)$$
$$= n + 2n + 4n + 8n + \cdots = n\underbrace{\sum_{i=0}^{\lg n - 1} 2^k}_{n\frac{2^{\lg n}-1}{2-1}} + \underbrace{4^{\lg n} T(1)}$$

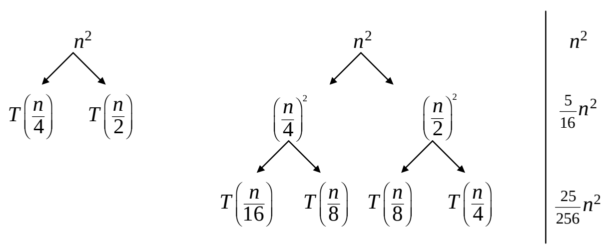
$\Theta(n^2)$ $\Theta(n^2)$

- **Disadvantages:**
  - » **Tedious**
  - » **Error-prone**

- **Use to generate initial guess, and then prove by induction !**

Recursion Tree

- **Example:** $T(n) = T(n/4) + T(n/2) + n^2$



- **At *k*-th level we get a general formula: *i* steps right, *k-i* left**

$$n^2 \sum_i \binom{k}{i} \left[2^{-i} 4^{-(k-i)}\right]^2 = n^2 \sum_i \binom{k}{i} \left[4^{-i} 16^{-(k-i)}\right] =$$
$$= n^2 \left[\frac{1}{4} + \frac{1}{16}\right]^k = n^2 \left[\frac{5}{16}\right]^k$$

- **Summing over all k, geometric sum, sums to** $\Theta(n^2)$ **(overcount, since T(1)=1)**

Master Method

- Consider the following recurrence: $T(n) = aT(n/b) + f(n);\ a \ge 1, b > 1$

  1. $f(n) = O(n^{\lg_b a - \epsilon}), \epsilon > 0 \quad \Rightarrow \quad \Theta(n^{\lg_b a})$

  2. $f(n) = \Theta(n^{\lg_b a} \lg^k n),\ k \ge 0 \quad \Rightarrow \quad \Theta(n^{\lg_b a} \lg^{k+1} n)$

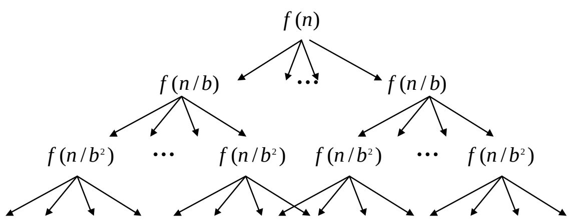
  3. $\left.\begin{array}{l} f(n) = \Omega(n^{\lg_b a + \epsilon}), \epsilon > 0 \\ af(n/b) \le cf(n) \text{ for some } c < 1 \end{array}\right\} \Rightarrow \quad \Theta(f(n))$

- More general than the book.

- Let $\Theta = n^{\lg_b a}$. Then the cases are:
  - » Q polynomially larger than f.
  - » f is larger than Q by a polylog factor.
  - » Q polynomially smaller than f.

33

Build recursion tree



$f(n)$

$af(n/b)$

$a^2 f(n/b^2)$

Last row: $\Theta(a^{\lg_b n}) = \Theta(n^{\lg_b a})$ elements, each one $\Theta(1)$.

Total: $\Theta(n^{\lg_b a}) + \sum_{i=1}^{\lg_b n - 1} a^i f(n/b^i)$

Which term dominates ?

34

First case: “f(n) small”

$$\frac{n^{\lg_b a}}{f(n)} = \Omega(n^{\varepsilon}) \Rightarrow \exists c \text{ s.t for "large enough n"}, \; f(n) \le c n^{\lg_b a} / n^{\varepsilon}$$

$$a^j f(n/b^j) \le c a^j (n/b^j)^{\lg_b a - \varepsilon} = c n^{\lg_b a - \varepsilon} a^j \frac{b^{j\varepsilon}}{b^{j \lg_b a}} = c n^{\lg_b a - \varepsilon} b^{j\varepsilon}$$

The ratio summed over all possible j: $\frac{b^{\varepsilon \lg_b n} - 1}{b^{\varepsilon} - 1} = \Theta(n^{\varepsilon})$.

Total: $O(n^{\lg_b a})$.

Lower bound is trivial (Why ?? First term in the original expression was already $\Theta(n^{\lg_b a})$.)

35

Second case

$$f(n) = \Theta(n^{\lg_b a} \lg^k n)$$

$$\sum \underbrace{a^j \left(\frac{n}{b^j}\right)^{\lg_b a}}_{= n^{\lg_b a}} \underbrace{\lg^k \left(\frac{n}{b^j}\right)}_{\le \lg^k n} = O(\lg^{k+1} n) n^{\lg_b a}$$ (there are $O(\lg n)$ elements in the sum)

This is an UPPER bound ! How to prove the lower bound ??

Rough and easy approach:

$$\sum_{i=1}^{\lg_b n - 1} \lg^k \left(\frac{n}{b^j}\right) \ge \sum_{i=1}^{(\lg_b n)/2} \lg^k \left(\frac{n}{b^j}\right) \ge \sum_{i=1}^{(\lg_b n)/2} \lg^k \sqrt{n} = (\text{const}) \; \lg^{k+1} n$$

(Note that we use the assumption that $k \ge 0$)

36

Third case

$a^j f(n/b^j) \le c^j f(n)$ for some $c<1$, and $f(n) = \Omega(n^{\lg_b a+e})$

$\Rightarrow \sum_{i=1}^{\lg_b n-1} c^j f(n) = \Theta(f(n))$

$\Rightarrow \sum_{i=1}^{\lg_b n-1} a^j f(n/b^j) = O(f(n))$ Note Big-Oh and not Theta !

The first term is already $\Theta(n^{\lg_b a}) = O(f(n))$

TOTAL: $\Theta(f(n))$

37