Design & Analysis of Algorithms

[Solving Recurrence]

Soharab Hossain Shaikh
BML Munjal University

1

Recurrences and Running Time

- An equation or inequality that describes a function in terms of its value on smaller inputs.

  $T(n) = T(n-1) + n$

- Recurrences arise when an algorithm contains recursive calls to itself
- What is the running time of the algorithm?
- Need to solve the recurrence
  - Find an explicit formula of the expression
  - Bound the recurrence by an expression that involves n

2

2

Example Recurrences

- $T(n) = T(n-1) + n$ $\Theta(n^2)$
  - Recursive algorithm that loops through the input to eliminate one item
- $T(n) = T(n/2) + c$ $\Theta(\lg n)$
  - Recursive algorithm that halves the input in one step
- $T(n) = 2T(n/2) + 1$ $\Theta(n)$
  - Recursive algorithm that splits the input into 2 halves and does a constant amount of other work

3

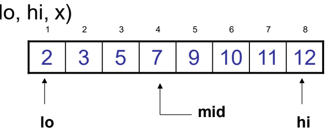
3

Recurrent Algorithm: Binary Search

- For an ordered array A, finds if *x* is in the array A[lo…hi]

Algo BINARY-SEARCH (A, lo, hi, x)

if (lo > hi)
    return FALSE
mid ← ⌊(lo+hi)/2⌋
if x = A[mid]
    return TRUE
if (x < A[mid])
    BINARY-SEARCH (A, lo, mid-1, x)
if (x > A[mid])
    BINARY-SEARCH (A, mid+1, hi, x)



4

4

Analysis of Binary Search

Algo: BINARY-SEARCH (A, lo, hi, x)
  if (lo > hi) ← constant time: c_1
    return FALSE
  mid ← $\lfloor$(lo+hi)/2$\rfloor$ ← constant time: c_2
  if x = A[mid] ← constant time: c_3
    return **TRUE**
  if (x < A[mid])
    BINARY-SEARCH (A, lo, mid-1, x) ← same problem of size n/2
  if (x > A[mid])
    BINARY-SEARCH (A, mid+1, hi, x) ← same problem of size n/2

- $T(n) = c + T(n/2)$
  - T(n) – running time for an array of size n

5

5

Methods for Solving Recurrences

- **Iteration Method**
- **Recursion Tree Method**
- **Master's Method**

6

6

The Iteration Method

- Convert the recurrence into a summation and try to bound it using known series
  - Iterate the recurrence until the initial condition is reached.
  - Use back-substitution to express the recurrence in terms of *n* and the initial (boundary) condition.

7

7

The Iteration Method Example:1

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

$T(n) = c + T(n/2)$ $T(n/2) = c + T(n/4)$
$= c + c + T(n/4)$ $T(n/4) = c + T(n/8)$
$= c + c + c + T(n/8)$

Assume $n = 2^k$

$$T(n) = \underbrace{c + c + \ldots + c}_{k \text{ times}} + T(1)$$

$= c\lg n + T(1)$
$= \Theta(\lg n)$

8

8

The Iteration Method
Example:2

$T(n) = n + 2T(n/2)$ Assume: $n = 2^k$

$T(n/2) = n/2 + 2T(n/4)$

$$
\begin{aligned}
T(n) &= n + 2T(n/2) \\
&= n + 2(n/2 + 2T(n/4)) \\
&= n + n + 4T(n/4) \\
&= n + n + 4(n/4 + 2T(n/8)) \\
&= n + n + n + 8T(n/8) \\
\ldots &= in + 2^iT(n/2^i) \\
&= kn + 2^kT(1) \\
&= n\lg n + nT(1) = \Theta(n\lg n)
\end{aligned}
$$

9

9

Changing Variables

$T(n) = 2T(\sqrt{n}\,) + \lg n$

– Rename: $m = \lg n \Rightarrow n = 2^m$

$T(2^m) = 2T(2^{m/2}) + m$

– Rename: $S(m) = T(2^m)$

$S(m) = 2S(m/2) + m \Rightarrow S(m) = O(m\lg m)$
(demonstrated before)

$T(n) = T(2^m) = S(m) = O(m\lg m) = O(\lg n\lg\lg n)$

Idea: transform the recurrence to one that you have seen before

10

10

Recursion Tree Method

Convert the recurrence into a tree:

