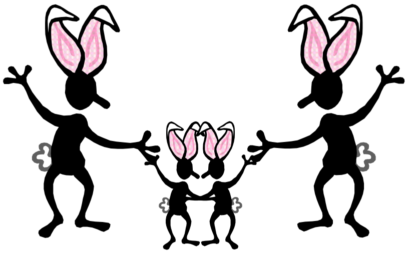
Design
and
Analysis of Algorithms

Recurrences
Instructor:Dr. M. A. Q.

Thanks to George Bebis **and** Shafi Goldwasser

Recurrence



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 actual 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**

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
- $T(n) = T(n/2) + n$ $\Theta(n)$
  - Recursive algorithm that halves the input but must examine every item in the input
- $T(n) = 2T(n/2) + 1$ $\Theta(n)$
  - Recursive algorithm that splits the input into 2 halves and does a constant amount of other work

Recurrent Algorithms
BINARY-SEARCH

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

Alg.: BINARY-SEARCH (A, lo, hi, x)

1	2	3	4	5	6	7	8
2	3	5	7	9	10	11	12
lo			mid				hi

```
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)
```


Example

- A[8] = {1, 2, 3, 4, 5, 7, 9, 11}
  - lo = 1 hi = 8 x = 7

1	2	3	4	5	6	7	8
1	2	3	4	5	7	9	11

mid = 4, lo = 5, hi = 8

				5	6	7	8
1	2	3	4	5	7	9	11

mid = 6, A[mid] = x
Found!

Another Example

- A[8] = {1, 2, 3, 4, 5, 7, 9, 11}
  - lo = 1 hi = 8 x = 6

1	2	3	4	5	6	7	8	
1	2	3	4	5	7	9	11	mid = 4, lo = 5, hi = 8
↑ low							↑ high	
1	2	3	4	5	7	9	11	mid = 6, A[6] = 7, lo = 5, hi = 5
				↑ low			↑ high	
1	2	3	4	5	7	9	11	mid = 5, A[5] = 5, lo = 6, hi = 5 NOT FOUND!
				↑	↑			
1	2	3	4	5	7	9	11	
			high ↑		↑ low			

