Solving More Types of Recurrences

Recurrences

We've already seen several run-times expressed as recurrence relations, and have solved for them.

There are different strategies that can be used to solve these.

We will explore determining the asymptotic classes of a variety of recurrences representing run-times.

Generic Example I

We previously saw that if

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

then $T(n) \in O(n \log n)$

What if $T(n)=2T(n/2+7)+n$?

Try using constructive induction to show that there is some constant c such that $T(n) \leq cn \log n$.

Generic Example II

What if $T(n)=2T(n/2)+1$?

Try using constructive induction to show that there is some constant c such that $T(n) \leq cn$.

For posting note: In class our usual constructive induction approach did not work, but it was close. We then tried again using $T(n) \leq cn + a$ where a is another constant (and it can be any real number). This time we were be able to construct constants c and a.

Generic Example III

What if were to use the same technique to try to prove that when $T(n)=2T(n/2)+n$ that there is some constant c such that $T(n)\leq cn$?

For posting note: In class we weren't even close this time. We ended up with a restriction that $n\leq 0$ which is problematic for two reasons; first it eliminates all input size, but second even if it were n being less than some positive value, it would still make n_0 impossible since we need the statement true for all $n>=n_0$.

It's not true. We need a little-ω proof.

Binary Search

We look at the middle element.

- If it is equal to what we want, we can stop.
- If it is greater than our search target, we search the "left" half of the list.
- Otherwise, we search the "right" half of the list.

If we look at the recursive part, we have $T(n) = 2+T(n/2)$. In the worst case, this happens over and over until we hit a stopping point when $n=1$.

Could we use constructive induction to show that $T(n) \in O(n)$? What about $T(n) \in \Omega(n)$?

More Binary Search

With recursion, sketching out a recursion tree representing what occurs can help build intuition and allow us to make a more intelligent guess at the class of run-time we wish to prove.

What is it for a binary search?

Mystery Algorithm I

I have an algorithm in which I ask n questions, am able to eliminate $1/4^{th}$ of the input as candidates, and am able to create three recursive sub-questions of equal size.

$$T(n)=n+3T(n/4)$$

Goal: Make an informed/intelligent guess at its runtime…

Mystery Algorithm II

I have an algorithm in which I am able to create two recursive sub-questions (one using 1/3rd of the data and the other using the remaining 2/3rd) and then by asking *n* questions of the information returned, am able to return the correct answer.

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

Goal: Make an informed/intelligent guess at its runtime…

Mystery Algorithm III

I have an algorithm in which I am able to create two recursive sub-questions (both using 2/3rd of the data so there is overlap) and then by asking n questions of the information returned, am able to return the correct answer.

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

Goal: Make an informed/intelligent guess at its runtime…

Mystery Algorithm IV

I have an algorithm in which I am able to create six recursive sub-questions (using $1/4^{th}$ of the data so there is overlap) and then by asking only 1 more questions of the information returned, am able to return the correct answer.

$$T(n)=6T(n/4)+1$$

Goal: Make an informed/intelligent guess at its runtime…

Are there patterns?

As we look at different recurrences, certain patterns come up where just by looking at the structure of the recursion, we might be able to guess at the run-time from experience.

Is there a way to more precisely describe some of these patterns?

Yes!

It's called…

The Master Theorem

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$ $b > 1$

Case 1: $f(n) \in O(n^{\log_b a - \varepsilon})$ $\varepsilon > 0$

$\Rightarrow T(n) \in \Theta(n^{\log_b a})$

Case 2: $f(n) \in \Theta(n^{\log_b a})$

$\Rightarrow T(n) \in \Theta(n^{\log_b a} \log n)$

Case 3: $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ $\varepsilon > 0$

and $a \cdot f(n/b) \leq c \cdot f(n)$ $c < 1$

$\Rightarrow T(n) \in \Theta(f(n))$

Example

$T(n)=n+3T(n/4)$

$T(n)=aT(n/b)+f(n)$

$a=3$

$b=4$

$f(n)=n$

Example

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

$T(n)=aT(n/b)+f(n)$

$a=2$

$b=3/2$

$f(n)=n$

Example

$T(n)=6T(n/4)+1$

$T(n)=aT(n/b)+f(n)$

$a=6$

$b=4$

$f(n)=1$

Example

Can we apply the Master Theorem to $T(n)=2T(n-5)+n$?

Things to consider…

We are given some recurrence relations that don't fit into the Master Theorem well, such as:

$T(1)=1$

$T(n)=T(n/4)+T(3n/4)+1$

Things we want to observe.

- Structure of the tree
  - Symmetrical?
  - Density?
  - Number of levels? (Best? Worst?)
- Work done
  - At full interior levels?
  - At leaves?
- Asymptotic relationships
  - Big-Omega
  - Big-O

What is the impact of the f(*n*)?

What if we make a change to only the *f(n)* in the previous recurrence:

$T(1)=1$

$T(n)=T(n/4)+T(3n/4)+n$

-vs-

$T(1)=1$

$T(n)=T(n/4)+T(3n/4)+1$