– Each node represents the cost incurred at various levels of recursion

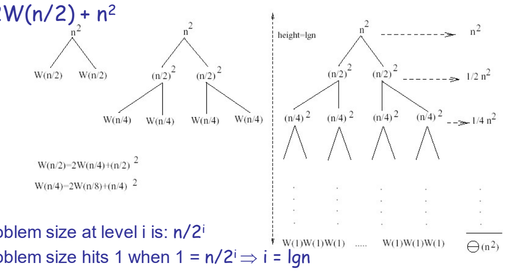
– Sum up the costs of all levels

11

11

Example 1

$W(n) = 2W(n/2) + n^2$



- Subproblem size at level i is: $n/2^i$
- Subproblem size hits 1 when $1 = n/2^i \Rightarrow i = \lg n$
- Cost of the problem at level i = $(n^2/2^i)$ No. of nodes at level i = 2^i
- Total cost:

$$W(n) = \sum_{i=0}^{\lg n-1} \frac{n^2}{2^i} + 2^{\lg n}W(1) = n^2\sum_{i=0}^{\lg n-1}\left(\frac{1}{2}\right)^i + n \le n^2\sum_{i=0}^{\infty}\left(\frac{1}{2}\right)^i + O(n) = n^2\frac{1}{1-\frac{1}{2}} + O(n) = 2n^2$$

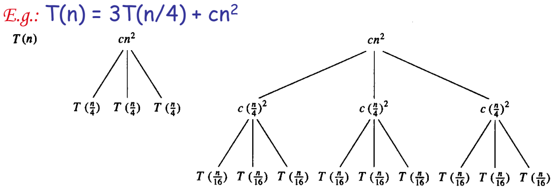
$\Rightarrow W(n) = O(n^2)$

12

12

Example 2

E.g.: $T(n) = 3T(n/4) + cn^2$



- Subproblem size at level i is: $n/4^i$
- Subproblem size hits 1 when $1 = n/4^i \Rightarrow i = \log_4 n$
- Cost of a node at level $i = c(n/4^i)^2$
- Number of nodes at level $i = 3^i \Rightarrow$ last level has $3^{\log_4 n} = n^{\log_4 3}$ nodes
- Total cost:

$$T(n) = \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) \le \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta\left(n^{\log_4 3}\right) = \frac{1}{1-\frac{3}{16}} cn^2 + \Theta\left(n^{\log_4 3}\right) = O(n^2)$$

$\Rightarrow T(n) = O(n^2)$

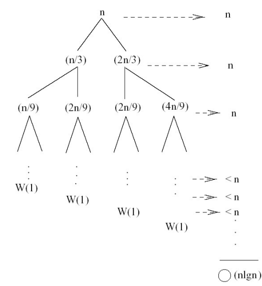
13

13

Example 3

$W(n) = W(n/3) + W(2n/3) + n$

- The longest path from the root to a leaf is: $n \to (2/3)n \to (2/3)^2 n \to \dots \to 1$
- Subproblem size hits 1 when $1 = (2/3)^i n \Leftrightarrow i = \log_{3/2} n$
- Cost of the problem at level $i = n$
- Total cost:



$$W(n) < n + n + \dots = n(\log_{3/2} n) = n\frac{\lg n}{\lg \frac{3}{2}} = O(n \lg n)$$

$\Rightarrow W(n) = O(n\lg n)$

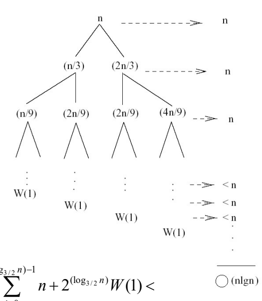
14

14

Example 3

$W(n) = W(n/3) + W(2n/3) + n$

- The longest path from the root to a leaf is: $n \to (2/3)n \to (2/3)^2 n \to \dots \to 1$
- Subproblem size hits 1 when $1 = (2/3)^i n \Leftrightarrow i = \log_{3/2} n$
- Cost of the problem at level $i = n$
- Total cost: $W(n) < n + n + \dots = \sum_{i=0}^{(\log_{3/2} n)-1} n + 2^{(\log_{3/2} n)} W(1) <$



$$< n\sum_{i=0}^{\log_{3/2} n} 1 + n^{\log_{3/2} 2} = n\log_{3/2} n + O(n) = n\frac{\lg n}{\lg 3/2} + O(n) = \frac{1}{\lg 3/2} n \lg n + O(n)$$

$\Rightarrow W(n) = O(n\lg n)$

15

15

Master's Method

- "Cookbook" for solving recurrences of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where, $a \ge 1$, $b > 1$, and $f(n) > 0$

