DSAII: Recurrences

Sattiraju Prabhakar
CS560: Classes_13_14
Wichita State University

Topics

- Introduction
- The Substitution Method
- Recursion-Tree Method
- Master Method

Introduction

Recurrences

- *Definition:*
  - A recurrence is an **equation** on **inequality** that describes a function in terms of **its value on smaller inputs**.
- Example: Worst case running time T(n) of Merge-Sort

$$T(n) = \begin{cases} \Theta(1) \text{.....if} & n=1 \\ 2T(n/2) + \Theta(n) \text{.....if} & n>1 \end{cases}$$

The Problem

- We would like to solve the recurrence equations
  - We would like to obtain **asymptotic Θ or O bounds** on the solution.
- Example, for Merge-Sort recurrence equation, the solution is:
  - T(n) = Θ(n lg n)

Three methods to find solutions

- Substitution Method:
  - We guess a bound
  - Then prove that the guess is correct by using mathematical induction
- Recursion-Tree Method:
  - Converts recurrence into a recursion tree
  - We compute the total cost
- Master Method:
  - This provides solutions for recurrence equations of the form:
    - T(n) = aT(n/b) + f(n)

Simplifying Assumptions

- We know n in T(n) is an integer
- We ignore floor and ceiling
- We ignore the functional values for boundary conditions.
  - Example:

$$T(n) = \begin{cases} \Theta(1) \text{.....if } n=1 \\ 2T(n/2) + \Theta(n) \text{.....if } n>1 \end{cases}$$

  - We simply write as: T(n) = 2T(n/2) + Θ(n)

The Substitution Method

The Method

- Step1:
  - Guess a solution: We call this hypothesis
- Step2:
  - Use mathematical induction to find the constants (c, n_0) for the hypothesis
  - Show the hypothesis forms the boundary for T(n)

Some Cautionary Notes

- There is no general way to guess correct solutions (hypotheses) to recurrences
- Guessing takes experience
  - You can guess a solution based on similarity:
    - Example:
      - $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$, is similar to
      - $T(n) = 2T(\lfloor n/2 \rfloor) + n$
        - » Hence we can guess their solutions to be similar

Example of Substitution Method: Part 1

- Problem: Solve the following recurrence equation:
  - $T(n) = 2T(\lfloor n/2 \rfloor) + n$
- Solution:
  - Step1: Guessing the Solution:
    - We guess, $T(n) = O(n \lg n)$

Substitution Example: Part 2

- Step2:
  - Our Objective:
    - show that $T(n) \leq cn \lg n$, for some $c > 0$.
  - Inductive Assumption: k^{th} case
    - $T(n) \leq cn \lg n$, holds good for $\lfloor n/2 \rfloor$
      - That is, $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor$
  - Showing that hypothesis holds good for k+1st case (Inductive case):
    - Substituting k^{th} case value of hypothesis into recurrence equation
    - $T(n) \leq 2(c\lfloor n/2 \rfloor \lg (\lfloor n/2 \rfloor)) + n$
    - $\leq cn\lg(n/2) + n$
    - $= cn \lg n - cn \lg 2 + n$
    - $= cn \lg n - cn + n$
    - $\leq cn \lg n$, for $c \geq 1$

Substitution Example: Part 3

- Base Case:
  - To show that our hypothesis holds good for base case (for induction)
  - Method: Find a boundary condition of recurrence for base case of induction
    - The boundary condition (of recurrence): n = 1, T(1) = 1
      - $T(1) = 2T(\lfloor 1/2 \rfloor) + 1 = 1$
    - Base case of inductive hypothesis
      - Use n = 1
      - Substituting in the hypothesis:
        - » $T(1) \leq cn \lg n = c\ 1 \lg 1 = 0$
  - Problem: Thus boundary condition of $n_0 = 1$ cannot be used for base case of induction

Substitution Example: Part 4

- Base Case (Continued):
  - Solution: We need to find another n_0 for which boundary condition can be used in base case of induction
- We would like to start with n_0, that does not use T(1) in the recurrence equation
  - $T(2) = 2T(\lfloor 2/2 \rfloor) + 2 = 2T(1) + 2 = 4$,
  - If we use this equation, then inductive hypothesis will be false
  - Conclusion: we will not use $n_0 = 1$
- Now let us try for n > 3
- For n = 4, the hypothesis seems to work for T(n).
  - $T(4) = 2T(\lfloor 4/2 \rfloor) + n = 2T(2) + 4 = 8$
  - That is, $T(4) = 2T(n_0) + n$
  - $T(6) = 2T(\lfloor 6/2 \rfloor) + n = 2T(3) + n$
  - They form the base cases
  - So we can take $n_0 = 2, 3$, for boundary conditions
- To complete the inductive proof, we need to find some c that works for the upper bound.

