CS146: Data Structures and Algorithms
Lecture 5

SOLVING RECURRENCE RELATIONS

INSTRUCTOR: KATERINA POTIKA
CS SJSU

Recurrence Examples

$$T(n) = \begin{cases} 1 & \text{if } n = 1\,, \\ T(n-1) + 1 & \text{if } n > 1\,. \end{cases}$$

Solution: $T(n) = n$.

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

Solution: $T(n) = n \lg n + n$.

$$T(n) = \begin{cases} 0 & \text{if } n = 2\,, \\ T(\sqrt{n}) + 1 & \text{if } n > 2\,. \end{cases}$$

Solution: $T(n) = \lg \lg n$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1\,, \\ T(n/3) + T(2n/3) + n & \text{if } n > 1\,. \end{cases}$$

Solution: $T(n) = \Theta(n \lg n)$.

Change of variables

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

  Replace m=lgn

- $\mathrm{T}(2^m) = T(2^{m/2}) + 1$

  Rename S(m)=$\mathrm{T}(2^m)$

- S(m)=S(m/2)+1=lgm

- $\mathrm{T(n)=T}(2^m) = S(m) = lgm = lglgn$

Solving Recurrences

1. Substitution method
2. Iteration method
3. Master method

Solving Recurrences

- 1. The substitution method (CLR 4.1)
  - A.k.a. the "making a good guess method"
  - Guess the form of the answer, then use induction to find the constants and show that solution works
  - Examples:
    - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = \Theta(n \lg n)$
    - $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n \rightarrow \Theta(n \lg n)$

Substitution method

- Guess the form of the solution
- Use mathematical induction to find constants and show that your guess was correct

Example: $T(n) = 2T(n/2)+\Theta(n)$ (I)

- Upper bound of $T(n) <= 2T(n/2) + cn$, $c >= 0$
- *Guess:* $T(n) \le dn \lg n$, constant $d >= 0$
- *Substitution:*

$$\begin{aligned}
T(n) &\le 2T(n/2) + cn \\
&= 2\left(d\frac{n}{2}\lg\frac{n}{2}\right) + cn \\
&= dn\lg\frac{n}{2} + cn \\
&= dn\lg n - dn + cn \\
&\le dn\lg n \qquad \text{if } -dn + cn \le 0\,, \\
&\qquad\qquad\qquad\qquad d \ge c
\end{aligned}$$

- Therefore, $T(n) = O(n \lg n)$.

Example: $T(n) = 2T(n/2)+\Theta(n)$ (II)

- Lower bound of $T(n) >= 2T(n/2)+cn$
- *Guess:* $T(n) >= dn \lg n$, constant $d>=0$
- *Substitution:*

$$
\begin{aligned}
T(n) &\geq 2T(n/2) + cn \\
&= 2\left(d\frac{n}{2}\lg\frac{n}{2}\right) + cn \\
&= dn\lg\frac{n}{2} + cn \\
&= dn\lg n - dn + cn \\
&\geq dn\lg n \qquad \text{if } -dn + cn \geq 0, \\
&\qquad\qquad\qquad\qquad\quad d \leq c
\end{aligned}
$$

- Therefore, $T(n) = \Omega(n \lg n)$.

Careful

The false proof for the recurrence

$T(n) = 4T(n/4) + n$, that $T(n) = O(n)$:

Proof: $T(n) \leq 4(c(n/4)) + n \leq cn + n = O(n)$

wrong!

Why?

Because we haven't proven the *exact form* of our inductive hypothesis (which is that $T(n) \leq cn$), this proof is false.

Solving Recurrences

- Another option is what the book calls the "iteration method"
  - Expand the recurrence
  - Work some algebra to express as a summation
  - Evaluate the summation
- We will show several examples

$$s(n)=\begin{cases} 0 & n=0 \\ c+s(n-1) & n>0 \end{cases}$$

- s(n) =

  c + s(n-1)

  c + c + s(n-2)

  2c + s(n-2)

  2c + c + s(n-3)

  3c + s(n-3)

  ...

  kc + s(n-k) = ck + s(n-k)

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have
  - s(n) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have
  - s(n) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn
- So

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

- Thus in general
  - s(n) = cn

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- s(n)

= n + s(n-1)

= n + n-1 + s(n-2)

= n + n-1 + n-2 + s(n-3)

= n + n-1 + n-2 + n-3 + s(n-4)

= …

= n + n-1 + n-2 + n-3 + … + n-(k-1) + s(n-k)

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- s(n)

= n + s(n-1)

= n + n-1 + s(n-2)

= n + n-1 + n-2 + s(n-3)