Idea: compare $f(n)$ with $n^{\log_b a}$

- $f(n)$ is asymptotically smaller or larger than $n^{\log_b a}$ by a polynomial factor n^{ε}
- $f(n)$ is asymptotically equal with $n^{\log_b a}$

16

16

Master's Method

- "Cookbook" for solving recurrences of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where, $a \geq 1$, $b > 1$, and $f(n) > 0$

Case 1: if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then: $T(n) = \Theta(n^{\log_b a})$

Case 2: if $f(n) = \Theta(n^{\log_b a})$, then: $T(n) = \Theta(n^{\log_b a} \lg n)$

Case 3: if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, and if

$af(n/b) \leq cf(n)$ for some $c < 1$ and all sufficiently large n, then:

$T(n) = \Theta(f(n))$

regularity condition

17

17

Why $n^{\log_b a}$?

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$$T(n) = a\underbrace{T\left(\frac{n}{b}\right)}$$

$$a^2\underbrace{T\left(\frac{n}{b^2}\right)}$$

$$a^3T\left(\frac{n}{b^3}\right)$$

$$\vdots$$

$$T(n) = a^i T\left(\frac{n}{b^i}\right) \quad \forall i$$

- Assume $n = b^k \Rightarrow k = \log_b n$
- At the end of iteration $i = k$:

$$T(n) = a^{\log_b n} T\left(\frac{b^i}{b^i}\right) = a^{\log_b n} T(1) = \Theta\left(a^{\log_b n}\right) = \Theta\left(n^{\log_b a}\right)$$

- Case 1:
  - If $f(n)$ is dominated by $n^{\log_b a}$:
    - $T(n) = \Theta(n^{\log_b n})$
- Case 3:
  - If $f(n)$ dominates $n^{\log_b a}$:
    - $T(n) = \Theta(f(n))$
- Case 2:
  - If $f(n) = \Theta(n^{\log_b a})$:
    - $T(n) = \Theta(n^{\log_b a} \log n)$

18

18

Example

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

$a = 2$, $b = 2$, $\log_2 2 = 1$

Compare $n^{\log_2 2}$ with $f(n) = n$

$\Rightarrow f(n) = \Theta(n) \Rightarrow$ Case 2

$\Rightarrow T(n) = \Theta(n\lg n)$

19

19

Example

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

$a = 2$, $b = 2$, $\log_2 2 = 1$

Compare n with $f(n) = n^2$

$\Rightarrow f(n) = \Omega(n^{1+\varepsilon})$ Case 3 $\Rightarrow$ verify regularity cond.

$a\, f(n/b) \leq c\, f(n)$

$\Leftrightarrow 2\, n^2/4 \leq c\, n^2 \Rightarrow c = \frac{1}{2}$ is a solution ($c<1$)

$\Rightarrow T(n) = \Theta(n^2)$

20

20

Example

$T(n) = 2T(n/2) + \sqrt{n}$

a = 2, b = 2, $\log_2 2 = 1$

Compare n with $f(n) = n^{1/2}$

$\Rightarrow f(n) = O(n^{1-\varepsilon})$ Case 1

$\Rightarrow T(n) = \Theta(n)$

21

21

Example

$T(n) = 3T(n/4) + n\lg n$

a = 3, b = 4, $\log_4 3 = 0.793$

Compare $n^{0.793}$ with $f(n) = n\lg n$

$f(n) = \Omega(n^{\log_4 3+\varepsilon})$ Case 3

Check regularity condition:

$3*(n/4)\lg(n/4) \leq (3/4)n\lg n = c*f(n)$, $c=3/4$

$\Rightarrow T(n) = \Theta(n\lg n)$

22

22

If f(n) = Θ(n^c)

$$T(n) = aT\left(\frac{n}{b}\right) + n^c \quad a \geq 1, b \geq 1, c > 0$$

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & a > b^c \\ \Theta(n^c \log_b n) & a = b^c \\ \Theta(n^c) & a < b^c \end{cases}$$

23

23

Inadmissible Cases

- $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$

  a is not a constant

- $T(n) = 0.5T\left(\frac{n}{2}\right) + n$

  a<1 cannot have less than one sub problem

- $T(n) = 64T\left(\frac{n}{8}\right) - n^2 \log n$

  f(n) is not positive

24

24

Proof

$$\boxed{\begin{array}{l} T(n) = aT\left(\frac{n}{b}\right) + n^c \quad a \geq 1, b \geq 1, c > 0 \\ \Downarrow \\ T(n) = \begin{cases} \Theta(n^{\log_b a}) & a > b^c \\ \Theta(n^c \log_b n) & a = b^c \\ \Theta(n^c) & a < b^c \end{cases} \end{array}}$$