7

Analysis of BINARY-SEARCH

Alg.: BINARY-SEARCH (A, lo, hi, x)

```
if (lo > hi)                                  ← constant time: c1
    return FALSE
mid ← ⌊(lo+hi)/2⌋                             ← constant time: c2
if x = A[mid]                                 ← constant time: c3
    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

8

Methods for Solving Recurrences

- Iteration method
- Substitution method
- Recursion tree method
- Master method

9

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.

10

The Iteration Method

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

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

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

$$\begin{aligned} T(n) &= c + T(n/2) \\ &= c + c + T(n/4) \\ &= c + c + c + T(n/8) \end{aligned}$$

Assume $n = 2^k$

$$\begin{aligned} T(n) &= \underbrace{c + c + \dots + c}_{k \text{ times}} + T(1) \\ &= c\lg n + T(1) \\ &= \Theta(\lg n) \end{aligned}$$

11

Iteration Method – Example

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

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

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

12

The substitution method

1. Guess a solution
2. Use induction to prove that the solution works

Recall: Integer Multiplication

- Let X = $\boxed{A|B}$ and Y = $\boxed{C|D}$ where A,B,C and D are n/2 bit integers
- Simple Method: $XY = (2^{n/2}A+B)(2^{n/2}C+D)$
- Running Time Recurrence

  $T(n) < 4T(n/2) + 100n$

How do we solve it?

Substitution method

The most general method:

1. ***Guess*** the form of the solution.
2. ***Verify*** by induction.
3. ***Solve*** for constants.

Example: $T(n) = 4T(n/2) + 100n$

- [Assume that $T(1) = \Theta(1)$.]
- Guess $O(n^3)$. (Prove O and Ω separately.)
- Assume that $T(k) \le ck^3$ for $k < n$.
- Prove $T(n) \le cn^3$ by induction.

Example of substitution

$$\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\le 4c(n/2)^3 + 100n \\
&= (c/2)n^3 + 100n \\
&= cn^3 - ((c/2)n^3 - 100n) \quad \longleftarrow \textit{desired – residual} \\
&\le cn^3 \quad \longleftarrow \textit{desired}
\end{aligned}$$

whenever $(c/2)n^3 - 100n \ge 0$, for example, if $c \ge 200$ and $n \ge 1$.

residual

Example (continued)

- We must also handle the initial conditions, that is, ground the induction with base cases.
- ***Base:*** $T(n) = \Theta(1)$ for all $n < n_0$, where n_0 is a suitable constant.
- For $1 \le n < n_0$, we have "$\Theta(1)$" $\le cn^3$, if we pick c big enough.

This bound is not tight!

A tighter upper bound?

We shall prove that $T(n) = O(n^2)$.

Assume that $T(k) \le ck^2$ for $k < n$:

$$\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\le cn2 + 100n \\
&\le cn2
\end{aligned}$$

for ***no*** choice of $c > 0$. Lose!

A tighter upper bound!

IDEA: Strengthen the inductive hypothesis.
- ***Subtract*** a low-order term.

Inductive hypothesis: $T(k) \leq c_1k^2 - c_2k$ for $k < n$.

$$
\begin{aligned}
T(n) &= 4T(n/2) + 100n \\
&\leq 4(c_1(n/2)^2 - c_2(n/2)) + 100n \\
&= c_1n^2 - 2c_2n + 100n \\
&= c_1n^2 - c_2n - (c_2n - 100n) \\
&\leq c_1n^2 - c_2n \quad \text{if } c_2 > 100.
\end{aligned}
$$

Pick c_1 big enough to handle the initial conditions.

Substitution Method

Example #1 (yes, it's right out of the book)

Solve $T(n) = 2T(\lfloor n/2 \rfloor) + n$

This looks very much like $T(n) = 2T(n/2) + n$, which is the recurrence for mergesort. So, let's assume the same solution: $T(n) = n\lg n$. Show that $T(n) \leq cn\lg n$ for $c \geq 0$ and $n \geq n_0 \geq 1$. We do this by induction. We assume that the recurrence holds for $\lfloor n/2 \rfloor$. This leads to

$T(\lfloor n/2 \rfloor) = c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$.

Sub into recurrence to get

$T(n) = 2\, c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor) + n$

ctd...

Substitution Method