= n + n-1 + n-2 + n-3 + s(n-4)

= …

= n + n-1 + n-2 + n-3 + … + n-(k-1) + s(n-k)

=

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

$$\sum_{i=1}^{n} i \quad + \quad s(0) \quad = \quad \sum_{i=1}^{n} i \quad + \quad 0 \quad = \quad n\frac{n+1}{2}$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

$$\sum_{i=1}^{n} i \quad + \quad s(0) \quad = \quad \sum_{i=1}^{n} i \quad + \quad 0 \quad = \quad n\frac{n+1}{2}$$

- Thus in general

$$s(n) \quad = \quad n\frac{n+1}{2}$$

- T(n) =
  2T(n/2) + c
  2(2T(n/2/2) + c) + c
  $2^2T(n/2^2) + 2c + c$
  $2^2(2T(n/2^2/2) + c) + 3c$
  $2^3T(n/2^3) + 4c + 3c$
  $2^3T(n/2^3) + 7c$
  $2^3(2T(n/2^3/2) + c) + 7c$
  $2^4T(n/2^4) + 15c$
  ...
  $2^kT(n/2^k) + (2^k - 1)c$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\dfrac{n}{2}\right) + c & n > 1 \end{cases}$$

$$T(n)=\begin{cases} c & n=1 \\ 2T\left(\dfrac{n}{2}\right)+c & n>1 \end{cases}$$

- So far for n > 2k we have
  - $T(n) = 2^k T(n/2^k) + (2^k - 1)c$
- What if k = lg n?
  - $T(n) = 2^{\lg n} T(n/2^{\lg n}) + (2^{\lg n} - 1)c$

    $= n\, T(n/n) + (n - 1)c$

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

    $= nc + (n-1)c = (2n - 1)c$

Solving Recurrences

- The "iteration method"
  - Expand the recurrence
  - Work some algebra to express as a summation
  - Evaluate the summation

The Master Theorem

Given: a *divide and conquer* algorithm

Algorithm divides the problem of size n into a subproblems, each of size n/b

The cost of each phase (i.e., time to divide the problem + combine solved subproblems) be described by the function $f(n)$

Then, the Master Theorem gives us a “cookbook” for the algorithm’s running time:

$$\mathbf{T(n) = aT(n/b) + f(n)}$$

The Master Theorem

if T(n) = aT(n/b) + f(n) then

$$T(n) = \left\{ \begin{array}{ll} \Theta\left(n^{\log_b a}\right) & ,\text{if} \quad f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & ,\text{if} \quad f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & ,\text{if} \quad f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{AND} \\ & \quad af(n/b) < cf(n) \ \text{ for large } n \end{array} \right\} \begin{array}{l} \varepsilon > 0 \\ c < 1 \end{array}$$

Using The Master Method

$T(n) = 9T(n/3) + n$

$a=9$, $b=3$, $f(n) = n$

$n^{\log_b a} = n^{\log_3 9} = n^2$

Since $n = O(n^{\log_3 9 - \varepsilon})$, where $\varepsilon=1$, case 1 applies:

Thus the solution is $T(n) = \Theta(n^2)$

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

$a=1$, $b=3/2$, $f(n)=1$

Case 2 $T(n)=\Theta(\lg n)$

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

A=3, b=4,

Case 3 $T(n)=\Theta(n\lg n)$

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

A=2, b=2, $f(n)=\lg n$

Gap between case 2 and 3

So Master Theorem does not solve every $T(n) = aT(n/b) + f(n)$ ☹

Self test

What is the asymptotic complexity of

T(n) = 7T(n/2)+cn^2

(remember Strassen's Algorithm for Matrix Multiplication)

A. T(n)=Θ($n^{log_2 7}$}

B. T(n)=Θ($n^{log_2 7} * log_2$n)

C. T(n)=Θ(n^2)

D. T(n)=Θ(nlgn)

Self test

What is the asymptotic complexity of

$T(n) = 9T(\frac{n}{3}) + n^2$:

a. $\Theta(n^2 \text{lgn})$

b. $\Theta((n \quad)$

c. $\Theta(n^2)$

d. None of the above

Example: Compare Algorithms

- Algorithm A solves problems by dividing them into 2 subproblems of half the size, recursively solving each subproblem, and then combining the solutions in O($\sqrt{n}$)
- Algorithm B solves problems of size n by recursively solving one subproblem of size n-1 and additional operations take linear time
- Algorithm C solves problems of size n by dividing them into 8 subproblems of size n/4, recursively solving each subproblem, and then combining the solutions in O(n^3) time.
- What are the running times of each of these algorithms (in big-O notation), and which would you choose?