CS 5321: Advanced Algorithms – Analysis Using Recurrence

Dr. Ali Ebnenasir
Department of Computer Science
Michigan Technological University

Acknowledgement

- Eric Torng
- Moon Jung Chung
- Charles Ofria

Outline

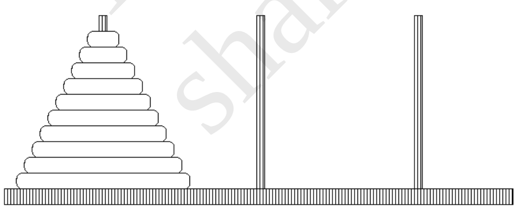
- Recursive algorithms vs. recurrence relations
- Specifying recurrence relations
- Solution techniques
  - Substitution method
  - Recursion-tree
  - Master theorem
  - Characteristic equations

Example: Towers of Hanoi

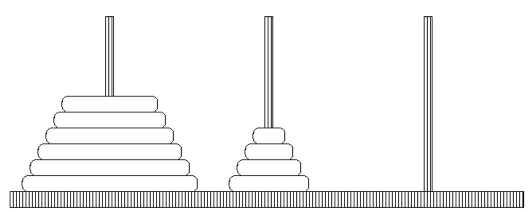
Problem Definition:
- Three pegs: A,B,C
- n disks
- Move n disks from A to B using C as a working peg

- *Constraints*:
  - One disk at a time
  - A larger disk cannot be on top of a smaller disk

The Towers of Hanoi

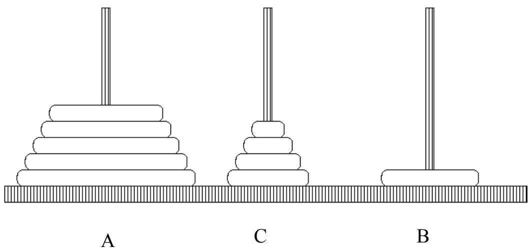


What if we knew we could solve *part* of the problem?

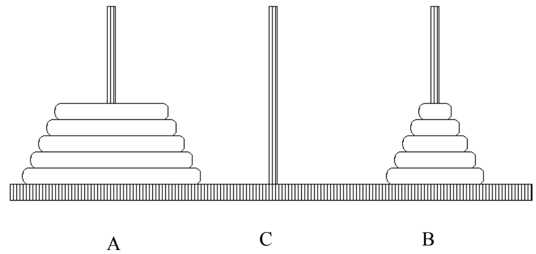


Assume we can move k (in this case, 4) different rings

Can we do one better?



Solved for one more!



Recurrence Relation

- T(n): minimum number of moves needed for n disks
- T(n-1): minimum number of moves needed for n-1disks
- What is the relation between these two?
  - Try to calculate the number of moves from small values of n

Where do recurrence relations come from?

- Analysis of a divide and conquer algorithm
  - Towers of Hanoi, Merge Sort, Binary Search
- Analysis of a combinatorial object
  - up-down permutations
- **This is the key analysis step you should master**
- Use small cases to check correctness of your recurrence relation

Can recurrence relations be solved?

- No general procedure for solving recurrence relations is known, which is why it is an art.
- **However, linear, finite history, constant coefficient recurrences always can be solved**
- Example: $a_n = 2a_{n-1} + 2a_{n-2} + 1$; $a_1 = 1$; $a_2 = 1$
  - degree = 1
  - history = 2
  - coefficients = 2, 2, and 1

Recurrence Formulas

- $T(n) = a\,T(n/b) + f(n)$
- Divide the problem into a subproblems
  - The size of each subproblem is n/b
- The function f(n) often represents the cost of dividing the problem to subproblems and then composing the solutions of subproblems; f(n) = D(n) + C(n)

Solution Techniques:
Substitution Method

Substitution Method

1. **Guess** a solution
   - Try back-substituting until you know what is going on
   - Draw a recursion tree

2. Use **induction** to verify your guess

Recursion *and* Mathematical Induction

In both, we have general and boundary conditions:

The **general** conditions break the problem into smaller and smaller pieces.

The **initial** or **boundary** condition(s) terminate the recursion.

Both take a **Divide and Conquer** approach to solving mathematical problems.

