More Recurrences

David Kauchak

cs140
Spring 2023

Recurrence

A function that is defined with respect to itself on smaller inputs

\[ T(n) = 2T(n/2) + n \]
\[ T(n) = 16T(n/4) + n \]
\[ T(n) = 2T(n-1) + n^2 \]

The challenge

Recurrences are often easy to define because they mimic the structure of the program

But... they do not directly express the computational cost, i.e. \( n, n^2, \ldots \)

We want to remove self-recurrence and find a more understandable form for the function
Three approaches

**Substitution method:** when you have a good guess of the solution, prove that it’s correct.

**Recursion-tree method:** If you don’t have a good guess, the recursion tree can help. Then solve with substitution method.

**Master method:** Provides solutions for recurrences of the form:
\[ T(n) = aT(n/b) + f(n) \]

---

**Substitution method**

Guess the form of the solution
Then prove it’s correct by induction

\[ T(n) = T(n/2) + d \]

Halves the input then constant amount of work

**Similar to binary search:**
Guess: \( O(\log_2 n) \)

---

**Guess the solution?**
Recurses into 2 sub-problems that are half the size and performs some operation on all the elements \( O(n \log n) \)

**What if we guess wrong, e.g. \( O(n^2) \)?**

Assume \( T(k) = O(k^2) \) for all \( k < n \)
- again, this implies that \( T(n/2) \leq c(n/2)^2 \)
Show that \( T(n) = O(n^2) \)

---

\[ T(n) = 2T(n/2) + n \]

Guess the solution?
- Recurses into 2 sub-problems that are half the size and performs some operation on all the elements \( O(n \log n) \)

What if we guess wrong, e.g. \( O(n^2) \)?

Assume \( T(k) = O(k^2) \) for all \( k < n \)
- again, this implies that \( T(n/2) \leq c(n/2)^2 \)
Show that \( T(n) = O(n^2) \)

---

\[ T(n) = 2T(n/2) + n \]
\[ \leq 2c(n/2)^2 + n \] from our inductive hypothesis
\[ = 2cn^2/4 + n \]
\[ = 1/2cn^2 + n \]
\[ = cn^2 - (1/2cn^2 - n) \] residual

if
\[-(1/2cn^2 - n) \leq 0 \]
\[-1/2cn^2 + n \leq 0 \]
\[ cn \geq 2 \]

overkill?
What if we guess wrong, e.g. $O(n)$?
Assume $T(k) = O(k)$ for all $k < n$
Again, this implies that $T(n/2) \leq c(n/2)$
Show that $T(n) = O(n)$

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

$$\leq 2cn/2 + n$$

$$= cn + n$$

$$\leq cn$$

\text{factor of } n \text{ so we can just roll it in?}$$

Recurse, recall $T(k) = O(k)$ for all $k < n$
Show that $T(n) = O(n)$

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

$$\leq 2cn/2 + n$$

$$= cn + n$$

$$\leq cn$$

\text{factor of } n \text{ so we can just roll it in?}

Must prove the exact form!

Recursion Tree

Guessing the answer can be difficult

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

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

The recursion tree approach

- Draw out the cost of the tree at each level of recursion
- Sum up the cost of the levels of the tree
- Find the cost of each level with respect to the depth
- Figure out the depth of the tree
- Figure out (or bound) the number of leaves
- Verify your answer using the substitution method
What is the depth of the tree?

At each level, the size of the data is divided by 4

\[
\begin{align*}
\frac{n}{4^d} &= 1 \\
\log \left(\frac{n}{4^d}\right) &= 0 \\
\log n - \log 4^d &= 0 \\
d \log 4 &= \log n \\
d &= \log_4 n
\end{align*}
\]
How many leaves?

How many leaves are there in a complete ternary tree of depth $d$?

$$3^d = 3^{\log_4 n}$$

Total cost

$$T(n) = 3T(n/4) + n^2$$

$$T(1)$$

Total cost

$$T(n) = \frac{16}{13}cn^2 + \Theta(n^{3\log_43})$$

Assignment 1!

$3^{\log_4 n} = 4^{\log_4 n^{3\log_43}}$

$= n^{\log_43}\cdot n^{3\log_43}$

$= n^{3\log_43}$

$T(n) = \frac{16}{13}cn^2 + \Theta(n^{3\log_43})$

$T(n) = O(n^2)$

let $x = 3/16$
Recursion tree

If you went through the exact calculation (like we just did), you can be done!

