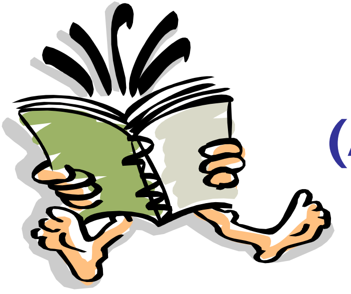
Analysis and Design of Algorithms

Recurrence Relations

Instructor: Morteza Zakeri

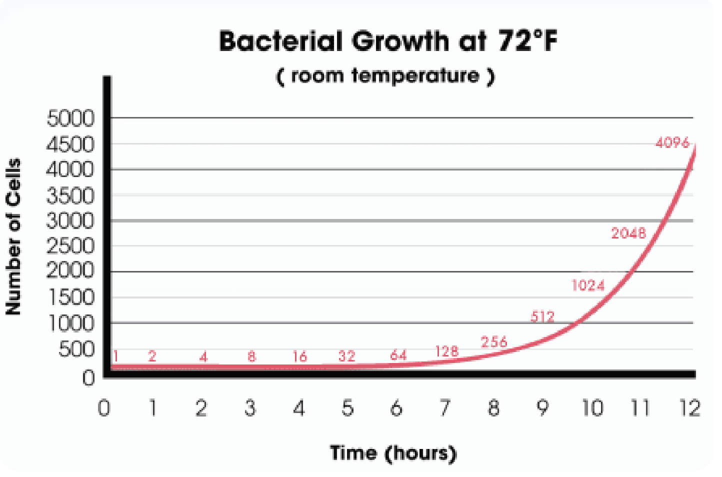
Modified and eXtended version of Slides from *George Bebis*



(Appendix A, Chapter 4)

Recurrence relations

- Many counting problems can be solved with recurrence relations
- Example:
  - The number of bacteria doubles every 2 hours. If a colony begins with 5 bacteria, how many will be present in n hours?
- Solution:
  - Let $a_n=2a_{n-1}$ where n is a positive integer with $a_0=5$



Recurrence relations

- A **recurrence relation** for the sequence $\{a_n\}$ is an equation that expresses an in terms of 1 or more of the previous terms of the sequence, *i.e.*, $a_0, a_1, \ldots, a_{n-1}$, for all integers n with $n \geq n_0$ where n_0 is a nonnegative integer
- A sequence is called a **solution** of a recurrence relation if its terms satisfy the recurrence relation

Example

- Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} - a_{n-2}$ for $n = 2, 3, 4, \ldots$ and suppose that $a_0 = 3$ and $a_1 = 5$, what are a_2 and a_3?
- Using the recurrence relation, $a_2 = a_1 - a_0 = 5 - 3 = 2$ and $a_3 = a_2 - a_1 = 2 - 5 = -3$

Example

- Determine whether the sequence $\{a_n\}$, where $a_n=3n$ for every nonnegative integer n, is a solution of the recurrence relation $a_n=2a_{n-1} – a_{n-2}$ for n=2, 3, 4, …
  - Suppose $a_n = 3n$ for every nonnegative integer n.
  - Then for n ≥ 2, we have $2a_n\text{-}1\text{-}a_n\text{-}2 = 2(3(n-1))-3(n-2) = 3n = a_n$.
  - Thus, {an} where $a_n = 3n$ is a solution for the recurrence relation

Modeling with recurrence relations

- Compound interest: Suppose that a person deposits \$10,000 in a savings account at a bank yielding 11% per year with **interest compounded** annually. How much will it be in the account after 30 years?
- Let P_n denote the amount in the account after n years.
- The amount after n years equals the amount in the amount after n-1 years plus interest for the n-th year, we see the sequence $\{P_n\}$ has the recurrence relation

$$P_n = P_{n-1} + 0.11P_{n-1} = (1.11)P_{n-1}$$

Modeling with recurrence relations

- The initial condition P_0=10,000, thus
- $P_1 = (1.11)P_0$
- $P_2 = (1.11)P_1 = (1.11)^2P_0$
- $P_3=(1.11)P_2 = (1.11)^3P_0$
- …
- $P_n = (1.11)P_{n-1} = (1.11)^nP_0$
- We can use mathematical induction to establish its validity