First Step

- Using the base case and the recursive case, calculate small values
- Use these values to help guess a solution
- Use induction to verify the correctness of your solution

Guessing a Solution

We can use mathematical induction to prove that a general function solves for a recursive one.

$T_n = 2T_{n-1} + 1$; $T_0 = 0$

n	= 0	1	2	3	4	5	6	7	8
T_n	= 0	1	3	7	15	31	63	127	255

Guess what the solution is?

Prove by Induction

Prove: $T_n = 2^n - 1$ by induction:

1. Base Case: n=0: $T_0 = 2^0 - 1 = 0$
2. Inductive hypothesis: assume $T_n = 2^n - 1$ for $n \geq 0$
3. Inductive Step: Show $T_{n+1} = 2^{n+1} - 1$ for $n \geq 0$

$$T_{n+1} = 2T_n + 1$$
$$= 2(2^n - 1) + 1$$
$$= 2^{n+1} - 1$$

Substitution Method Drawbacks

- Sometimes it is difficult to guess a solution
- We need systematic approaches for coming up with a guess
  - Back-substitution
  - Recursion tree

Back-Substitution

Example: $T(n) = 3T(\lfloor n/4 \rfloor) + n$, $T(1) = 1$

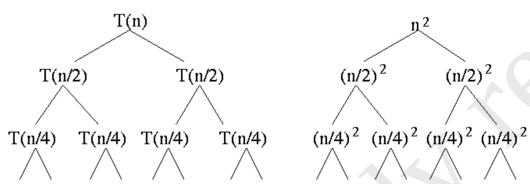
$$= 3(3T(\lfloor n/16 \rfloor) + n/4) + n$$
$$= 9T(\lfloor n/16 \rfloor) + 3n/4 + n$$

$$= 9(3T(\lfloor n/64 \rfloor) + n/16) + 3n/4 + n$$
$$= 27T(\lfloor n/64 \rfloor) + 9n/16 + 3n/4 + n$$

$$= n \cdot \sum_{i=0}^{?} \left(\frac{3}{4}\right)^i \leq \frac{1}{1-3/4} n = 4n$$

Recursion Trees

$T(n) = 2\,T(n/2) + n^2$, $T(1) = 1$



Example Problem

Use induction to prove that MergeSort is an $O(n \log n)$ algorithm.

```
Mergesort(array)
    n = size(array)
    if ( n == 1) return array
    array1 = Mergesort(array[1 .. n/2])
    array2 = Mergesort(array[n/2 .. n])
    return Merge(array1, array2)
```

Induction Proof

Example: Prove that $T(n) = 2T(\lfloor n/2 \rfloor) + n$, $T(1) = 1$ is $O(n \log n)$.

We need to prove that $T(n) \leq c\, n \log n$, for all n greater than some value.

1. Base cases: $T(2) = 4 \leq c\,2$ and $T(3) = 5 \leq c\,3 \log_2 3$
   $c \geq 2$ suffices
2. Inductive hypothesis: Assume $T(\lfloor n/2 \rfloor) \leq c\,(\lfloor n/2 \rfloor) \log(\lfloor n/2 \rfloor)$

What is happening to the quantifications?

Induction Step

Given : $T(\lfloor n/2 \rfloor) \leq c\,(\lfloor n/2 \rfloor) \log(\lfloor n/2 \rfloor)$

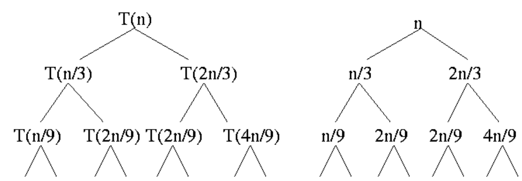
$$
\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\leq 2(\, c(\lfloor n/2 \rfloor) \log(\lfloor n/2 \rfloor)\,) + n \\
&\leq 2(\, c(n/2) \log(n/2)\,) + n \quad (\text{dropping floors makes it bigger!}) \\
&= c\, n \log(n/2) + n \\
&= c\, n\,(\log(n) - \log(2)) + n \\
&= c\, n \log(n) - c\, n + n \quad (\log_2 2 = 1) \\
&= c\, n \log(n) - (c-1)\, n \\
&< c\, n \log(n) \quad (c > 1)
\end{aligned}
$$

Example Problem 2

$T(n) = T(n/3) + T(2n/3) + n$
$T(1) = 1$

Show $T(n)$ is $\Omega(n \log n)$ by appealing to the recursion tree

Recursion Tree



What is the length of the longest path from root to a leaf?

