More Recurrences

David Kauchak

Administrative
Assignment 2

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: \( O(\log_2 n) \)

---

Assume \( T(k) = O(\log_2 k) \) for all \( k < n \).

Show that \( T(n) = O(\log_2 n) \).

From our assumption, \( T(n/2) = O(\log_2 n/2) \):

\[ O(g(n)) = \begin{cases} f(n) : & \text{there exists positive constants } c \text{ and } n \text{ such that } \ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{cases} \]

From the definition of big-O: \( T(n/2) \leq c \log_2(n/2) \)

How do we now prove \( T(n) = O(\log n) \)?

---

To prove that \( T(n) = O(\log_2 n) \) identify the appropriate constants:

\[ O(g(n)) = \begin{cases} f(n) : & \text{there exists positive constants } c \text{ and } n \text{ such that } \ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{cases} \]

i.e. some constant \( c' \) such that \( T(n) \leq c' \log_2 n \)

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

\[ \leq c \log_2(n/2) + d \quad \text{from our inductive hypothesis} \]

\[ \leq c \log_2 n - c \log_2 2 + d \]

\[ \leq c \log_2 n + (c + d) \quad \text{residual} \]

Key question: does a constant exist such that:

\[ T(n) \leq c \log_2 n \]
To prove that $T(n) = O(\log_2 n)$, identify the appropriate constants:

Key question: does a constant exist such that:

$T(n) \leq c' \log_2 n$

if $c \geq d$, then, yes!
(if not, just let $c' = d$)

Guess the solution?

At each iteration, does a linear amount of work (i.e. iterate over the data) and reduces the size by one at each step $O(n^2)$

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

- again, this implies that $T(n-1) \leq c(n-1)^2$

Show that $T(n) = O(n^2)$, i.e. $T(n) \leq c'n^2$
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 $T(n) = O(n^2)$

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

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 $T(n) = O(n)$

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

if
$-(1/2cn^2 - n) \leq 0$
$-1/2cn^2 + n \leq 0$

$cn \geq 2$

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

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

$T(n) = 2T(n/2) + n$
$\leq 2cn / 2 + n$
$= cn + n$
$\leq cn$ factor of $n$ so we can just roll it in?

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

Must prove the exact form!
$cn + n \leq cn$ ??
Prove $T(n) = O(n \log_2 n)$
Assume $T(k) = O(k \log_2 k)$ for all $k < n$
→ again, this implies that $T(k) = ck \log_2 k$
Show that $T(n) = O(n \log_2 n)$

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

- $T(n) = 2T(n/2) + n$
- $\leq 2cn / 2 \log(n/2) + n$
- $\leq cn(\log_2 n - \log_2 2) + n$
- $\leq cn \log_2 n + cn + n$
- residual
- $\leq cn \log_2 n$
  if $cn \geq n$, $c > 1$

Recursion Tree

Guessing the answer can be difficult

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

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

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

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

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

$T(n) = 3T(n/4) + n^2$
What is the cost at each level?

\[ T(n) = 3T(n/4) + n^2 \]

Cost

\[ \text{cn}^2 \]

\[ (3/16) \text{cn}^2 \]

Depth

\[ \log_4 n \]

How many leaves?

\[ 3^d = 3^{\log_4 n} \]
Verify solution using substitution

\[ T(n) = 3T(n/4) + n^2 \]

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

\[ O(g(n)) = \begin{cases} f(n) : & \text{there exists positive constants } c \text{ and } n \text{ such that} \\ & 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{cases} \]

