Recurrences

Recurrences and Running Time

- Recurrences arise when an algorithm contains recursive calls to itself
- An equation or inequality that describes a function in terms of its value on smaller inputs.

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

- 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

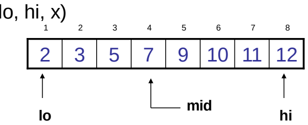
Example Recurrences

- $T(n) = T(n-1) + n$ $\quad\Theta(n^2)$
  - Recursive algorithm that loops through the input to eliminate one item
- $T(n) = T(n/2) + c$ $\quad\Theta(\lg n)$
  - Recursive algorithm that halves the input in one step
- $T(n) = T(n/2) + n$ $\quad\Theta(n)$
  - Recursive algorithm that halves the input but must examine every item in the input
- $T(n) = 2T(n/2) + 1$ $\quad\Theta(n)$
  - 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 ← ⌊(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!

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
- Master method
- Recursion tree 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

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

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

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

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

Assume $n = 2^k \rightarrow k = \lg n$

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

$= ck + T(1)$

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

$= \Theta(\lg n)$

9

Iteration Method – Example

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

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

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

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

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

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

$\ldots = in + 2^iT(n/2^i)$

$= kn + 2^kT(1)$

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

10

Methods for Solving Recurrences

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

11

The substitution method

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

12

Substitution method

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

Example: Binary Search

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

- Guess: $T(n) = O(\lg n)$
  - Induction goal: $\mathbf{T(n) \leq d \lg n}$, for some d and $n \geq n_0$
  - Induction hypothesis: $T(n/2) \leq d \lg(n/2)$
- Proof of induction goal:

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

$\leq d \lg n - \mathbf{d} + c$

$\mathbf{T(n) \leq d \lg n}$

$(d \lg n - d + c \leq d \lg n)$ if: $-d + c \leq 0, d \geq c$

Example 2

$$\mathbf{T(n) = T(n-1) + n}$$

- Guess: $T(n) = O(n^2)$
  - Induction goal: $\mathbf{T(n) \leq c\, n^2}$, for some c and $n \geq n_0$
  - Induction hypothesis: $T(n-1) \leq c(n-1)^2$
- Proof of induction goal:

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

$\leq cn^2 - (2cn - c - n)$

$\mathbf{T(n) \leq cn^2}$ (i.e. $cn^2 - (2cn - c - n) \leq cn^2$)

if: $2cn - c - n \geq 0 \Rightarrow c \geq n/(2n-1) \Rightarrow c \geq 1/(2 - 1/n)$

- **For $n \geq 1 \Rightarrow 2 - 1/n \geq 1 \Rightarrow$ any $c \geq 1$ will work**

Methods for Solving Recurrences

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

Master's method

- 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

- 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 **$n^{\log_2 2}$** (**$n^{\log_b a}$**) with **$f(n) = n$**

➔ $f(n) = \Theta(n)$ **(Case 2)**

➔ **$T(n) = \Theta(n\lg n)$**

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

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

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

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

➔ **$f(n) = \Omega(n^{1+\varepsilon})$ (Case 3)**

➔verify regularity condition $(a\, f(n/b) \le c\, f(n))$

$2\, n^2/4 \le c\, n^2$ ➔ $c = ½$ is a solution $(c<1)$

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

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 $\lg n$

Master's method

- 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

Why $n^{\log_b a}$?

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

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

$$\underbrace{a^2T\left(\frac{n}{b^2}\right)}_{a^3T\left(\frac{n}{b^3}\right)}$$

$$\vdots$$

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

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

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

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

25

Methods for Solving Recurrences

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

26

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

27

Common Summations

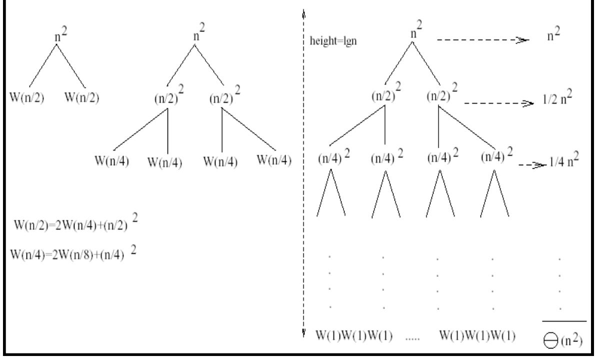
- Arithmetic series: $\sum_{k=1}^{n} k = 1 + 2 + ... + n = \frac{n(n+1)}{2}$
- Geometric series: $\sum_{k=0}^{n} x^k = 1 + x + x^2 + ... + x^n = \frac{x^{n+1}-1}{x-1} (x \neq 1)$
  - Special case: |x| < 1: $\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$
- Harmonic series: $\sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + ... + \frac{1}{n} \approx \ln n$
- Other important formulas: $\sum_{k=1}^{n} \lg k \approx n \lg n$

$$\sum_{k=1}^{n} k^p = 1^p + 2^p + ... + n^p \approx \frac{1}{p+1} n^{p+1}$$

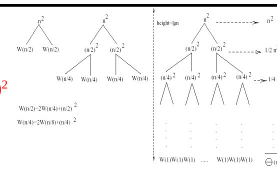
28

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:



$$W(n) = \sum_{i=0}^{\lg n-1} \frac{n^2}{2^i} + 2^i * W(1) = \sum_{i=0}^{\lg n-1} \frac{n^2}{2^i} + 2^{\lg n} * W(1)$$

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

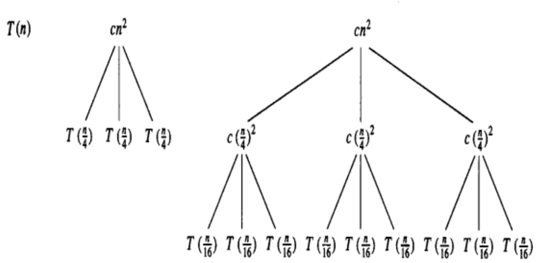
$$= n^2 \frac{1}{1-\frac{1}{2}} + O(n) = 2n^2$$

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

30

Example 2

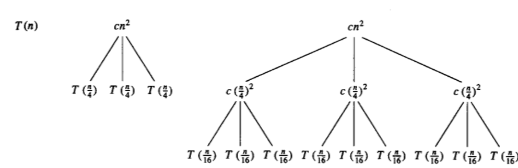
$T(n) = 3T(n/4) + cn^2$



31

Example 2

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



- 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:

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

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

32

Changing variables

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

– 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

34

Substitution method more examples

35

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:

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

$= cn\lg n - cn + n$

$T(n) \le cn\lg n$ (**i.e. $cn\lg n - cn + n \le cn\lg n$**)

if: $-cn + n \le 0 \Rightarrow$ **$c \ge 1$**

36

Example 4

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

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

37