$T(n) = 2\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor) + n$ *(repeated from last page)*

Simplify. We know that $\lfloor n \rfloor \leq n$, so definitely $\lfloor n/2 \rfloor \leq n/2$. Why? consider cases.

What if n is even? Then let n = 2m, and sub back in:

$\lfloor n/2 \rfloor = \lfloor 2m/2 \rfloor = \lfloor m \rfloor \leq m = n/2$.

What about odd n? Then n = 2m+1. Sub in to get:

$\lfloor n/2 \rfloor = \lfloor (2m+1)/2 \rfloor = \lfloor m + ½ \rfloor \leq m + ½ = (2m+1)/2 = n/2$

So, $\lfloor n/2 \rfloor \leq n/2$ for even or odd n.

(It really worked!!! I am definitely shocked!)

Substitution Method

So, back to the problem.

$T(n) = 2\, c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor) + n$

$\leq 2c(n/2)\lg(\lfloor n/2 \rfloor) + n$

$\leq 2c(n/2)\lg(n/2) + n$

But

$\lg(n/2) = \lg n - \lg 2 = \lg n - \log_2 2^1 = \lg n - 1$ *(Sorry for the details)*

So,

$T(n) \leq 2c(n/2)(\lg n - 1) + n$

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

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

$\leq cn\lg n$ for $1-c \leq 0$ or $c \geq 1$

Substitution Method

Now we just need to verify boundary conditions.

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

Plug in n = 1:

T(1) = 0 since ln1 = 0. But the recurrence gives something different:

$T(1) = 2T(0)+1 = 0+1 = 1 \neq 0$, so this fails. Must have T(1)=1.

So our solution is not valid for n=1. But it doesn't have to be valid for every n; only for some $n \geq n_0$

Substitution Method

So, consider n=2. T(2) = 2cln2 = 2c. The bc of the recurrence gives

T(1) = 1, so use this in Recurrence yields

T(2) = 2T(1) + 2 = 4

T(3) = 2T(1) + 3 = 5

T(4) = 2T(2) + 4

Just let $n \geq 4$. Then our solution does not depend on T(1).

We now have to prove our solution holds for $n \geq 4$ by picking the appropriate value for c so that the boundary conditions at n=2 and n=3 hold.

Substitution Method

Now just pick a value for c that holds for n = 2 and n = 3

$T(2) = 4 \le c2 \cdot \lg 2 = 2c$

$T(3) = 5 \le c3 \cdot \lg 3$

The first equation gives us

$c \ge 2$.

If we put c=2 into the second equation, we get

$5 \le 6 \cdot \lg 3$

But lg 3 > 1. So $6 \cdot \lg 3 > 6$, so definitely $6 \cdot \lg 3 \ge 5$.

So, for $n \ge 4$ and c = 2, we have proven, by induction, that $T(n) = n \cdot \lg n$

Example: Binary Search

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

- Guess: $T(n) = O(\lg n)$
  - Induction goal: $T(n) \le c \lg n$, for some c and $n \ge n_0$
  - Induction hypothesis: $T(n/2) \le c \lg(n/2)$
- Proof of induction goal:

  $T(n) = T(n/2) + c1 \le c \lg(n/2) + c1$

  $= c \lg n - c + c1 \le c \lg n$

  if: $-c + c1 \le 0$, $c \ge c1$
- Base case?

Example 2

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

- Guess: $T(n) = O(n^2)$
  - Induction goal: $T(n) \le c\, n^2$, for some c and $n \ge n_0$
  - Induction hypothesis: $T(n-1) \le c(n-1)^2$ for all k < n
- Proof of induction goal:

  $T(n) = T(n-1) + n \le c\,(n-1)^2 + n$

  $= cn^2 - (2cn - c - n) \le cn^2$

  if: $2cn - c - n \ge 0 \Leftrightarrow c \ge n/(2n-1) \Leftrightarrow c \ge 1/(2 - 1/n)$
  - For $n \ge 1 \Rightarrow 2 - 1/n \ge 1 \Rightarrow$ any $c \ge 1$ will work

Example 3

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

- Guess: $T(n) = O(n\lg n)$
  - Induction goal: $T(n) \le cn \lg n$, for some c and $n \ge n_0$
  - Induction hypothesis: $T(n/2) \le cn/2 \lg(n/2)$
- Proof of induction goal:

  $T(n) = 2T(n/2) + n \le 2c\,(n/2)\lg(n/2) + n$

  $= cn \lg n - cn + n \le cn \lg n$

  if: $-cn + n \le 0 \Rightarrow c \ge 1$
- Base case?

The 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

Used to "guess" a solution for the recurrence

Recursion-tree method

- A recursion tree models the costs (time) of a recursive execution of an algorithm.
- The recursion tree method is good for generating guesses for the substitution method.