Modeling with recurrence relations

- We can use mathematical induction to establish its validity
- Assume $P_n=(1.11)^n 10{,}000$.
- Then from the recurrence relation and the induction hypothesis
  - $P_{n+1} = (1.11)P_n$
  - $= (1.11)(1.11)^n 10{,}000 = (1.11)^{n+1} 10{,}000$
  - $N = 30$, $P_{30} = (1.11)^{30} 10{,}000 = 228{,}922.97$

Recursion and recurrence

- A recursive algorithm provides the solution of a problem of size n in terms of the solutions of one or more instances of the same problem of smaller size
- When we analyze the complexity of a recursive algorithm, we obtain a recurrence relation that expresses the number of operations required to solve a problem of size n in terms of the number of operations required to solve the problem for one or more instance of smaller size.

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

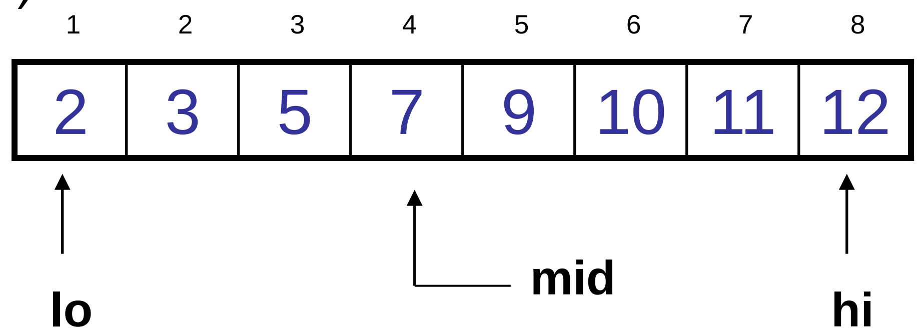
Example Recurrences

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

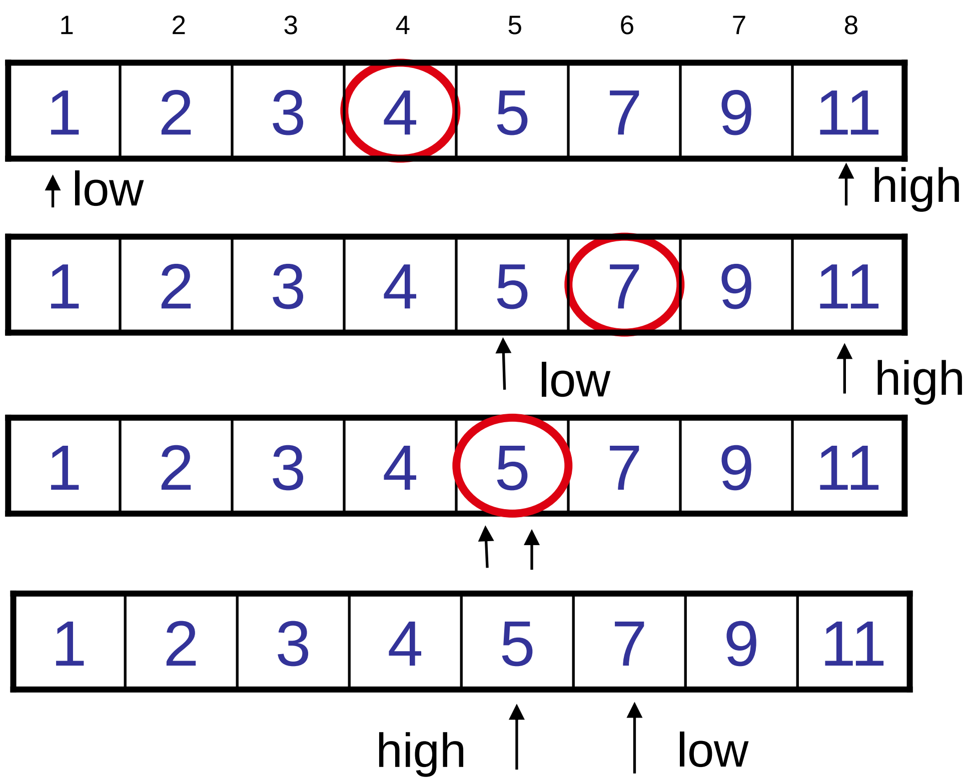
				5	6	7	8
1	2	3	4	5	7	9	11

mid = 6, A[mid] = x

Found!

Another Example

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



mid = 4, lo = 5, hi = 8

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

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

NOT FOUND!

Analysis of BINARY-SEARCH

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

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

Types of recurrence relations

- Linear *vs.* non-linear
  - A recurrence relation for a sequence S(n) is linear if the earlier values of S appearing in the definition occur only to the first power.

$$S(n) = f_1(n)S(n-1) + f_2(n)S(n-2) + \cdots + f_k(n)S(n-k) + g(n)$$

  - e.g., linear $a_n = na_{n-1} - 1$

    nonlinear $a_n = 1/(1 + a_{n-1})$

- Constant coefficient *vs.* variable coefficients
  - The recurrence relation has constant coefficients is the f_i's are all constants.
  - e.g., $a_n = na_{n-1} + (n-1)a_{n-2} + 1$

Types of recurrence relations

- First order *vs.* higher order
  - It is first-order if the n^{th} term depends only on term n−1.
  - e.g., second order recurrence relations:

linear	$a_n = a_{n-1} + 2a_{n-2}$
nonlinear	$a_n = a_{n-1}a_{n-2} + \sqrt{a_{n-2}}$

- Homogeneous *vs.* non-homogeneous
