Advanced Data Structure and Algorithm

Recurrence Relations and how to solve them!

Part-2

Understanding the Master Theorem

- Let $a \geq 1, b > 1,$ and d be constants.
- Suppose $T(n) = a \cdot T\left(\frac{n}{b}\right) + O\left(n^d\right)$. Then

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

- What do these three cases mean?

The eternal struggle

Branching causes the number of problems to explode!
The most work is at the bottom of the tree!

The problems lower in the tree are smaller!
The most work is at the top of the tree!

Consider our three recursive cases

1. $T(n) = T\left(\frac{n}{2}\right) + n$
2. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$
3. $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n$

First example: tall and skinny tree

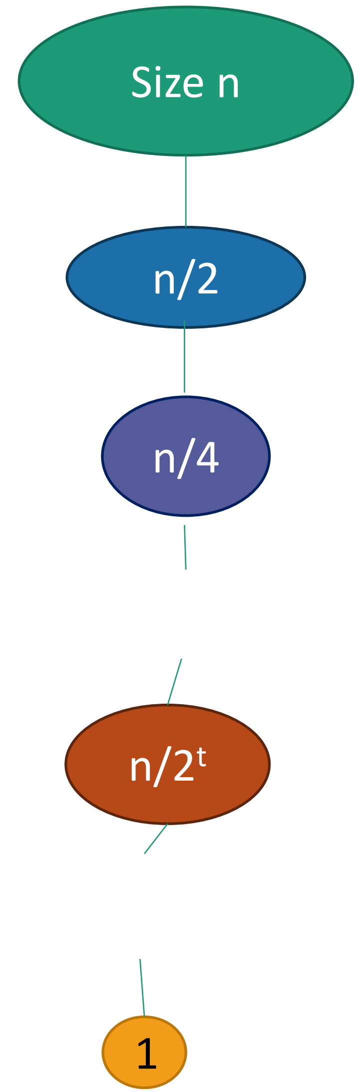
1. $T(n) = T\left(\frac{n}{2}\right) + n, \qquad \left(a < b^d\right)$

- The amount of work done at the top (the biggest problem) swamps the amount of work done anywhere else.

- T(n) = O(work at top) = O(n)

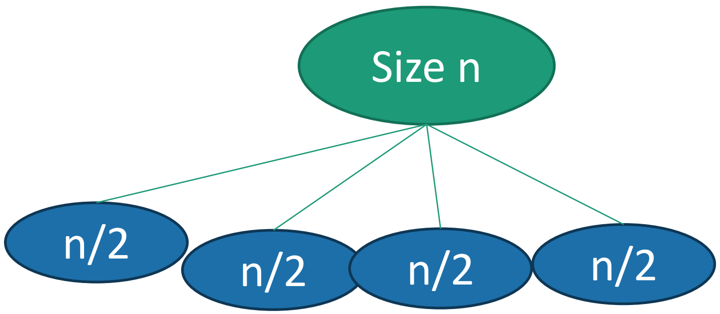
WINNER

Most work at the top of the tree!



Third example: bushy tree

3. $T(n) = 4 \cdot T\left(\frac{n}{2}\right) + n, \qquad \left(a > b^d\right)$

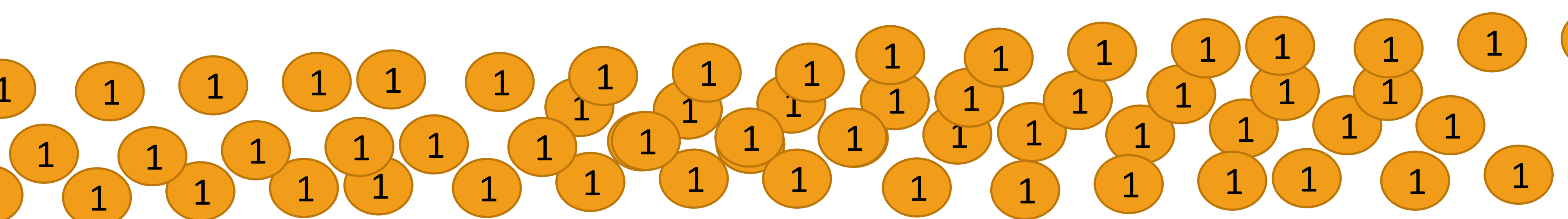


WINNER

Most work at the bottom of the tree!