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

<table>
<thead>
<tr>
<th>Page</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Total cost</td>
</tr>
<tr>
<td></td>
<td>[ T(n) = cn^2 + \frac{3}{16}cn^2 + \frac{1}{4}cn^2 + \ldots + \frac{1}{16}cn^2 + \Theta(3^{4^k}) ]</td>
</tr>
<tr>
<td></td>
<td>[ = cn^2 \sum_{k=0}^{\log_2 n - 1} \left( \frac{1}{16} \right)^k + \Theta(3^{4^k}) ]</td>
</tr>
<tr>
<td></td>
<td>[ &lt; cn^2 \sum_{k=0}^{\log_2 n - 1} \left( \frac{1}{16} \right)^k + \Theta(3^{4^k}) ]</td>
</tr>
<tr>
<td></td>
<td>[ = \frac{1}{1 - (3/16)}cn^2 + \Theta(3^{4^k}) ]</td>
</tr>
<tr>
<td></td>
<td>[ = \frac{16}{13}cn^2 + \Theta(3^{4^k}) ]</td>
</tr>
<tr>
<td></td>
<td>Let ( x = 3/16 )</td>
</tr>
<tr>
<td>28</td>
<td>Total cost</td>
</tr>
<tr>
<td></td>
<td>[ T(n) = \frac{16}{13}cn^2 + \Theta(3^{4^k}) ]</td>
</tr>
<tr>
<td></td>
<td>Assignment 1!</td>
</tr>
<tr>
<td></td>
<td>[ 3^{2^{2^{2^k}}} = 4^{4^{4^{4^k}}} ]</td>
</tr>
<tr>
<td></td>
<td>[ = 4^{4^{4^{4^{4^k}}} \times 3} ]</td>
</tr>
<tr>
<td></td>
<td>[ = 4^{4^{4^{4^{4^k} \times 3}}} ]</td>
</tr>
<tr>
<td></td>
<td>[ = n^{4^{4^{4^{4^k} \times 3}}} ]</td>
</tr>
<tr>
<td></td>
<td>[ T(n) = \frac{16}{13}cn^2 + \Theta(n^{4^{4^{4^k} \times 3}}) ]</td>
</tr>
<tr>
<td></td>
<td>[ T(n) = O(n^2) ]</td>
</tr>
<tr>
<td>29</td>
<td>Verify solution using substitution</td>
</tr>
<tr>
<td></td>
<td>[ T(n) = 3T(n/4) + n^2 ]</td>
</tr>
</tbody>
</table>
|      | To prove that \( T(n) = O(n^2) \) we need to identify the appropriate constants:
|      | \[ O(g(n)) = \begin{cases} f(n) : & \text{there exists positive constants } c \text{ and } n \text{ such that} \\ & 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{cases} \] |
|      | i.e. some constant \( c \) such that \( T(n) \leq cn^2 \) |
|      | \[ T(n) = 3T(n/4) + n^2 \] |
|      | \[ \leq 3c(n/4)^2 + n^2 \] |
|      | \[ = cn^2 \frac{3}{16} + n^2 \] |
|      | \[ = cn^2 - cn^2 \times \frac{13}{16} + n^2 \] |
|      | residual |
|      | a constant exists if, if \( -cn^2 \times \frac{13}{16} + n^2 \leq 0 \) |
| 30   | Total cost |
|      | \[ T(n) = 3T(n/4) + n^2 \] |
|      | To prove that \( T(n) = O(n^2) \) we need to identify the appropriate constants:
|      | \[ O(g(n)) = \begin{cases} f(n) : & \text{there exists positive constants } c \text{ and } n \text{ such that} \\ & 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \end{cases} \] |
|      | i.e. some constant \( c \) such that \( T(n) \leq cn^2 \) |
|      | \[ T(n) = 3T(n/4) + n^2 \] |
|      | \[ \leq 3c(n/4)^2 + n^2 \] |
|      | \[ = cn^2 \frac{3}{16} + n^2 \] |
|      | \[ = cn^2 - cn^2 \times \frac{13}{16} + n^2 \] |
|      | residual |
|      | a constant exists if, if \( -cn^2 \times \frac{13}{16} + n^2 \leq 0 \) |
The appropriate constants:

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

$O(g(n)) = \begin{cases} f(n): & \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \\ \end{cases}$

i.e. some constant $c$ such that $T(n) \leq cn^2$

$-c n^2 + n^2 \leq 0$

$-\frac{13}{16} n^2 + n^2 \leq 0$

$c n^2 - \frac{13}{16} n^2 \geq n^2$

$c \geq \frac{16}{13}$

---

**Master Method**

Provides solutions to the recurrences of the form:

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

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

---

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

**Case 1:** $\Omega(n^2)$

- $a = 16$
- $b = 4$
- $f(n) = n$

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

---

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

**Case 1:** $\Theta(n^2)$

- $a = 16$
- $b = 4$
- $f(n) = n$

- $n^{log_b a} = n^{log_4 16} = n^4$

---

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

**Case 3?**

- $a = 1$
- $b = 2$
- $f(n) = 2^n$

- is $2^n = O(n^{0-\varepsilon})$?
- is $2^n = \Theta(n^{0})$?
- is $2^n = \Omega(n^{0+\varepsilon})$?
\[ T(n) = T(n/2) + 2^n \]

- If \( f(n) = O(2^n) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(2^n) \)
- If \( f(n) = \Theta(2^n) \), then \( T(n) = \Theta(2^n \log n) \)
- If \( f(n) = \Omega(2^n) \) for \( \epsilon > 0 \) and \( g(n/b) \leq cf(n) \) for \( c < 1 \) then \( T(n) = \Theta(f(n)) \)

is \( 2^{n/2} \leq c2^n \) for \( c < 1 \)?

