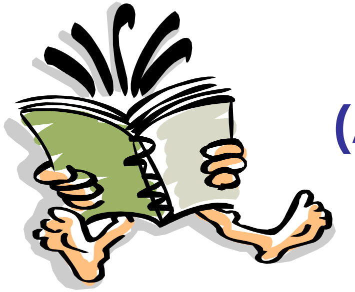
Analysis of Algorithms

Recurrences



(Appendix A, Chapter 4)

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

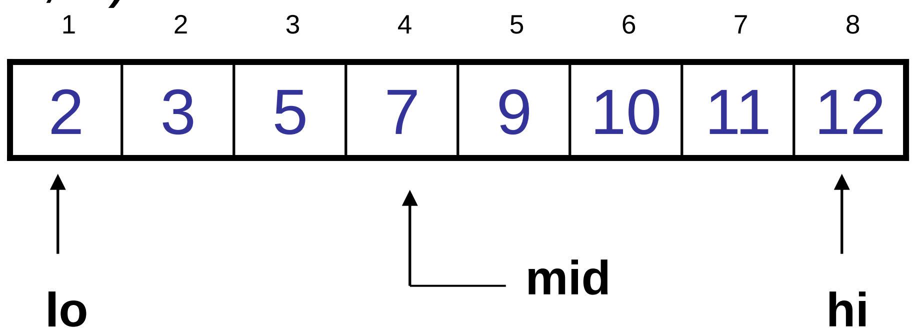
Example Recurrences

- $T(n) = T(n-1) + n$ $\Theta(n^2)$
  – 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)



if (lo > hi)

  return FALSE

mid ← $\lfloor$(lo+hi)/2$\rfloor$

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

low high

mid = 4, lo = 5, hi = 8

1	2	3	4	5	7	9	11

low high

mid = 6, A[6] = 7, lo = 5, hi = 5

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

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

Methods for Solving Recurrences

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

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.

The Iteration Method

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

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

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

Assume $n = 2^k$

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

Iteration Method – Example

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

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

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

The substitution method

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

Substitution method

- Guess a solution
  - $T(n) = O(g(n))$
  - Induction goal: apply the definition of the asymptotic notation
    - $T(n) \leq d\, g(n)$, for some $d > 0$ and $n \geq n_0$
  - Induction hypothesis: $T(k) \leq d\, g(k)$ for all $k < n$ (strong induction)
- Prove the induction goal
  - Use the **induction hypothesis** to find some values of the constants d and n_0 for which the **induction goal** holds

Example: Binary Search

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

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

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

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

if: $-d + c \le 0$, $d \ge c$

- 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:

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

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

$$\text{if: } 2cn - c - n \ge 0 \Leftrightarrow c \ge n/(2n-1) \Leftrightarrow c \ge 1/(2 - 1/n)$$

  - 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:

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

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

$$\text{if: } -cn + n \le 0 \Rightarrow c \ge 1$$

- Base case?

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

The recursion-tree method

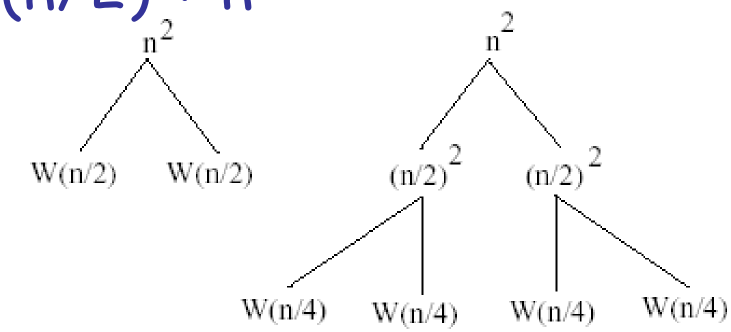
Convert the recurrence into a tree:

- 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

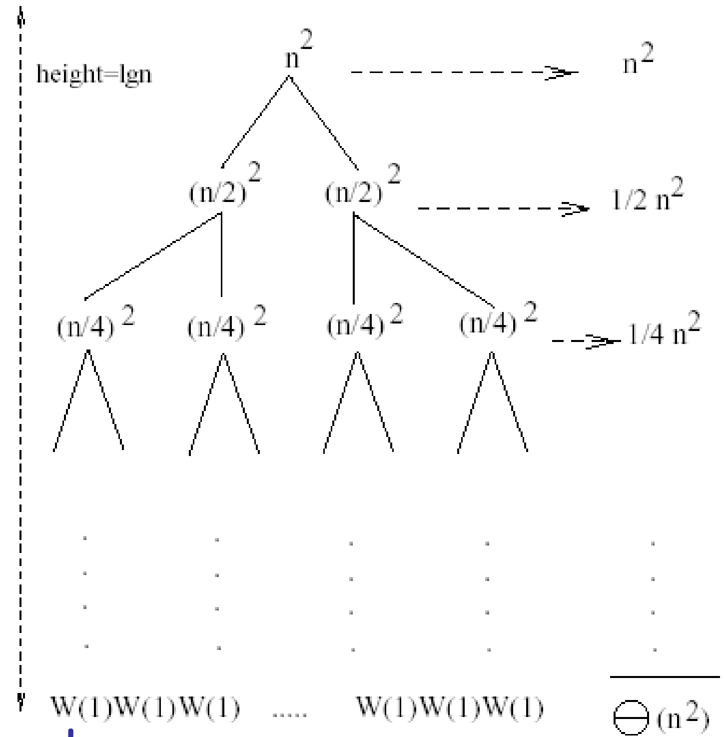
Example 1

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



$W(n/2)=2W(n/4)+(n/2)^2$

$W(n/4)=2W(n/8)+(n/4)^2$



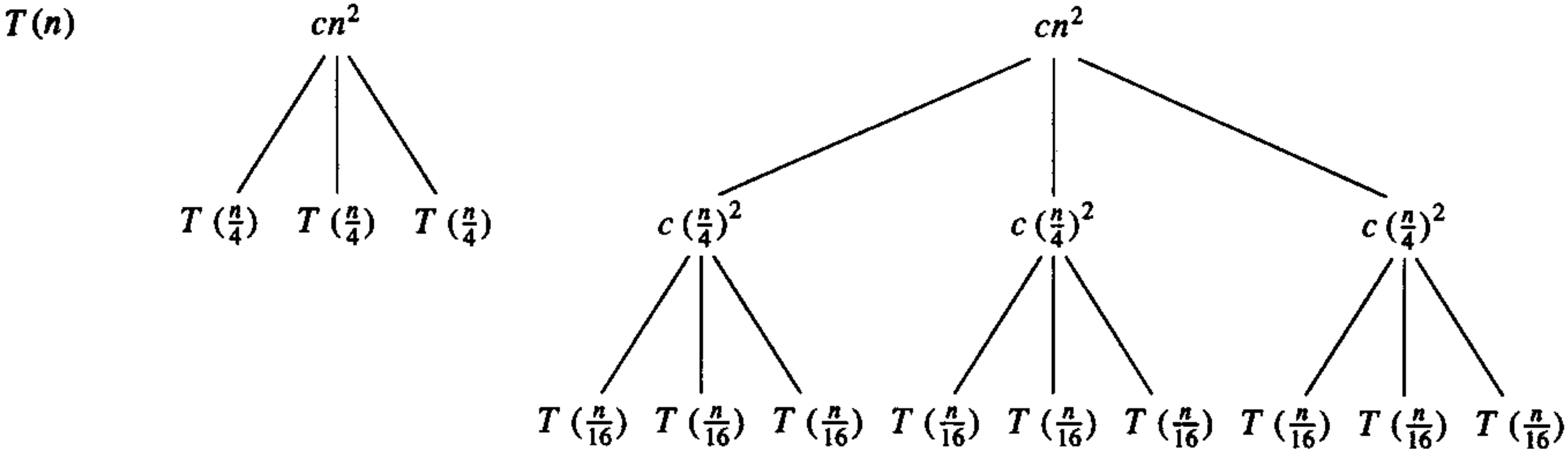
- Subproblem size at level i is: $n/2^i$
- Subproblem size hits 1 when $1 = n/2^i \Rightarrow i = \lg n$
- Cost of the problem at level i = $(n/2^i)^2$ No. of nodes at level $i = 2^i$
- Total cost:

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

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

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:

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

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

Example 2 - Substitution

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

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

$$\begin{aligned} T(n) &= 3T(n/4) + cn^2 \\ &\leq 3d\,(n/4)^2 + cn^2 \\ &= (3/16)\,d\,n^2 + cn^2 \\ &\leq d\,n^2 \qquad \text{if: } d \geq (16/13)c \end{aligned}$$

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

Example 3 (simpler proof)