- The recursion-tree method can be unreliable, just like any method that uses ellipses (…).
- The recursion-tree method promotes intuition, however.

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

L2.31

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

$T(n)$

L2.32

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

n^2

$T(n/4)$ $\quad$ $T(n/2)$

L2.33

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

n^2

$(n/4)^2$ $\quad$ $(n/2)^2$

$T(n/16)$ $\quad$ $T(n/8)$ $\quad$ $T(n/8)$ $\quad$ $T(n/4)$

L2.34

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

n^2

$(n/4)^2$ $\quad$ $(n/2)^2$

$(n/16)^2$ $\quad$ $(n/8)^2$ $\quad$ $(n/8)^2$ $\quad$ $(n/4)^2$

$\vdots$

$\Theta(1)$

L2.35

Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:

n^2 $\quad\cdots\cdots\cdots$ n^2

$(n/4)^2$ $\quad$ $(n/2)^2$

$(n/16)^2$ $\quad$ $(n/8)^2$ $\quad$ $(n/8)^2$ $\quad$ $(n/4)^2$

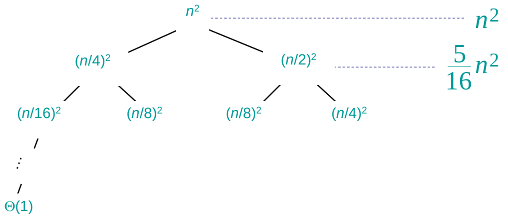
$\vdots$

$\Theta(1)$

L2.36

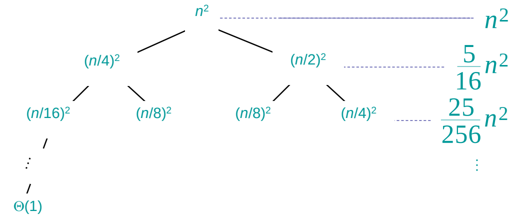
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



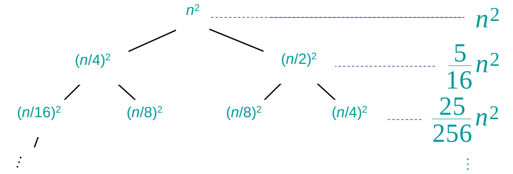
Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



Example of recursion tree

Solve $T(n) = T(n/4) + T(n/2) + n^2$:



$\Theta(1)$

Total $= n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$

$= \Theta(n^2)$ *geometric series*

Appendix: geometric series

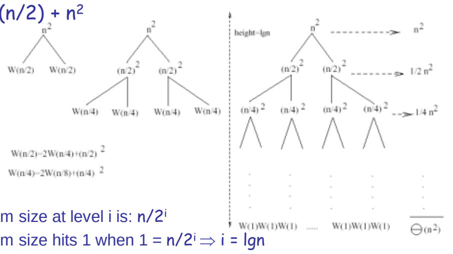
$$1 + x + x^2 + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x} \quad \text{for } x \neq 1$$

$$1 + x + x^2 + \cdots = \frac{1}{1 - x} \quad \text{for } |x| < 1$$

Return to last slide viewed.

Example 1

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



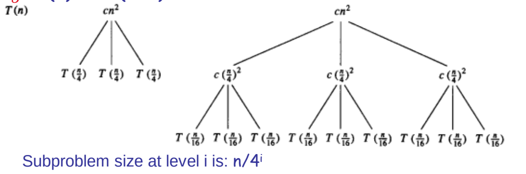
- Subproblem size at level i is: $n/2^i$
- Subproblem size hits 1 when $1 = n/2^i \Rightarrow i = \lg n$
- Cost of the problem at level $i = (n/2^i)^2$ No. of nodes at level $i = 2^i$
- Total cost:

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

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

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:

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

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

Example 2 - Substitution

$T(n) = 3T(n/4) + cn^2$

- Guess: $T(n) = O(n^2)$
  - Induction goal: $T(n) \le dn^2$, for some d and $n \ge n_0$
  - Induction hypothesis: $T(n/4) \le d\,(n/4)^2$
- Proof of induction goal:

  $T(n) = 3T(n/4) + cn^2$

  $\le 3d\,(n/4)^2 + cn^2$

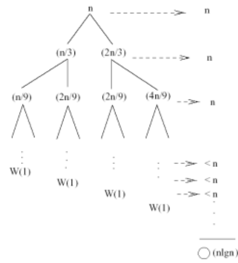
  $= (3/16)\,d\,n^2 + cn^2$

  $\le d\,n^2$ if: $d \ge (16/13)c$

- Therefore: $T(n) = O(n^2)$

Example 3 (simpler proof)

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



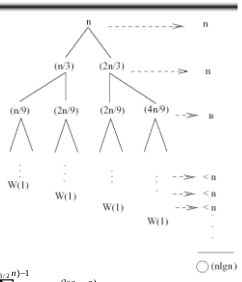
- The longest path from the root to a leaf is: $n \to (2/3)n \to (2/3)^2 n \to \ldots \to 1$
- Subproblem size hits 1 when $1 = (2/3)^i n \Leftrightarrow i = \log_{3/2} n$
- Cost of the problem at level $i = n$
- Total cost:

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

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

Example 3

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



- The longest path from the root to a leaf is: $n \to (2/3)n \to (2/3)^2 n \to \ldots \to 1$
- 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:

$$W(n) < n + n + \ldots = \sum_{i=0}^{(\log_{3/2} n)-1} n + 2^{(\log_{3/2} n)} W(1) <$$

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

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

Example 3 - Substitution

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

- Guess: $W(n) = O(n\lg n)$
  - Induction goal: $W(n) \le dn\lg n$, for some d and $n \ge n_0$
  - Induction hypothesis: $W(k) \le d\,k\lg k$ for any $K < n$ $(n/3, 2n/3)$
- Proof of induction goal:

  Try it out as an exercise!!

- $T(n) = O(n\lg n)$

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 $n^{\log_b a}$