Substitution Example: Part 5

- To find c:
  - We use base case and boundary conditions
- We know, T(1) = 1
  - $T(2) = 2T(1) + 2 = 4$
    - $T(2) \leq c2 \lg 2 = 2c$
  - $T(3) = 2T(1) + 3 = 5$
    - $T(3) \leq c3 \lg 3$
- We need to satisfy the following:
  - $T(n) \leq c\ n \lg n$
- Both can be satisfied if, $c \geq 2$
- Hence our hypothesis is correct: cn lgn, for $c \geq 2$

Guessing a solution - 1

- There are no general principles for guessing
- Some heuristics for guessing
  - Method1: By Similarity of recurrence equations
    - Example: What is the solution for
      - $T_1(n) = 2T_1(\lfloor n/2 \rfloor + 17) + n$
    - Solution: It is O(n lg n)
      - Because $T(n) = 2T(\lfloor n/2 \rfloor) + n$
      - Has solution, O(n lg n)
      - By similarity, $T_1(n)$ has the similar solution

Guessing a solution 2

- Method2:
  - Guess **loose** upper and lower bounds on T(n)
  - Example:
    - To guess the bounds for $T(n) = 2T(\lfloor n/2 \rfloor) + n$
    - Lower Bound, $T(n) = \Omega(n)$
      - This is because equation for T(n) contains n term
    - Upper Bound, $T(n) = O(n^2)$
      - This is because, there are no n terms of higher order than n^2.
  - Then we refine the bounds
    - Lower the upper bound and increase the lower bound, until we reach a tight solution
      - $T(n) = \Theta(n \lg n)$

Induction requires rigorous mathematical application

- Example:
  - Solve $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
  - Step1: Guess the solution:
    - $T(n) = O(n)$
  - Step2: Proof by Induction
    - To show that $T(n) \leq cn$
    - $T(n) \leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1$
      - $= cn + 1$
    - This **does not imply** $T(n) \leq cn$
- ***Conclusion***: We cannot use inductive hypothesis because our hypothesis does not have "1" in it

Overcoming the Problem of Using Inductive Hypothesis

- We subtract a lower order term from previous guess.
- Previous Example:
  - Our new guess: $T(n) \leq cn - b$
    - where $b \geq 0$
  - $T(n) \leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1$
    - $= cn - 2b + 1$
    - $\leq cn - b$, for $b \geq 1$
- ***Conclusion***: We can prove something stronger by assuming something **stronger for smaller values**

Some Common Errors in Asymptotic Notation

- Let us say we want to prove
  - For, $T(n) = 2(\lfloor n/2 \rfloor) + n$,
    - Solution is $T(n) = O(n)$
  - We guess, $T(n) \leq cn$
  - Proof:
    - $T(n) \leq 2(c\lfloor n/2 \rfloor) + n$
    - $\leq cn + n$
    - $= O(n)$ **Wrong!!!!**
- **This is because mathematical induction requires proving the exact hypothesis: $T(n) \leq cn$**

Finding similarities between solutions by changing variables

- This by changing variables, as in algebra
- Consider, $T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \lg n$
- Simplification,
  - Let $m = \lg n$, (renaming)
    - $T(2^m) = 2T(2^{m/2}) + m$ //Ignore rounding errors
  - Let $S(m) = T(2^m)$
    - $S(m) = 2S(m/2) + m$
  - This recurrence has the solution
    - $S(m) = O(m \lg m)$
  - $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

Recursion-Tree Method

Recursion Tree

- Limitations of Substitution Method:
  - Sometimes, it is difficult to come up with good guesses
- Recursive Tree allows us to make good guesses
  - Method: The recursive trees give a simple mechanism to compute the total cost

Cost Distribution in a Tree

- Each node in the recursive tree represents the cost for a subproblem
- All the nodes at a level represent the cost of solving a problem at that level
- The total cost is it sum of costs at all levels.
- Assumption: We will simplify or round off T(n) wherever possible.

Example

- Find a good guess for
  - $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$
  - Let us say we want to find the upper bound
- Simplification:
  - We drop the floors and ceilings
    - $T(n) = 3T(n/4) + \Theta(n^2)$
  - We replace the bounds
    - $T(n) = 3T(n/4) + cn^2$
  - Assumption: n is an exact power of 4.