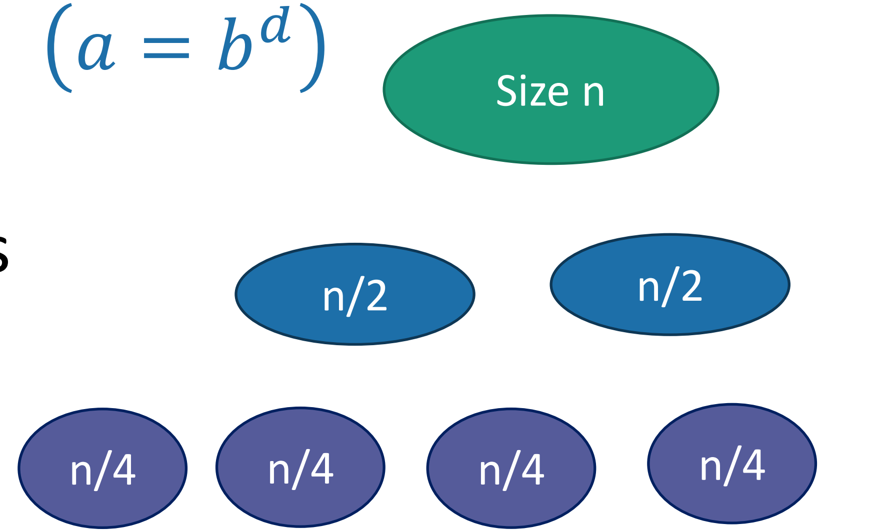
- There are a HUGE number of leaves, and the total work is dominated by the time to do work at these leaves.

- $T(n) = O(\text{ work at bottom }) = O(\ 4^{\text{depth of tree}}\ ) = O(n^2)$



Second example: just right

2. $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n, \qquad \left(a = b^d\right)$



- The branching **just** balances out the amount of work.
- The same amount of work is done at every level.
- T(n) = (number of levels) * (work per level)
- = log(n) * O(n) = O(nlog(n))

What have we learned?

- The “Master Method” makes our lives easier.
- But it’s basically just codifying a calculation we could do from scratch if we wanted to.

The Substitution Method

- Another way to solve recurrence relations.
- More general than the master method.

- Step 1: Generate a guess at the correct answer.
- Step 2: Try to prove that your guess is correct.
- (Step 3: Profit.)

The Substitution Method

first example

- Let's return to:

  $$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$, with $T(0) = 0, T(1) = 1$.
- The Master Method says $T(n) = O(n\log(n))$.
- We will prove this via the Substitution Method.

$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$, with $T(1) = 1$.

Step 1: Guess the answer

- $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$
- $T(n) = 2 \cdot \left(2 \cdot T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$
- $T(n) = 4 \cdot T\left(\frac{n}{4}\right) + 2n$
- $T(n) = 4 \cdot \left(2 \cdot T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$
- $T(n) = 8 \cdot T\left(\frac{n}{8}\right) + 3n$
- ...

Expand $T\left(\frac{n}{2}\right)$

Simplify

Expand $T\left(\frac{n}{4}\right)$

Simplify

Guessing the pattern: $T(n) = 2^t \cdot T\left(\frac{n}{2^t}\right) + t \cdot n$

Plug in $t = \log(n)$, and get

$$T(n) = n \cdot T(1) + \log(n) \cdot n = n(\log(n) + 1)$$

$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$, with $T(1) = 1$.

Step 2: Prove the guess is correct.

- Inductive Hyp. (n): $T(j) = j(\log(j) + 1)$ for all $1 \leq j \leq n$.
- Base Case (n=1): $T(1) = 1 = 1 \cdot (\log(1) + 1)$
- Inductive Step:
  - Assume Inductive Hyp. for n=k-1:
    - Suppose that $T(j) = j(\log(j) + 1)$ for all $1 \leq j \leq k - 1$.
  - $T(k) = 2 \cdot T\left(\frac{k}{2}\right) + k$ by definition
  - $T(k) = 2 \cdot \left(\frac{k}{2}\left(\log\left(\frac{k}{2}\right) + 1\right)\right) + k$ by induction.
  - $T(k) = k(\log(k) + 1)$ by simplifying.
  - So Inductive Hyp. holds for n=k.
- Conclusion: For all $n \geq 1, T(n) = n(\log(n) + 1)$

We just replaced the "n" in the statement of the inductive hypothesis with an "k-1" to get the I.H. for k-1.

Step 3: Profit

- Pretend like you never did Step 1, and just write down:

- *Theorem:* $T(n) = O(n \log(n))$
- *Proof:* [Whatever you wrote in Step 2]

What have we learned?

- The substitution method is a different way of solving recurrence relations.

- Step 1: Guess the answer.
- Step 2: Prove your guess is correct.
- Step 3: Profit.

Another example

- $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 32 \cdot n$
- $T(2) = 2$

- Step 1: Guess: $O(n \log(n))$ (divine inspiration).
- But I don't have such a precise guess about the form for the $O(n \log(n))$ …
  - That is, what's the leading constant?
- Can I still do Step 2?

Step 2: Prove it, working backwards to figure out the constant

- Guess: $T(n) \leq C \cdot n \log(n)$ for some constant C TBD.
- Inductive Hypothesis: $T(j) \leq C \cdot j \log(j)$ for $2 \leq j \leq n$
- Base case: $T(2) = 2 \leq C \cdot 2 \log(2)$ as long as $C \geq 1$
- Inductive Step:

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 32 \cdot n$$
$$T(2) = 2$$

Inductive step

- Assume that the inductive hypothesis holds for n=k-1.
- $T(k) = 2T\left(\frac{k}{2}\right) + 32k$
- $\qquad \leq 2C\frac{k}{2}\log\left(\frac{k}{2}\right) + 32k$
- $\qquad = k(C \cdot \log(k) + 32 \; - C)$
- $\qquad \leq k(C \cdot \log(k))$ as long as $C \geq 32$.
- Then the inductive hypothesis holds for n=k.