Solution Techniques:
Master Theorem

Master Theorem

- $T(n) = a\,T(n/b) + f(n)$
  - Ignore floors and ceilings for n/b
  - constants $a \geq 1$ and $b > 1$ and $\varepsilon > 0$
  - f(n) an asymptotically positive function
- Case 1:
  - If $f(n) = O(n^{(\log\_b\, a)-\varepsilon})$ for constant $\varepsilon>0$, $T(n) = \Theta(n^{\log\_b\, a})$
- Case 2:
  - If $f(n) = \Theta(n^{\log\_b\, a})$, $T(n) = \Theta(n^{\log\_b\, a} \lg n)$
- Case 3: If
  - $f(n) = \Omega(n^{(\log\_b\, a)+\varepsilon})$ for some constant $\varepsilon > 0$, and
  - $a.f(n/b) \leq c.f(n)$ for some constant $c < 1$ and all sufficiently large n,
  - Then, we have $T(n) = \Theta(f(n))$.
    - Regularity condition: $a.f(n/b) \leq c.f(n)$ for some constant $c < 1$ and all sufficiently large n

Key idea: Compare $n^{\log\_b\, a}$ with f(n)

Examples

- $T(n) = 9\,T(n/3) + n$
  - $a = 9$, $b=3$, $n^{\log\_b\, a} = n^{\log\_3\, 9} = \Theta(n^2)$
  - Thus, we have $f(n) = O(n^{\log\_3\, 9-\varepsilon})$, case 1 applies
  - $T(n) = \Theta(n^2)$
- $T(n) = T(2n/3) + 1$
  - $a = 1$, $b=3/2$, $n^{\log\_b\, 1} = n^0 = 1$
  - Thus, we have $f(n) = \Theta(n^{\log\_b\, a})$; case 2 applies
  - $T(n) = \Theta(n^{\log\_b\, a} \lg n) = \Theta(\lg n)$
- $T(n) = 3T(n/4) + n \log n$
  - $a = 3$, $b=4$, $n^{\log\_4\, 3} = O(n^{0.79})$
  - we have $f(n) = \Omega(n^{\log\_b\, a+\varepsilon})$ for $\varepsilon \cong 0.2$, and
  - $3(n/4) \log (n/4) <= (3/4)\, n \log n$, for $c = 3/4$; case 3 applies
  - $T(n) = \Theta(f(n)) = \Theta(n \log n)$

Examples

- $T(n) = 2T(n/2) + n \log n$
  - $a = 2$, $b=2$, $n^{\log\_b\, a} = n^{\log\_2\, 2} = \Theta(n)$
  - f(n) is asymptotically larger than $(n^{1+\varepsilon})$
  - Thus case 3 applies
  - Wrong! Why?
  - f(n) is not polynomially larger; $f(n)/n = \log n$, which is asymptotically smaller than n^{ε} for any $\varepsilon > 0$
  - Therefore, the regularity condition does not hold.

Solution Techniques:
Characteristics Equations

Characteristic Equation Approach

- $t_n = 3t_{n-1} + 4t_{n-2}$ for $n > 1$
  - $t_0 = 0$, $t_1 = 5$
- Rewrite recurrence
  - $t_n - 3t_{n-1} - 4t_{n-2} = 0$
- Properties
  - Homogeneous: no terms not involving t_n
  - Linear: t_n terms have no squares or worse
  - Constant coefficients: 1, -3, -4

Characteristic Equation

- $t_n - 3t_{n-1} - 4t_{n-2} = 0$
- Rewrite assuming solution of the form $t_n = x^n$
- $x^n - 3x^{n-1} - 4x^{n-2} = 0$
- $x^{n-2}(x^2 - 3x - 4) = 0$
- Find roots of $(x^2 - 3x - 4)$
  - $(x+1)(x-4) \rightarrow$ roots are -1 and 4
- Solution is of form $c_1(-1)^n + c_2 4^n$

Solving for Constants

- $t_n = c_1(-1)^n + c_2 4^n$
- Use base cases to solve for constants
  - $t_0 = 0 = c_1(-1)^0 + c_2 4^0 = c_1 + c_2$
  - $t_1 = 5 = c_1(-1)^1 + c_2 4^1 = -c_1 + 4c_2$
  - $5c_2 = 5 \rightarrow c_2 = 1 \rightarrow c_1 = -1$
- $t_n = (-1)^{n+1} + 4^n$
- Always test solution on small values!