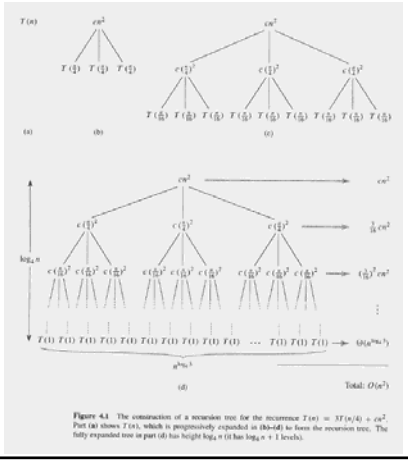

Figure 4.1 The construction of a recursion tree for the recurrence $T(n) = 3T(n/4) + cn^2$. Part (a) shows $T(n)$, which is progressively expanded in (b)–(d) to form the recursion tree. The fully expanded tree in part (d) has height $\log_4 n$ (it has $\log_4 n + 1$ levels).

Computing Node Costs: Begin to build a recursion tree

- See figure 4.1a, b
- Here each of the T(n/4) is the cost for each sub problem
- The cn^2 – includes the divide and combining cost.
  - This is for the root node level (depth 0).

Computing Node Costs: Expanding tree by a level

- See figure 4.1c
- Each T(n/4) is expanded
- Substituting (n/4) in our T(n) equation
  - $T(n/4) = 3T(n/(4*4)) + c(n/4)^2$

Computing Node Costs: Boundary Conditions

- See figure 4.1d
- The boundary condition is n = 1, that is when the tree terminates.
  - The subproblem size is (at depth i):
    - $n/4^i = 1$
    - $n = 4^i$
    - $\log_4 n = i \log_4 4$
    - $\log_4 n = i$
- The number of levels is: $(\log_4 n) + 1$

Cost at Each Level of Tree

- Observations:
  - Each level has three times more nodes than the level above
    - The number of nodes at depth ***i*** is: 3^i
  - The subproblem size reduces by ¼(Except the last level at depth $\log_4 n$)
    - The cost of a node at depth ***i*** is: $c(n/4^i)^2$
- Total Cost at depth ***i*** is (except at depth $\log_4 n$) :
  - $3^i * c(n/4^i)^2 = (3/16)^i cn^2$

Cost at depth $\log_4 n$

- The total number of nodes in the last level: 3^i, where $i = \log_4 n$
  - $3^{\log_4 n} = x$ = number of nodes
  - $\log_4 3 * \log_4 n = \log_4 x$
  - $\log_4 x = (\log_4 3) * \log_4 n$
  - $\log_4 x = \log_4 n^{\log_4 3}$
  - $x = n^{\log_4 3}$
  - $3^{\log_4 n} = n^{\log_4 3}$ (substituting value of x)

Cost at the level of depth $\log_4 n$

- The cost of each node: T(1)
- Total cost at the last level: $n^{\log_4 3}$
- $n^{\log_4 3} * T(1)$
  - $= \Theta(n^{\log_4 3})$

Total cost over entire tree

$$T(n) = cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \ldots + \left(\frac{3}{16}\right)^{\log_4(n-1)} cn^2 + \Theta(n^{\log_4 3})$$

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

$$= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})$$

See next page

Simplifying Total Cost

According to summation of geometric sequences

$$\sum_{i=0}^{k} x^i = \frac{x^{k+1} - 1}{x - 1}$$

When summation is infinite and $|x| < 1$, we have infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}$$

Simplifying Total Cost

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

$$< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

For upper bound

$$= \frac{1}{1 - (3/16)} cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) = O(n^2)$$

Proving the Guess to be Correct Using Substitution Method

- Guess: $O(n^2)$ is an upper bound for $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$
  - Theorem: $T(n) \le dn^2$, $d > 0$
- Proof:
  - $T(n) \le 3T(\lfloor n/4 \rfloor) + cn^2$
    - $\le 3d(\lfloor n/4 \rfloor)^2 + cn^2$
    - $\le 3d(n/4)^2 + cn^2$
    - $= (3/16)dn^2 + cn^2$
    - $\le dn^2$, $d \ge (16/13)c$

Another Example

- T(n) = T(n/3) + T(2n/3) + O(n)
- See next page

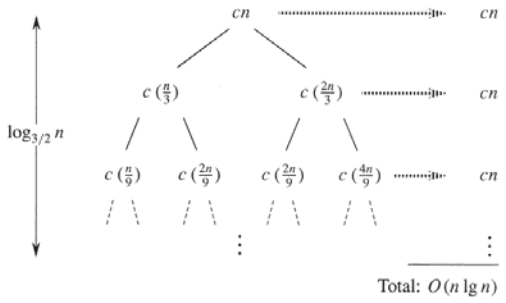

Figure 4.2 A recursion tree for the recurrence $T(n) = T(n/3) + T(2n/3) + cn$.