  - Recurrence relation is homogeneous if g(n)=0 for all n.
- **Linear first-order recurrence relations with constant coefficients** have the form:

$$S(n) = cS(n-1) + g(n)$$

Example of recurrence relations

- These are some examples of well-known linear recurrence equations

Recurrence relations	Initial values	Solutions
$F_n = F_{n-1} + F_{n-2}$	$a_1 = a_2 = 1$	Fibonacci number
$F_n = F_{n-1} + F_{n-2}$	$a_1 = 1, a_2 = 3$	Lucas Number
$F_n = F_{n-2} + F_{n-3}$	$a_1 = a_2 = a_3 = 1$	Padovan sequence
$F_n = 2F_{n-1} + F_{n-2}$	$a_1 = 0, a_2 = 1$	Pell number

Solving recurrences relations

- From mathematical induction
- Recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

- A sequence satisfying the recurrence relation in the definition is uniquely defined by this recurrence relation and the k initial conditions

$$a_0 = C_0, a_1 = C_1, \dots, a_{k-1} = C_{k-1}$$

- We can often solve a recurrence relation in a manner analogous to solving a **differential equations.**

Methods for Solving Recurrences

- Iteration method
  - Most simple method
- Characteristic equation
  - Mostly for Linear Recurrence Relations
- Substitution method
  - Mostly for Linear Recurrence Relations
- Recursion tree method
  - Mostly for Divide and Conquer Recurrence Relations
- Master theorem method
  - Mostly for Divide and Conquer Recurrence Relations

Methods for Solving Recurrences

- **Iteration method**
- Characteristic equation
- Substitution method
- Recursion tree method
- Master theorem 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 $n = 2^k$

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

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

$= \Theta(\lg n)$

Iteration Method – Example

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

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

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

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

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

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

… $= in + 2^i T(n/2^i)$

$= kn + 2^k T(1)$

$= n\lg n + nT(1) = \Theta(n\lg n)$

Methods for Solving Recurrences

- Iteration method
- **Characteristic equation**
- Substitution method
- Recursion tree method
- Master theorem method