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

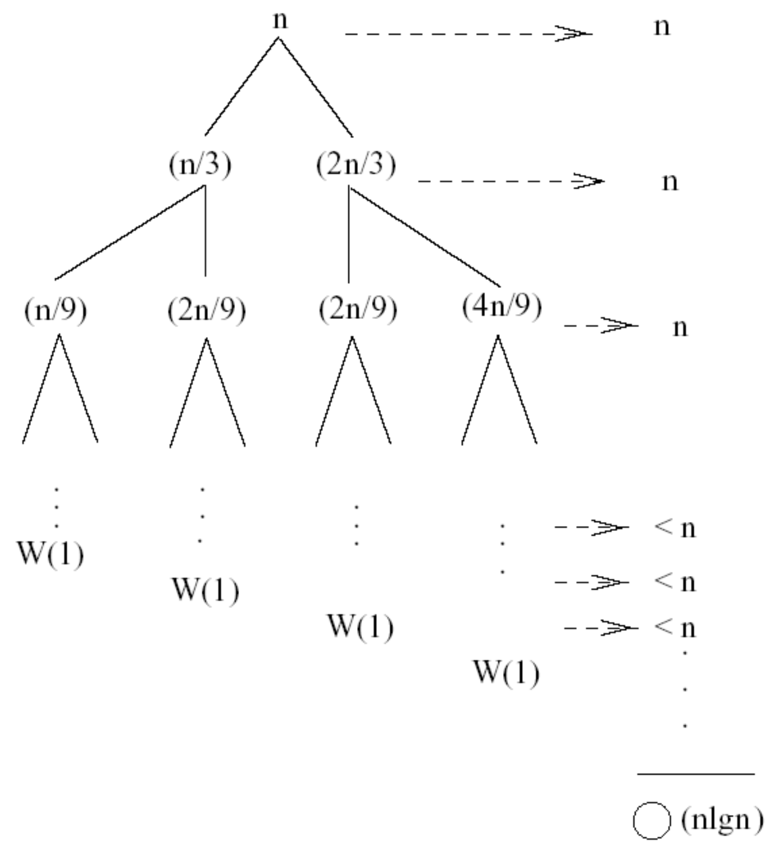
- The longest path from the root to a leaf is:

  $n \to (2/3)n \to (2/3)^2\, n \to \ldots \to 1$

- 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:

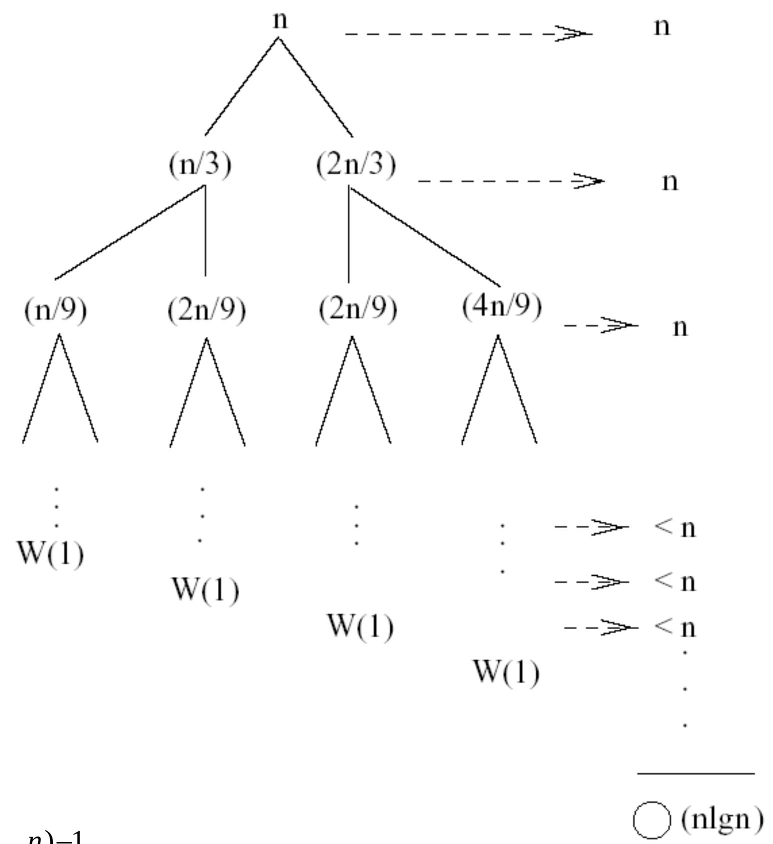
  $n \rightarrow (2/3)n \rightarrow (2/3)^2\, n \rightarrow \ldots \rightarrow 1$

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

  Try it out as an exercise!!

- $T(n) = O(nlgn)$

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 $f(n) > 0$

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, $a \geq 1$, $b > 1$, 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

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.

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

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

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

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:

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

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

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 $\lg n$

Readings