Height of the Tree

- The root node has value *cn*
- The leaf node has value *c1*
- The (longest) progression is:
  - n → (2/3)n → $(2/3)^2$n → ..→1
- Total number of elements in this series:
  - Let $(2/3)^k$n = 1 , the last item in the series (k^{th} level is the last level)
  - $(2/3)^k = 1/n \rightarrow (3/2)^k = n$
  - $k \log_{3/2}(3/2) = \log_{3/2} n \rightarrow k = \log_{3/2} n$
- The depth of the tree: $\log_{3/2} n$

Total Cost

- Computing Cost at Leaf Nodes Level:
  - Total Number of Leaves (for complete binary tree): for tree of height $\log_{3/2}$n
    - $2^{\log_{3/2} n} = n^{\log_{3/2} 2}$
    - This is because the binary tree with depth k has 2^k leaf nodes.
  - Total Cost of leaf nodes: c1 * $n^{\log_{3/2} 2}$
  - log3/22 = 1.7 (approx.), $n^{\log_{3/2} 2} = n^{1.7} = n * n^{0.7}$
- Simplifying the cost:
  - lg 2 = 0.7, lg 10 = 2.3, lg 100 = 4.6, lg 1000 = 6.9
  - $2^{0.7} = 1.6$, $10^{0.7} = 5.1$, $100^{0.7} = 25.1$, $1000^{0.7} = 125$
  - Though $n^{\log_{3/2} 2} > \lg n$,
    - Typically all the nodes in the binary tree are not present
    - $n^{\log_{3/2} 2} < n \lg n$
- Conclusion: The total cost of all leaves: O(n lg n)
- The total cost of the tree is O(n lg n)
  - Total cost (assuming last level has same cost as others)

Proof - 1

- Theorem, for the recurrence equation:
  - $T(n) = T(n/3) + T(2n/3) + cn$, the solution is:
  - $T(n) \leq d\ (n \lg n),\ d \geq 0$
- See proof next page:

Proof - 2

- $T(n)$
  - $\leq T(n/3) + T(2n/3) + cn$
  - $\leq d(n/3) \lg(n/3) + d(2n/3) \lg(2n/3) + cn$
  - $\leq dn \lg n$

Proof of Correctness by Substitution

- Our guess was: $T(n) \leq cn \lg n$
- $T(n) \leq T(n/3) + T(2n/3) + cn$
  - $\leq d(n/3) \lg (n/3) + d(2n/3) \lg (2n/3) + cn$
  - $=(d(n/3) \lg n - d(n/3) \lg 3) + d(2n/3) \lg n - d(2n/3) \lg (3/2)) + cn$
  - $dn \lg n - d((n/3) \lg 3 + (2n/3) \lg (3/2)) + cn$
  - $dn \lg n - d((n/3) \lg 3 + (2n/3) \lg 3 - (2n/3) \lg 2) + cn$
  - $dn \lg n - dn(\lg 3 - 2/3) + cn$
  - $\leq dn \lg n$
- This inequality is true if $d \geq c/(\lg 3 - (2/3))$

The Master Method

About the method

- Applicable to recurrences of the form:
  - T(n) = aT(n/b) + f(n), a ≥ 1, b > 1
    - f(n) is asymptotically positive function
    - f(n): Describes the cost of dividing and combining solutions to subproblems.
- We make simplifying assumptions as before
  - Like rounding off
  - Ignoring the floors and ceilings
- Master method depends upon master theorem

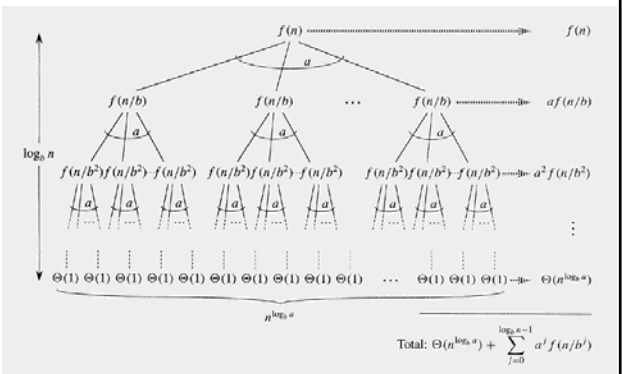

Figure 4.3 The recursion tree generated by $T(n) = aT(n/b) + f(n)$. The tree is a complete a-ary tree with $n^{\log_b a}$ leaves and height $\log_b n$. The cost of each level is shown at the right, and their sum is given in equation (4.6).