Proof (Iteration method)

$$\begin{aligned} T(n) &= aT\left(\tfrac{n}{b}\right) + n^c \\ &= n^c + a\left(\left(\tfrac{n}{b}\right)^c + aT\left(\tfrac{n}{b^2}\right)\right) \\ &= n^c + \left(\tfrac{a}{b^c}\right)n^c + a^2T\left(\tfrac{n}{b^2}\right) \\ &= n^c + \left(\tfrac{a}{b^c}\right)n^c + a^2\left(\left(\tfrac{n}{b^2}\right)^c + aT\left(\tfrac{n}{b^3}\right)\right) \\ &= n^c + \left(\tfrac{a}{b^c}\right)n^c + \left(\tfrac{a}{b^c}\right)^2 n^c + a^3T\left(\tfrac{n}{b^3}\right) \\ &= \ldots \\ &= n^c + \left(\tfrac{a}{b^c}\right)n^c + \left(\tfrac{a}{b^c}\right)^2 n^c + \left(\tfrac{a}{b^c}\right)^3 n^c + \left(\tfrac{a}{b^c}\right)^4 n^c + \ldots + \left(\tfrac{a}{b^c}\right)^{\log_b n - 1} n^c + a^{\log_b n}T(1) \\ &= n^c \textstyle\sum_{k=0}^{\log_b n - 1} \left(\tfrac{a}{b^c}\right)^k + a^{\log_b n} \\ &= n^c \textstyle\sum_{k=0}^{\log_b n - 1} \left(\tfrac{a}{b^c}\right)^k + n^{\log_b a} \end{aligned}$$

Recall geometric sum $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1} = \Theta(x^n)$

25

Proof

- $\boxed{a < b^c}$

$a < b^c \Leftrightarrow \frac{a}{b^c} < 1 \Rightarrow \sum_{k=0}^{\log_b n - 1} \left(\frac{a}{b^c}\right)^k \leq \sum_{k=0}^{+\infty} \left(\frac{a}{b^c}\right)^k = \frac{1}{1-\left(\frac{a}{b^c}\right)} = \Theta(1)$

$a < b^c \Leftrightarrow \log_b a < \log_b b^c = c$

$$\begin{aligned} T(n) &= n^c \textstyle\sum_{k=0}^{\log_b n - 1} \left(\tfrac{a}{b^c}\right)^k + n^{\log_b a} \\ &= n^c \cdot \Theta(1) + n^{\log_b a} \\ &= \Theta(n^c) \end{aligned}$$

- $\boxed{a = b^c}$

$a = b^c \Leftrightarrow \frac{a}{b^c} = 1 \Rightarrow \sum_{k=0}^{\log_b n - 1} \left(\frac{a}{b^c}\right)^k = \sum_{k=0}^{\log_b n - 1} 1 = \Theta(\log_b n)$

$a = b^c \Leftrightarrow \log_b a = \log_b b^c = c$

$$\begin{aligned} T(n) &= \textstyle\sum_{k=0}^{\log_b n - 1} \left(\tfrac{a}{b^c}\right)^k + n^{\log_b a} \\ &= n^c\Theta(\log_b n) + n^{\log_b a} \\ &= \Theta(n^c \log_b n) \end{aligned}$$

26

Proof

- $\boxed{a > b^c}$

$a > b^c \Leftrightarrow \frac{a}{b^c} > 1 \Rightarrow \sum_{k=0}^{\log_b n - 1} \left(\frac{a}{b^c}\right)^k = \Theta\left(\left(\frac{a}{b^c}\right)^{\log_b n}\right) = \Theta\left(\frac{a^{\log_b n}}{(b^c)^{\log_b n}}\right) = \Theta\left(\frac{a^{\log_b n}}{n^c}\right)$

$$\begin{aligned} T(n) &= n^c \cdot \Theta\left(\tfrac{a^{\log_b n}}{n^c}\right) + n^{\log_b a} \\ &= \Theta(n^{\log_b a}) + n^{\log_b a} \\ &= \Theta(n^{\log_b a}) \end{aligned}$$

27

End of Lecture

28