Often, this isn’t feasible (or desirable)

Instead, use the recursion tree to get a good guess

Verify solution using substitution

\[ T(n) = 3T\left(\frac{n}{4}\right) + n^2 \]

Assume \( T(k) = O(k^2) \) for all \( k < n \)

Show that \( T(n) = O(n^2) \)

Given that \( T(n/4) = O((n/4)^2) \), then

\[ T(n/4) \leq c(n/4)^2 \]

To prove that \( T(n) = O(n^2) \) we need to identify the appropriate constants:

\[ O(g(n)) = \left\{ f(n) : \text{there exists positive constants } c \text{ and } n \text{ such that } \right. \]

\[ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \]

\[ c \text{ such that } T(n) \leq cn^2 \]

\[ T(n) = 3T\left(\frac{n}{4}\right) + n^2 \]

\[ \leq 3c(n/4)^2 + n^2 \]

\[ = cn^2/3 + 16 + n^2 \]

\[ = cn^2 - cn^2 \times \frac{13}{16} + n^2 \] residual

a constant exists if, if \(-cn^2 \times \frac{13}{16} + n^2 \leq 0\)
Master Method

Provides solutions to the recurrences of the form:

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

if \( f(n) = O(n^{\log_a b - \epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(n^{\log_a b}) \)
if \( f(n) = \Theta(n^{\log_a b}) \), then \( T(n) = \Theta(n^{\log_a b} \log n) \)
if \( f(n) = \Omega(n^{\log_a b + \epsilon}) \) for \( \epsilon > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

---

**Case 3?**

- is \( 2^x = O(n^{0 - \epsilon}) \)?
- is \( 2^x = \Theta(n^x) \)?
- is \( 2^x = \Omega(n^{0 + \epsilon}) \)?

---

**T(n) = 16T(n/4) + n**

if \( f(n) = O(n^{\log_a b - \epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(n^{\log_a b}) \)
if \( f(n) = \Theta(n^{\log_a b}) \), then \( T(n) = \Theta(n^{\log_a b} \log n) \)
if \( f(n) = \Omega(n^{\log_a b + \epsilon}) \) for \( \epsilon > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[ a = 16 \]
\[ b = 4 \]
\[ f(n) = n \]

\[ n^{\log_a b} = n \]

is \( n = O(n^{2 - \epsilon}) \)?
is \( n = \Theta(n^2) \)?
is \( n = \Omega(n^{2 + \epsilon}) \)?

---

**T(n) = T(n/2) + 2^n**

if \( f(n) = O(n^{\log_a b - \epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(n^{\log_a b}) \)
if \( f(n) = \Theta(n^{\log_a b}) \), then \( T(n) = \Theta(n^{\log_a b} \log n) \)
if \( f(n) = \Omega(n^{\log_a b + \epsilon}) \) for \( \epsilon > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[ a = 1 \]
\[ b = 2 \]
\[ f(n) = 2^n \]

\[ n^{\log_a b} = n^{\log_2 1} = n^0 \]

is \( 2^x = O(n^{0 - \epsilon}) \)?
is \( 2^x = \Theta(n^x) \)?
is \( 2^x = \Omega(n^{0 + \epsilon}) \)?

---

**T(n) = T(n/2) + 2^n**

if \( f(n) = O(n^{\log_a b - \epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(n^{\log_a b}) \)
if \( f(n) = \Theta(n^{\log_a b}) \), then \( T(n) = \Theta(n^{\log_a b} \log n) \)
if \( f(n) = \Omega(n^{\log_a b + \epsilon}) \) for \( \epsilon > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

is \( 2^x \leq c2^n \) for \( c < 1 \)?

Let \( c = 1/2 \)

\[ 2^{x+1} \leq (1/2)2^n \]
\[ 2^x \leq 2^{x+1} \]
\[ 2^{x+1} \leq 2^x \]

**T(n) = \Theta(2^n)**
\[ T(n) = 2T(n/2) + n \]

if \( f(n) = O(n^{\log_2 2}) \) for \( c > 0 \), then \( T(n) = \Theta(n^{\log_2 2}) \)

if \( f(n) = \Theta(n^{\log_2 2}) \), then \( T(n) = \Theta(n^{\log_2 2} \log n) \)

if \( f(n) = \Omega(n^{\log_2 2}) \) for \( c > 0 \) and \( e(n/b) \leq e(n) \) for \( c < 1 \)

\[ a = 2 \quad b = 2 \quad f(n) = n \]

is \( n = O(n^{\log_2 2}) \)?

is \( n = \Theta(n^{\log_2 2}) \)?

is \( n = \Omega(n^{\log_2 2}) \)?

**Case 2: \( \Theta(n \log n) \)**

\[ T(n) = 16T(n/4) + n! \]

if \( f(n) = O(n^{\log_{16} 4}) \) for \( c > 0 \), then \( T(n) = \Theta(n^{\log_{16} 4}) \)

if \( f(n) = \Theta(n^{\log_{16} 4}) \), then \( T(n) = \Theta(n^{\log_{16} 4} \log n) \)

if \( f(n) = \Omega(n^{\log_{16} 4}) \) for \( c > 0 \) and \( e(n/b) \leq e(n) \) for \( c < 1 \)

\[ a = 16 \quad b = 4 \quad f(n) = n! \]

is \( n! = O(n^{\log_{16} 4}) \)?

is \( n! = \Theta(n^{\log_{16} 4}) \)?

is \( n! = \Omega(n^{\log_{16} 4}) \)?