Repeated Roots for Characteristic Equations

- $t_n - 5t_{n-1} + 8t_{n-2} - 4t_{n-3} = 0$
  - boundary conditions: $t_n = n$ for $n = 0, 1, 2$
- $x^3 - 5x^2 + 8x - 4 = 0$
- $(x-1)(x-2)^2 \rightarrow$ roots are 1, 2, 2
- Solution is of form $c_1(1)^n + c_2 2^n + c_3 n 2^n$
  - If root is repeated third time, then $n^2 2^n$ term, and so on

Solving for Constants

- $t_n = c_1(1)^n + c_2 2^n + c_3 n 2^n$
- Use base cases to solve for constants
  - $t_0 = 0 = c_1(1)^0 + c_2 2^0 + c_3 0\ 2^0 = c_1 + c_2$
  - $t_1 = 1 = c_1(1)^1 + c_2 2^1 + c_3 1\ 2^1 = c_1 + 2c_2 + 2c_3$
  - $t_2 = 2 = c_1(1)^2 + c_2 2^2 + c_3 2\ 2^2 = c_1 + 4c_2 + 8c_3$
  - $c_1 = -2$, $c_2 = 2$, $c_3 = -1/2$
- $t_n = 2^{n+1} - n2^{n-1} - 2$
- Test the solution on small values!

Inhomogeneous Equation

- $t_n - 2t_{n-1} = 3^n$
  - base case value for t_0 only
- $(x – 2) (x-3) = 0$
  - (x-2) term comes from homogeneous solution
  - If rhs is of form b^n poly(n) of degree d
    - In this case, b = 3, poly (n) = 1 is of degree 0
  - Plug $(x-b)^{d+1}$ into characteristic equation

Solving for constants

- $(x – 2) (x-3) = 0$
- $t_n = c_1 2^n + c_2 3^n$
- Solve for c_1 and c_2 with only t_0 base case
- This is only 1 equation and 2 unknowns
- Use recurrence to generate extra equations
  - $t_n - 2t_{n-1} = 3^n \rightarrow t_1 = 2t_0 + 3$
- Now we have two equations
  - $t_0 = c_1 2^0 + c_2 3^0 = c_1 + c_2$
  - $t_1 = 2t_0 + 3 = 2\ c_1 + 2\ c_2 + 3 = 2c_1 + 3c_2$ results in $c_2 = 3$
  - $c_1 = t_0 – 3$ and $c_2 = 3$
- $t_n = (t_0-3)2^n + 3^{n+1}$

Changing variable

- $t_n – 3t_{n/2} = n$ if n is a power of 2
  - $t_1 = 1$
- Let $n = 2^i$ and $s_i = t_n$
- $s_i – 3s_{i-1} = 2^i$ for $i >= 1$
  - $s_0 = 1$
- $(x-3)(x-2) = 0$
  - x-3 from characteristic equation
  - x-2 b^npoly(n) rhs

Solving for constants

- $(x-3)(x-2) = 0$
- $s_i = c_1 3^i + c_2 2^i$
- Generating two equations
  - $t_1 = s_0 = 1 = c_1 3^0 + c_2 2^0 = c_1 + c_2$
  - $t_2 = s_1 = 3t_1 + 2 = 5 = c_1 3^1 + c_2 2^1 = 3c_1 + 2c_2$
  - $c_1 = 3, c_2 = -2$
- $s_i = 3^{i+1} – 2^{i+1}$ for $i >= 0$
- $t_n = 3n^{\lg 3} – 2n$ for n a power of 2 >= 1

Example: Towers of Hanoi

- Suppose we have two disks of each size.
- Let n be the number of sizes of disks
  - What is recurrence relation?
  - What is solution?

Example: Merge Sort

- Merge sort breaking array into 3 pieces
  - What is recurrence relation?
  - What is solution?
  - How does this compare to breaking into 2 pieces?