Let \( c = \frac{1}{2} \)

\[
\begin{align*}
2^{n/2} &\leq (1/2)2^n \\
2^{n/2} &\leq 2^{1/2}2^n \\
2^{n/2} &\leq 2^{1/4}2^n \\
\end{align*}
\]

\[ T(n) = \Theta(2^n) \]

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

- If \( f(n) = O(2^n) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(2^n) \)
- If \( f(n) = \Theta(2^n) \), then \( T(n) = \Theta(2^n \log n) \)
- If \( f(n) = \Omega(2^n) \) for \( \epsilon > 0 \) and \( g(n/b) \leq cf(n) \) for \( c < 1 \) then \( T(n) = \Theta(f(n)) \)

\[
\begin{align*}
a &= 16 \\
b &= 4 \\
f(n) &= n! \\
n_{\log, a} &= n_{\log, 16} \\
&= n^{2} \\
\end{align*}
\]

**Case 3?**

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

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

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

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

- If \( f(n) = O(2^n) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(2^n) \)
- If \( f(n) = \Theta(2^n) \), then \( T(n) = \Theta(2^n \log n) \)
- If \( f(n) = \Omega(2^n) \) for \( \epsilon > 0 \) and \( g(n/b) \leq cf(n) \) for \( c < 1 \) then \( T(n) = \Theta(f(n)) \)

\[
\begin{align*}
a &= 2 \\
b &= 2 \\
f(n) &= n \\
n_{\log, a} &= n^{\log, 2} \\
&= n \\
\end{align*}
\]

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

is \( n = \Theta(n) \)?

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

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

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

- If \( f(n) = O(2^n) \) for \( \epsilon > 0 \), then \( T(n) = \Theta(2^n) \)
- If \( f(n) = \Theta(2^n) \), then \( T(n) = \Theta(2^n \log n) \)
- If \( f(n) = \Omega(2^n) \) for \( \epsilon > 0 \) and \( g(n/b) \leq cf(n) \) for \( c < 1 \) then \( T(n) = \Theta(f(n)) \)

is \( 16n/4)! \leq cn! \) for \( c < 1 \)?

Let \( c = 1/2 \)

\[
\begin{align*}
cn! &= 1/2n! \\
&> (n/2)! \\
\end{align*}
\]

therefore,

\[
16(n/4)! \leq (n/2)! < 1/2n! \\
T(n) = \Theta(n!) \\
\]
\[ T(n) = \sqrt{2T(n/2)} + \log n \]

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

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

if \( f(n) = \Omega(n^{\log_2 a}) \) for \( c > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[
\begin{align*}
a &= \sqrt{2} \\
b &= 2 \\
f(n) &= \log n
\end{align*}
\]

is \( \log n = O(n^{1/2 - \epsilon}) \)?

is \( \log n = \Theta(n^{1/2}) \)?

is \( \log n = \Omega(n^{1/2 + \epsilon}) \)?

**Case 1:** \( \Theta(\sqrt{n}) \)

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

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

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

if \( f(n) = \Omega(n^{\log_2 a}) \) for \( c > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[
\begin{align*}
a &= 4 \\
b &= 2 \\
f(n) &= n
\end{align*}
\]

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

is \( n = \Theta(n^2) \)?

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

**Case 1:** \( \Theta(n^2) \)

**Recurrences**

\[
\begin{align*}
T(n) &= 2T(n/3) + d \\
T(n) &= 7T(n/7) + n
\end{align*}
\]

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

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

if \( f(n) = \Omega(n^{\log_2 a}) \) for \( c > 0 \) and \( af(n/b) \leq cf(n) \) for \( c < 1 \)
then \( T(n) = \Theta(f(n)) \)

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

**Why does the master method work?**

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

\[
\begin{align*}
a &= \log_2 a \\
b &= \log_b n \\
f(n) &= \begin{cases} f(n/b) & \text{if } n \geq b^\alpha f(n/b^\beta) & \text{if } n \geq b^\alpha f(n/b^\gamma) & \text{if } n \geq b^\alpha \end{cases}
\end{align*}
\]
What is the depth of the tree?

At each level, the size of the data is divided by $b$

$$\frac{n}{b} = 1$$
$$\log \left( \frac{n}{b} \right) = 0$$
$$\log n - \log b^d = 0$$
$$d \log b = \log n$$
$$d = \log_b n$$

How many leaves?

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

$$a^d = a^{\log_b n}$$
$$= n^{\log_a a}$$

Total cost

$$T(n) = cf(n) + a'f(n/b) + a^2f(n/b^2) + ... + a^{d-1}f(n/b^{d-1}) + \Theta(a^{d-1})$$

Case 1: cost is dominated by the cost of the leaves

$$= \sum_{k=0}^{d-1} a^k f(n/b^k) < \Theta(a^{d-1})$$

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 \( f(n) = \Theta(n^{\log a}) \) \( \text{for } c > 0 \), then \( T(n) = \Theta(n^{\log a}) \)
if \( f(n) = \Theta(n^{\log a}) \), then \( T(n) = \Theta(n^{\log a} \log n) \)
if \( f(n) = \Theta(n^{\log a}) \) \( \text{for } c > 0 \) and \( af(n/b) \leq cf(n) \) \( \text{for } c < 1 \)
then \( T(n) = \Theta(f(n)) \)

\[
T(n) = cf(n) + af(n/b) + a^2 f(n/b^2) + \cdots + a^{d-1} f(n/b^{d-1}) + \Theta(n^{\log a})
\]

\[
= \sum_{i=0}^{d-1} a^i f(n/b^i) + \Theta(n^{\log a})
\]

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

Other forms of the master method

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

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