**Case 3: \( n! \)**

\[ T(n) = \sqrt{2}T(n/2) + \log n \]

if \( f(n) = O(n^{\log_2 \sqrt{2}}) \) for \( c > 0 \), then \( T(n) = \Theta(n^{\log_2 \sqrt{2}}) \)

if \( f(n) = \Theta(n^{\log_2 \sqrt{2}}) \), then \( T(n) = \Theta(n^{\log_2 \sqrt{2}} \log n) \)

if \( f(n) = \Omega(n^{\log_2 \sqrt{2}}) \) for \( c > 0 \) and \( e(n/b) \leq e(n) \) for \( c < 1 \)

\[ a = \sqrt{2} \quad b = 2 \quad f(n) = \log n \]

is \( \log n = O(n^{\log_2 \sqrt{2}}) \)?

is \( \log n = \Theta(n^{\log_2 \sqrt{2}}) \)?

is \( \log n = \Omega(n^{\log_2 \sqrt{2}}) \)?

**Case 1: \( \Theta(\sqrt{n}) \)**
Why does the master method work?

\[ T(n) = aT(n/b) + f(n) \]

\( T(n) = 4T(n/2) + n \)

- If \( f(n) = O(n^k \log^i n) \) for \( i > 0 \), then \( T(n) = \Theta(n^k \log^i n) \)
- If \( f(n) = \Theta(n^k \log^i n) \), then \( T(n) = \Theta(n^k \log^{i+1} n) \)
- If \( f(n) = \Omega(n^k \log^i n) \) for \( k > 0 \) and if \( a \leq b^d \) \( \leq c \) \( f(n) \) for \( c < 1 \), then \( T(n) = \Theta(f(n)) \)

\[ a = 4 \quad n \log_2 a = n \log_2 4 \]
\[ b = 2 \quad n \log_2 b = n^2 \]
\[ f(n) = n \]

- \( n = \Theta(n^2) \)
- \( n = \Theta(n^2) \)
- \( n = \Omega(n^2) \)

What is the depth of the tree?

At each level, the size of the data is divided by \( b \)

\[ \frac{n}{b^d} = 1 \]
\[ \log \left( \frac{n}{b^d} \right) = 0 \]
\[ \log n - \log 4^d = 0 \]
\[ d \log b = \log n \]
\[ d = \log_n n \]

Recurrences

\[ T(n) = 2T(n/3) + d \quad T(n) = 7T(n/7) + n \]

- If \( f(n) = O(n^{\log_b c}) \) for \( c > 1 \), then \( T(n) = \Theta(n^{\log_b c}) \)
- If \( f(n) = \Theta(n^{\log_b c}) \), then \( T(n) = \Theta(n^{\log_b c} \log n) \)
- If \( f(n) = \Omega(n^{\log_b c}) \) for \( c > 1 \) and if \( a \leq b^d \) \( \leq c \) \( f(n) \) for \( c < 1 \), then \( T(n) = \Theta(f(n)) \)

\[ T(n) = T(n-1) + \log n \quad T(n) = 8T(n/2) + n^3 \]
How many leaves?

How many leaves are there in a complete a-ary tree of depth d?

\[ a^d = a^{\log_a n} = n^{\log_a a} \]

Total cost

if \( f(n) = \Theta(n^{c_1}) \) for \( c > 0 \), then \( T(n) = \Theta(n^{c_2}) \)
if \( f(n) = \Theta(n^{c_3}) \), then \( T(n) = \Theta(n^{c_4} \log n) \)
if \( f(n) = \Omega(n^{c_5}) \) for \( c > 0 \) and \( af(n/b) \leq f(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[ T(n) = cf(n) + af(n/b) + a^2 f(n/b^2) + \ldots + a^{k-1} f(n/b^{k-1}) + \Theta(n^{c_6}) \]

Case 2: cost is evenly distributed across tree

As we saw with mergesort, \( \log n \) levels to the tree and at each level \( f(n) \) work

Total cost

if \( f(n) = \Theta(n^{c_7}) \) for \( c > 0 \), then \( T(n) = \Theta(n^{c_8}) \)
if \( f(n) = \Theta(n^{c_9}) \), then \( T(n) = \Theta(n^{c_{10} \log n}) \)
if \( f(n) = \Omega(n^{c_{11}}) \) for \( c > 0 \) and \( af(n/b) \leq f(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[ T(n) = cf(n) + af(n/b) + a^2 f(n/b^2) + \ldots + a^{k-1} f(n/b^{k-1}) + \Theta(n^{c_{12}}) \]

Case 3: cost is dominated by the cost of the root

As we saw with mergesort, \( \log n \) levels to the tree and at each level \( f(n) \) work
Other forms of the master method

\[ T(n) = aT(n/b) + O(n^d) \]

\[
T(n) = \begin{cases} 
O(n^d) & \text{if } d > \log_b a \\
O(n^d \log n) & \text{if } d = \log_b a \\
O(n^{d \log_b b}) & \text{if } d < \log_b a 
\end{cases}
\]