The characteristic equation method

- Linear homogenous recurrence relations with constant coefficients

- Look for solutions of the form $a_n = r^n$ where r is a non-zero constant
- Since $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$
- So, $r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}$
- Divide both sides by r^{n-k},

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$$

- The sequence $\{a_n\}$ with $a_n = r^n$ is a solution <u>if and only if</u> r is a solution of this characteristic equation

Theorem 1

- Let c_1 and c_2 be real numbers. Suppose that $r^2 - c_1 r - c_2 = 0$ has two distinct roots r_1 and r_2 . Then the sequence $\{a_n\}$ is a solution of the recurrence relation

  $a_n = c_1\, a_{n-1} + c_2\, a_{n-2}$

  if and only if

  $a_n = \alpha_1 {r_1}^n + \alpha_2 {r_2}^n$ for $n = 0, 1, 2, \ldots$ where

  α_1 and α_2 are constants

Example

- Find solution for $a_n = a_{n-1} + 2a_{n-2}$ where $a_0 = 2, a_1 = 7$
- The characteristic equation is $r^2 - r - 2 = 0$, and the roots are $r_1 = 2$ and $r_2 = -1$
- Hence $a_n = \alpha_1 2^n + \alpha_2(-1)^n$
- Thus, $a_0 = \alpha_1 + \alpha_2 = 2, a_1 = 2\alpha_1 - \alpha_2 = 7$
- So, $\alpha_1 = 3, \alpha_2 = -1$
- Hence, $a_n = 3 \cdot 2^n - (-1)^n$

Example: Fibonacci numbers

- Recall $f_n = f_{n-1} + f_{n-2}$ and $f_0 = 0, f_1 = 1$
- The roots of the characteristic equation $r^2 - r - 1 = 0$ are $r_1 = (1 + \sqrt{5})/2$ and $r_2 = (1 - \sqrt{5})/2$. Thus $f_n = \alpha_1 (\frac{1+\sqrt{5}}{2})^n + \alpha_2 (\frac{1-\sqrt{5}}{2})^n$
- $f_0 = \alpha_1 + \alpha_2 = 0, f_1 = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)$
- Thus $\alpha_1 = 1/\sqrt{5}, \alpha_2 = -1/\sqrt{5}$
- Consequently, $f_n = \frac{1}{\sqrt{5}} (\frac{1+\sqrt{5}}{2})^n - \frac{1}{\sqrt{5}} (\frac{1-\sqrt{5}}{2})^n$

Proof: $\alpha_1 {r_1}^n + \alpha_2 {r_2}^n \rightarrow c_1 a_{n-1} + c_2 a_{n-2}$

- Show if $a_n = \alpha_1 {r_1}^n + \alpha_2 {r_2}^n$ (and r^2-$c_1 r - c_2 = 0$) then $\{a_n\}$ is the solution of the recurrence relation $a_n = c_1 a_{n-1} + c_2 a_{n-2}$
- $c_1 a_{n-1} + c_2 a_{n-2}$=$c_1(\alpha_1 {r_1}^{n-1} + \alpha_2 {r_2}^{n-1}) + c_2(\alpha_1 {r_1}^{n-2} + \alpha_2 {r_2}^{n-2}) =$ $\alpha_1 {r_1}^{n-2}(c_1 r_1 + c_2) + \alpha_2 {r_2}^{n-2}(c_1 r_2 + c_2)$= $\alpha_1 {r_1}^{n-2} {r_1}^2 + \alpha_2 {r_2}^{n-2} {r_2}^2$=$\alpha_1 {r_1}^n + \alpha_2 {r_2}^n = a_n$

Proof: $c_1 a_{n-1} + c_2 a_{n-2} \rightarrow \alpha_1 {r_1}^n + \alpha_2 {r_2}^n$