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 $f(n) > 0$

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

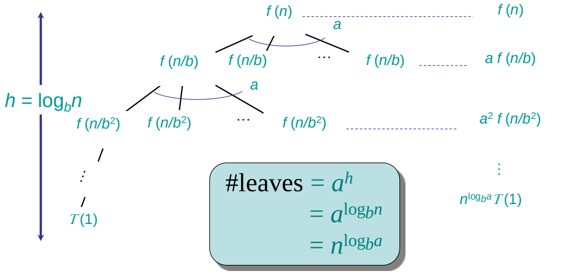
$af(n/b) \le cf(n)$ for some $c < 1$ and all sufficiently large n, then:

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

regularity condition

Idea of master theorem

Recursion tree:



Three common cases

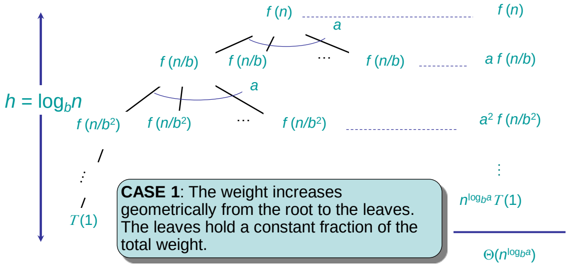
Compare $f(n)$ with $n^{\log_b a}$:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially slower than $n^{\log_b a}$ (by an n^{ε} factor).

   Solution: $T(n) = \Theta(n^{\log_b a})$.

Idea of master theorem

Recursion tree:



Three common cases

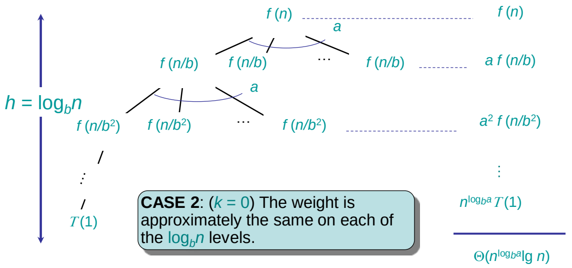
Compare $f(n)$ with $n^{\log_b a}$:

2. $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.
   - $f(n)$ and $n^{\log_b a}$ grow at similar rates.

   Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

Idea of master theorem

Recursion tree:



Three common cases (cont.)

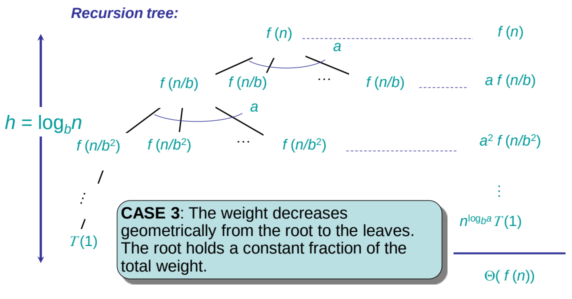
Compare $f(n)$ with $n^{\log_b a}$:

3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$.
   - $f(n)$ grows polynomially faster than $n^{\log_b a}$ (by an n^{ε} factor),

   and $f(n)$ satisfies the ***regularity condition*** that $a\,f(n/b) \leq c\,f(n)$ for some constant $c < 1$.

   Solution: $T(n) = \Theta(f(n))$.

Idea of master theorem



Examples

- ***Ex.*** $T(n) = 4T(n/2) + n$
- $a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n$
- CASE 1: $f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 1$
- $\therefore\ T(n) = \Theta(n^2)$

Ex. $T(n) = 4T(n/2) + n^2$
$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n^2$
CASE 2: $f(n) = \Theta(n^2 \lg^0 n)$, that is, $k = 0$
$\therefore\ T(n) = \Theta(n^2 \lg n)$

Examples

Ex. $T(n) = 4T(n/2) + n^3$
$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n^3$.
CASE 3: $f(n) = \Omega(n^{2+\varepsilon})$ for $\varepsilon = 1$
and $4(cn/2)^3 \le cn^3$ (reg. cond.) for $c = 1/2$.
$\therefore\ T(n) = \Theta(n^3)$.

Ex. $T(n) = 4T(n/2) + n^2/\lg n$
$a = 4$, $b = 2 \Rightarrow n^{\log_b a} = n^2$; $f(n) = n^2/\lg n$.
Master method does not apply. In particular, for every constant $\varepsilon > 0$, we have $n^{\varepsilon} = \omega(\lg n)$.

Examples

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

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

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

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

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

Examples

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

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

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

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

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

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

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

Examples (cont.)

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

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

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

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

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

Examples

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

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

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

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

Check regularity condition:

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

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

62

Examples

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

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

- Compare n with $f(n) = n\lg n$
  - seems like case 3 should apply
- f(n) must be polynomially larger by a factor of n^{ε}
- In this case it is only larger by a factor of $\lg n$

63

Readings

64