CS 361, Lecture 8

Jared Saia
University of New Mexico

Outline

- Recurrence Relations, Induction, and Substitution Method

Recurrence Relations

- $T(n) = 2 * T(n/2) + n$ is an example of a *recurrence* relation
- A *Recurrence Relation* is any equation for a function T, where T appears on both the left and right sides of the equation.
- We always want to "solve" these recurrence relation by getting an equation for T, where T appears on just the left side of the equation

Recurrence Relations

- Whenever we analyze the run time of a recursive algorithm, we will first get a recurrence relation
- To get the real run time, we need to solve the recurrence relation

Recurrences and Induction

Recurrences and Induction are closely related:

- To *find* some solution to $f(n)$, solve a recurrence
- To *prove* that a solution for $f(n)$ is correct, use induction

For both recurrences and induction, we always solve a big problem by reducing it to smaller problems!

Some Examples

- The next several problems can be attacked by induction/recurrences
- For each problem, we'll need to reduce it to smaller problems
- Question: How can we reduce each problem to a smaller subproblem?

Sum Problem

- $f(n)$ is the sum of the integers $1, \ldots, n$

Tree Problem

- $f(n)$ is the maximum number of leaf nodes in a binary tree of height n

Recall:

- In a binary tree, each node has at most two children
- A *leaf* node is a node with no children
- The height of a tree is the length of the longest path from the root to a leaf node.

Binary Search Problem

- $f(n)$ is the maximum number of queries that need to be made for binary search on a sorted array of size n.

Dominoes Problem

- $f(n)$ is the number of ways to tile a 2 by n rectangle with dominoes (a domino is a 2 by 1 rectangle)

Simpler Subproblems

- Sum Problem: What is the sum of all numbers between 1 and $n-1$ (i.e. $f(n-1)$)?
- Tree Problem: What is the maximum number of leaf nodes in a binary tree of height $n-1$? (i.e. $f(n-1)$)
- Binary Search Problem: What is the maximum number of queries that need to be made for binary search on a sorted array of size $n/2$? (i.e. $f(n/2)$)
- Dominoes problem: What is the number of ways to tile a 2 by $n-1$ rectangle with dominoes? What is the number of ways to tile a 2 by $n-2$ rectangle with dominoes? (i.e. $f(n-1)$, $f(n-2)$)

Recurrences

- Sum Problem: $f(n) = f(n-1) + n$, $f(1) = 1$
- Tree Problem: $f(n) = 2 * f(n-1)$, $f(0) = 1$
- Binary Search Problem: $f(n) = f(n/2) + 1$, $f(1) = 0$
- Dominoes problem: $f(n) = f(n-1) + f(n-2)$, $f(1) = 1$, $f(2) = 1$

Guesses

- Sum Problem: $f(n) = (n+1)n/2$
- Tree Problem: $f(n) = 2^n$
- Binary Search Problem: $f(n) = \log n$
- Dominoes problem: $f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$

Inductive Proofs

"Trying is the first step to failure" - Homer Simpson

- Now that we've made these guesses, we can try using induction to prove they're correct
- (This is the *Substitution Method*)
- We'll give inductive proofs that these guesses are correct for the first three problems

Sum Problem

- Want to show that $f(n) = (n+1)n/2$.
- Prove by induction on n
- Base case: $f(1) = 2 * 1/2 = 1$
- Inductive hypothesis: for all $j < n$, $f(j) = (j+1)j/2$
- Inductive step:

$$
\begin{aligned}
f(n) &= f(n-1) + n && (1)\\
&= n(n-1)/2 + n && (2)\\
&= (n+1)n/2 && (3)
\end{aligned}
$$

Tree Problem

- Want to show that $f(n) = 2^n$.
- Prove by induction on n
- Base case: $f(0) = 2^0 = 1$
- Inductive hypothesis: for all $j < n$, $f(j) = 2^j$
- Inductive step:

$$
\begin{aligned}
f(n) &= 2 * f(n-1) && (4)\\
&= 2 * (2^{n-1}) && (5)\\
&= 2^n && (6)
\end{aligned}
$$

Binary Search Problem

- Want to show that $f(n) = \log n$. (assume n is a power of 2)
- Prove by induction on n
- Base case: $f(1) = \log 1 = 0$
- Inductive hypothesis: for all $j < n$, $f(j) = \log j$
- Inductive step:

$$
\begin{aligned}
f(n) &= f(n/2) + 1 && (7)\\
&= \log n/2 + 1 && (8)\\
&= \log n - \log 2 + 1 && (9)\\
&= \log n && (10)
\end{aligned}
$$

In Class Exercise

Consider the following interview question:

- Out of n coins, one weighs less than the others
- You have a scale
- What is the minimum number of weighs on the scale you can do to find the odd coin?

The Smaller Problem

- Q: How can we reduce the problem of finding the odd coin among n coins to a smaller problem???

Solving the Big Problem

- A: The simpler problem is: "How many weighings does it take to find an odd coin in a set of size $n/3$?"
- Idea:
  - We divide the coins into 3 piles of size $n/3$.
  - We pick two of these piles at random and put them on opposite sides of the scale
  - If one of these two piles weighs less than the other, we know the odd coin is in that pile
  - If both piles weigh the same, we know the odd coin is in the third pile
  - Thus we now know which pile of size $n/3$ contains the odd coin, so we recursively find the odd coin in this pile.

Simplest Case

- If $n = 3$, we can find the odd coin in a single weighing:
  - Choose two coins at random and put each on either side of the scale
  - If both weigh the same, odd coin is the third one. If one coin weighs less, that coin is the odd one

Recurrence

(Assume n is a power of 3)

- Let $f(n)$ be the number of weighings needed to find the odd coin
- Q: What is the recurrence for $f(n)$?
- Note: We first do a single weighing for the two piles of size $n/3$, then the problem reduces to the problem on a pile of coins of size $n/3$

Recurrence

- A: The recurrence is: $f(n) = f(n/3) + 1$, $f(3) = 1$
- "Guess" that the solution to this is $f(n) = \log_3 n$

In Class Exercise

Goal: Prove by induction that the solution to $f(n) = f(n/3) + 1$, $f(3) = 1$ is $f(n) = \log_3 n$

- Q1: What is the base case? Prove that it holds.
- Q2: What is the inductive hypothesis?
- Q3: Prove the inductive step.

Inequalities

- Often easier to prove that a recurrence is no more than some quantity than to prove that it equals something
- Consider: $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$
- "Guess" that $f(n) \leq 2^n$

Inequalities (II)

Goal: Prove by induction that for $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$, $f(n) \leq 2^n$

- Base case: $f(1) = 1 \leq 2^1$, $f(2) = 1 \leq 2^2$
- Inductive hypothesis: for all $j < n$, $f(j) \leq 2^j$
- Inductive step:

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) && (11)\\ &\leq 2^{n-1} + 2^{n-2} && (12)\\ &< 2 * 2^{n-1} && (13)\\ &= 2^n && (14) \end{aligned}$$

Take Away

- Recurrences and Induction are closely related
- Both techniques require that we solve a big problem by using a solution to a smaller problem
- One technique for solving recurrences is to "guess" the solution and then prove this guess is right by induction

Things to think about

- Up to this point, I've been supplying you with good "guesses" for recurrence solutions
- Q: How do we get these guesses?

Good Guesses

Following are some good guesses for solutions to recurrences.
$\log n$
$\sqrt{n}$
n
$n \log n$
n^2
n^3
2^n

Todo

- Read Chapter 4 in book (skip proof of the Masters Theorem)