- Show if $a_n = c_1 a_{n-1} + c_2 a_{n-2}$ (and r^2-$c_1 r - c_2 = 0$) then $a_n = \alpha_1 {r_1}^n + \alpha_2 {r_2}^n$ for some α_1 and α_2
- From initial conditions: $a_0 = C_0$=$\alpha_1 + \alpha_2$
  $a_1 = C_1$=$\alpha_1 r_1 + \alpha_2 r_2$
- It follows, $\alpha_1 = \frac{C_1 - C_0 r_2}{r_1 - r_2}$, $\alpha_2 = \frac{C_0 r_1 - C_1}{r_1 - r_2}$ when $r_1 \neq r_2$
- Both $\{a_n\}$ and $\{\alpha_1 {r_1}^n + \alpha_2 {r_2}^n\}$ are solutions
- As there is a unique solution of a linear homogenous recurrence relation of degree 2 with the same initial conditions, these two must be the same

Methods for Solving Recurrences

- Iteration method
- Characteristic equation
- **Substitution method**
- Recursion tree method
- Master theorem method

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
    - $T(n) \le d\ g(n)$, for some $d > 0$ and $n \ge n_0$
  - Induction hypothesis: $T(k) \le d\ g(k)$ 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:

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

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

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

- 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:

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

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

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

Methods for Solving Recurrences

- Iteration method
- Characteristic equation
- Substitution method
- **Recursion tree method**
- Master theorem method

The recursion-tree method

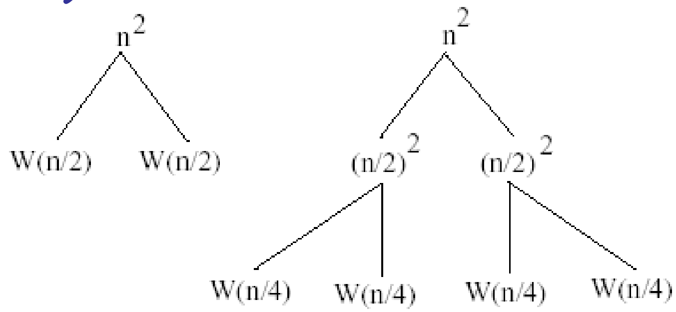
Convert the recurrence into a tree:

- 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

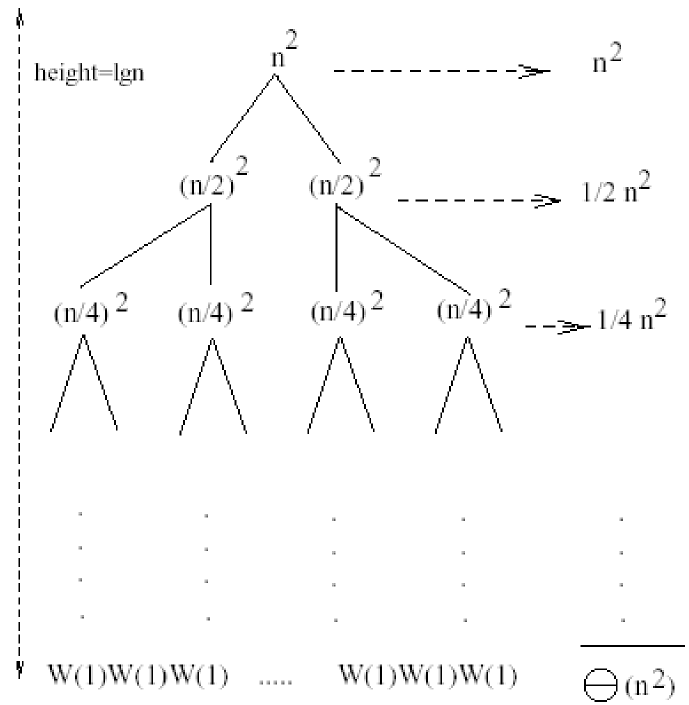
Example 1

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



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

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



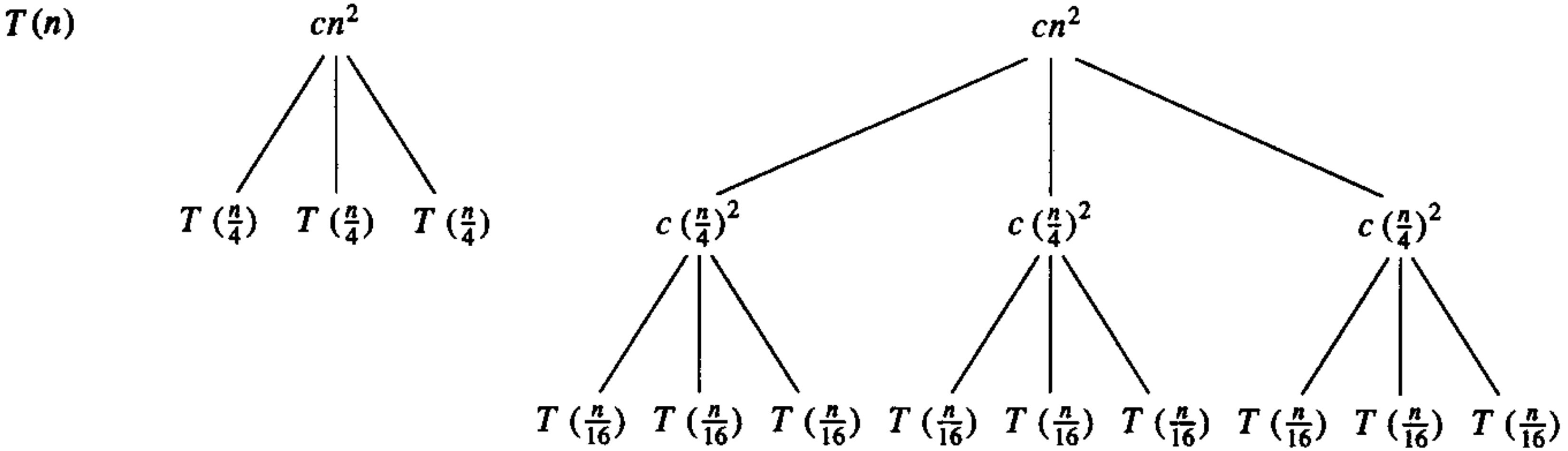
- Subproblem size at level i is: $n/2^i$
- Subproblem size hits 1 when $1 = n/2^i \Rightarrow i = \lg n$
- Cost of the problem at level i = $(n/2^i)^2$ No. of nodes at level i = 2^i
- Total cost:

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

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

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

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

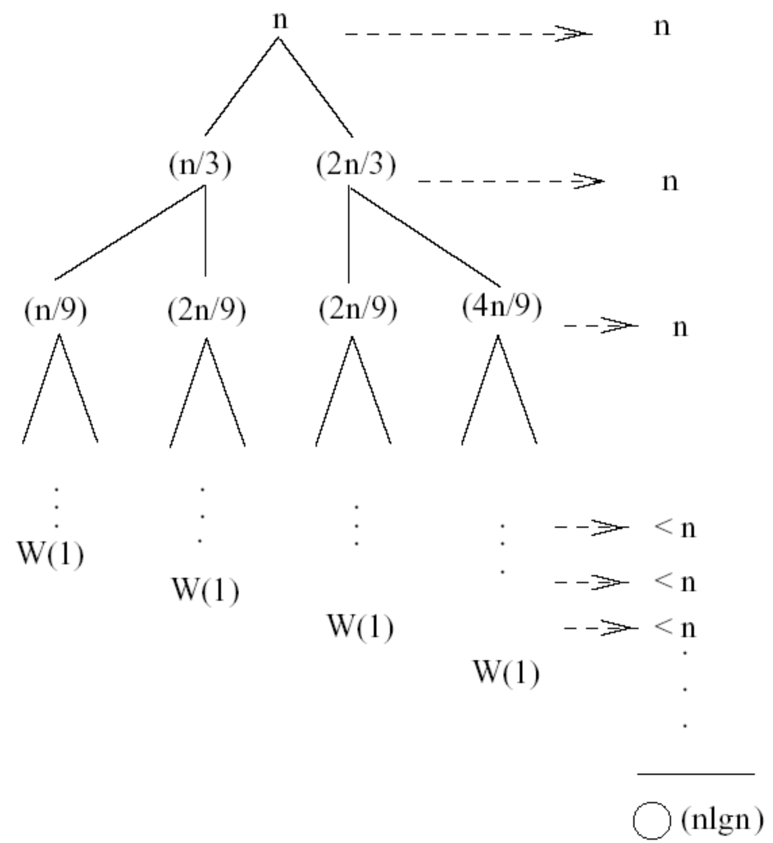
- Guess: $T(n) = O(n^2)$
  - Induction goal: $T(n) \le dn^2$, for some d and $n \ge n_0$
  - Induction hypothesis: $T(n/4) \le d\,(n/4)^2$
- Proof of induction goal:

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

- 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:

  $n \to (2/3)n \to (2/3)^2 n \to \dots \to 1$

- Subproblem size hits 1 when

  $1 = (2/3)^i n \Leftrightarrow i = \log_{3/2} n$

- Cost of the problem at level i = n

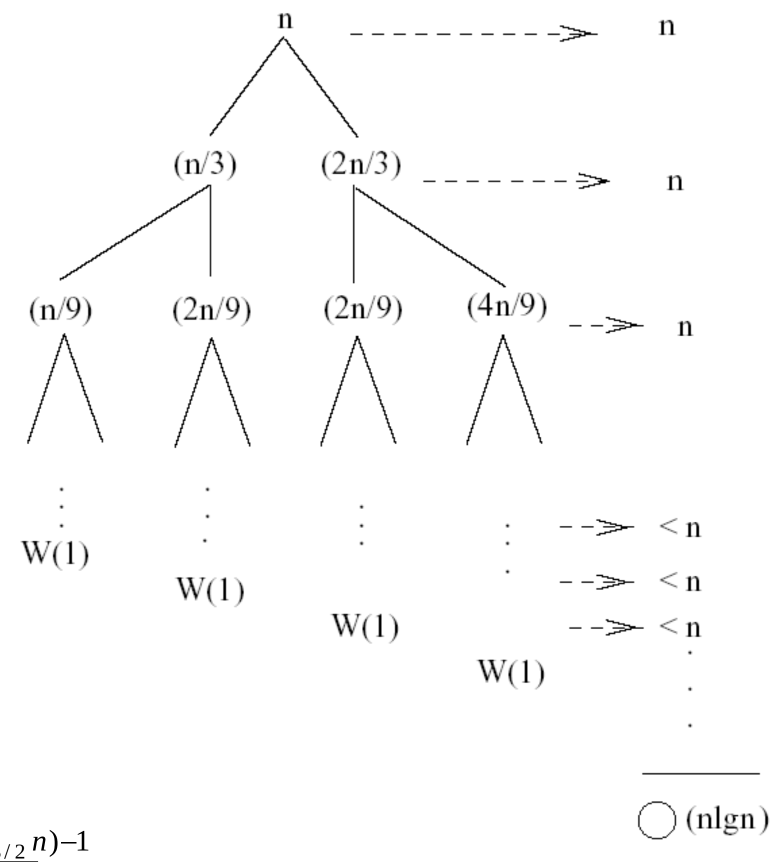
- Total cost:

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

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

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!!

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

Methods for Solving Recurrences

- Iteration method
- Characteristic equation
- Substitution method
- Recursion tree method
- **Master (theorem) method**

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:

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

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

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

- **Appendix A, Chapter 4**