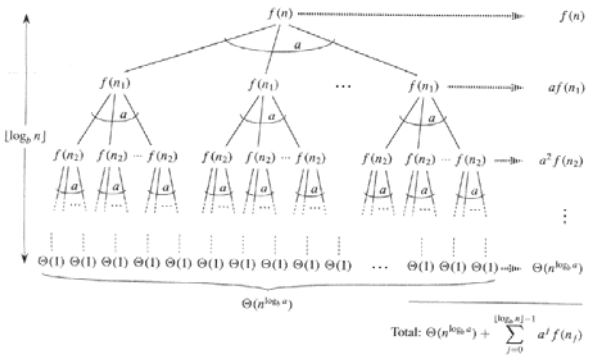

Figure 4.4 The recursion tree generated by $T(n) = aT(\lfloor n/b \rfloor) + f(n)$. The recursive argument n_j is given by equation (4.12).

The Master Theorem - 1

- Let a ≥1 and b >1 be constants
- Let f(n) be a function
- Let T(n) be defined on the nonnegative integers by the recurrence
  - T(n) = aT(n/b) + f(n)

The Master Theorem - 2

- The asymptotic bounds for T(n):
  - If $f(n) = O(n^{\log_b(a-\varepsilon)})$, for $\varepsilon > 0$
    - Then $T(n) = \Theta(n^{\log_b a})$
  - If $f(n) = \Theta(n^{\log_b a})$
    - Then $T(n) = \Theta(n^{\log_b a} \lg n)$
  - If
    1. $f(n) = \Omega(n^{\log_b a + \varepsilon})$, $\varepsilon > 0$
    2. $af(n/b) \leq cf(n)$, $c < 1$, large n
       - Then $T(n) = \Theta(f(n))$

Master Theorem Interpretation

- We are comparing in three cases
  - f(n) and $n^{\log_b a}$
- The solution to the recurrence equation
  - Larger $\{f(n), n^{\log_b a}\}$
  - Case1: $n^{\log_b a}$ is larger
    - Solution: $\Theta(n^{\log_b a})$
  - Case3: f(n) is larger
    - Solution: $\Theta(f(n))$
  - Case2: Functions are of the same size
    - Solution: We multiply by lg n

Method

- Method: To solve a given recurrence equation, we determine which case is applicable
- Applicability: The difference in between $\{f(n), n^{\log_b a}\}$ should be polynomial.
  - Case1: f(n) should be smaller by n^{ε}, $\varepsilon > 0$
  - Case3: f(n) must be polynomially larger than $n^{\log_b a}$

Example1: Condition1

- Example: $T(n) = 9T(n/3) + n$
  - Here, $a = 9$, $b = 3$, $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Case 1 applies, $f(n) = O(n^{\log_3 9 - \varepsilon})$, $\varepsilon = 1$
  - We conclude, $T(n) = \Theta(n^2)$

Example2: Condition2

- $T(n) = T(2n/3) + 1$
  - $a = 1$, $b = 3/2$, $f(n) = 1$
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
  - Case 2 applies, $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$
  - Solution, $T(n) = \Theta(\lg n)$

Example3: Condition3

- $T(n) = 3T(n/4) + n \lg n$
- $a = 3$, $b = 4$, $f(n) = n \lg n$
- $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$
- Since $f(n) = \Omega(n^{\log_4 3+\varepsilon})$, $\varepsilon \approx 0.2$
  - And regularity condition: $af(n/b) \leq cf(n)$
  - $af(n/b) = 3f(n/4) = 3(n/4) \lg(n/4) \leq (3/4) n \lg n = c f(n)$, for $c = ¾$
- Hence condition 3 is satisfied
- Hence, $T(n) = \Theta(f(n)) = \Theta(n \lg n)$

Example4: All conditions Fail

- $T(n) = 2T(n/2) + n \lg n$
- $a = 2$, $b = 2$, $f(n) = n \lg n$, $n^{\log_b a} = n$
- Here, $n = O(f(n)) = O(n \lg n)$
- But $\lg n < n^{\varepsilon}$, for any $\varepsilon < 1$
- Hence, we cannot apply the Master's theorem

Tutorial

- Using the Master method, give tight asymptotic bounds for the following recurrences
- $T(n) = 4T(n/2) + n$
- $T(n) = 4T(n/2) + n^2$
- $T(n) = 4T(n/2) + n^3$