More Recurrences

David Kauchak
.cs140
Spring 2024

Administrative

Group sessions
Assignment 1

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:

\[ T(n) = aT\left(\frac{n}{b}\right) + f(n) \]

Recursion Tree

Guessing the answer can be difficult

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

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

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

```
T(n) = 3T(n/4) + n²
cost

T(n) = 3T(n/4) + n²
cost
```

```
T(n) = 3T(n/4) + n²
cost

T(n) = 3T(n/4) + n²
cost
```
What is the cost at each level?

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

\[ \text{cost} \]

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

\[ \frac{n}{4} = 1 \]

\[ \log(\frac{n}{4}) = 0 \]

\[ \log n - \log 4 = 0 \]

\[ d \log 3 = \log n \]

\[ d = \log_3 n \]

What is the depth of the tree?

How many leaves are there?

How many leaves are there in a complete ternary tree of depth \( d \)?

\[ 3^d = 3^{\log_3 n} \]
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

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

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

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

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

Provides solutions to the recurrences of the form:

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

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

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

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

\[ O(g(n)) = \Theta(f(n)) \text{ if } \exists c > 0 \text{ such that } T(n) \leq cn^2 \]

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

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

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

\[ = cn^2 - cn^{13/16} + n^2 \]

residual

a constant exists if \(-cn^{13/16} + n^2 \leq 0\)

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

if \( f(n) = O(n^{1/4}) \) then \( T(n) = O(n^{1/4} \log n) \)

\[ t(n) = 4n + n \]

\[ a = 16 \quad n^{log_b a} = n^{\log_4 16} \]

\[ b = 4 \quad n^t = n^1 \]

\[ f(n) = n \]

\[ \text{Case 1: } \Theta(n^2) \]
Let $c = \frac{1}{2}$.

Case 3?

Case 2: $\Theta(n) = \Theta(\log n)$

Case 3?

$T(n) = \Theta(2^n)$
\( T(n) = 16T(n/4) + n! \)

- if \( f(n) = O(n^{-\epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = O(n^{\Theta(\log n)}) \)
- if \( f(n) = O(n^{\epsilon}) \), then \( T(n) = O(n^{\Theta(\log n)}) \)
- if \( f(n) = O(n^{\Theta(\log n)}) \) for \( \epsilon > 0 \) and \( n/4 \leq n/2 \)

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

Let \( c_1 = 1/2 \)
\[ c_1 n! = 1/2 n! > (n/2)! \]

therefore,
\[ 16(n/4)! \leq (n/2)! < 1/2 n! \]

\[ T(n) = \Theta(n!) \]

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

- if \( f(n) = O(n^{-\epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = O(n^{\Theta(\log n)}) \)
- if \( f(n) = O(n^{\epsilon}) \), then \( T(n) = O(n^{\Theta(\log n)}) \)
- if \( f(n) = O(n^{\Theta(\log n)}) \) for \( \epsilon > 0 \) and \( n/4 \leq n/2 \)

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

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

Case 1: \( \Theta(n^{1/2}) \)

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

- if \( f(n) = O(n^{-\epsilon}) \) for \( \epsilon > 0 \), then \( T(n) = O(n^{\Theta(\log n)}) \)
- if \( f(n) = O(n^{\epsilon}) \), then \( T(n) = O(n^{\Theta(\log n)}) \)
- if \( f(n) = O(n^{\Theta(\log n)}) \) for \( \epsilon > 0 \) and \( n/4 \leq n/2 \)

is \( n = O(n^{\Theta(\log n)}) \)?

Case 1: \( \Theta(n^{\Theta(\log n)}) \)

Recurrences

\( T(n) = 2T(n/3) + d \)

\( T(n) = 7T(n/7) + n \)

\( T(n) = T(n-1) + \log n \)

\( T(n) = 8T(n/2) + n^{1/2} \)
Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?

Why does the master method work?
Total cost

\[ T(n) = \sum_{i=1}^{\log_2 n} (c_i f(n) + O(n^{a_i})) \]

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.

Other forms of the master method

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

- \( O(n^d) \) if \( d > \log_b a \)
- \( O(n^d \log a) \) if \( d = \log_b a \)
- \( O(n^{d+\epsilon}) \) if \( d < \log_b a \)

Changing variables

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

Guesses?

We can do a variable change: let \( m = \log_2 n \) (or \( n = 2^m \))

\[ T(2^m) = 2T(2^{m/2}) + m \]

Now, let \( S(m) = T(2^m) \)

\[ S(m) = 2S(m/2) + m \]
**Changing variables**

\[ S(m) = 2S(m/2) + m \]

Guess? \[ S(m) = O(m \log m) \]

\[ T(n) = T(2^m) = S(m) = O(m \log m) \]

substituting \( m = \log n \)

\[ T(n) = O(\log n \log \log n) \]