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 32 \cdot n$$
$$T(2) = 2$$

Step 2: Prove it, working backwards to figure out the constant

- Guess: $T(n) \leq C \cdot n \log(n)$ for some constant C TBD.
- Inductive Hypothesis: $T(j) \leq C \cdot j \log(j)$ for $2 \leq j \leq n$
- Base case: $T(2) = 2 \leq C \cdot 2 \log(2)$ as long as $C \geq 1$
- Inductive step: Works as long as $C \geq 32$
  - So choose $C = 32$.
- Conclusion: $T(n) \leq 32 \cdot n \log(n)$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 32 \cdot n$$
$$T(2) = 2$$

Step 3: Profit.

- *Theorem*: $T(n) = O(n\log(n))$
- *Proof*:
  - Inductive Hypothesis: $T(j) \leq 32 \cdot j\log(j)$ for $2 \leq j \leq n$
  - Base case: $T(2) = 2 \leq 32 \cdot 2\log(2)$ is true.
  - Inductive step:
    - Assume Inductive Hyp. for n=k-1.
    - $T(k) = 2T\left(\frac{k}{2}\right) + 32k$ By the def. of T(k)
    - $\leq 2 \cdot 32 \cdot \frac{k}{2}\log\left(\frac{k}{2}\right) + 32k$ By induction
    - $= k(32 \cdot \log(k) + 32 - 32)$
    - $= 32 \cdot k\log(k)$
    - This establishes inductive hyp. for n=k.
  - Conclusion: $T(n) \leq 32 \cdot n\log(n)$ for all $n \geq 2$.

Why two methods?

- Sometimes the Substitution Method works where the Master Method does not.

A fun recurrence relation

- $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + n$ for $n > 10$.
- Base case: $T(n) = 1 \text{ when } 1 \leq n \leq 10$

The Substitution Method

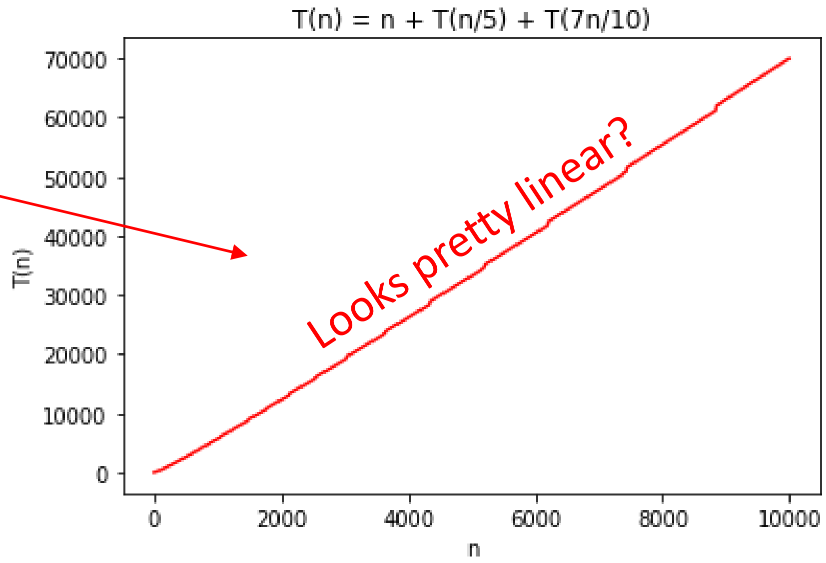
- Step 1: Guess what the answer is.
- Step 2: Prove by induction that your guess is correct.
- Step 3: Profit.

Step 1: guess the answer

$$T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + n \text{ for } n > 10.$$

Base case: $T(n) = 1$ when $1 \le n \le 10$

- Trying to work backwards gets gross fast…
- We can also just try it out.
- Let's guess O(n) and try to prove it.



Step 2: prove our guess is right

$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + n$ for $n > 10$.

Base case: $T(n) = 1$ when $1 \leq n \leq 10$

C is some constant we'll have to fill in later!

- Inductive Hypothesis: $T(j) \leq \boldsymbol{C}j$ for all $1 \leq j \leq n$.
- Base case: $1 = T(j) \leq \boldsymbol{C}j$ for all 1 ≤ j ≤ 10
- Inductive step:
  - Assume that the IH holds for n=k-1.
  - $T(k) \leq k + T\left(\frac{k}{5}\right) + T\left(\frac{7k}{10}\right)$

$$\leq k + \boldsymbol{C} \cdot \left(\frac{k}{5}\right) + \boldsymbol{C} \cdot \left(\frac{7k}{10}\right)$$

$$= k + \frac{\boldsymbol{C}}{5}k + \frac{7\boldsymbol{C}}{10}k$$

$$\leq \boldsymbol{C}k \text{ ??}$$

  - (aka, want to show that IH holds for k=n).

Whatever we choose C to be, it should have C≥1

Let's solve for C and make this true!
C = 10 works.

- Conclusion:
  - There is some $\boldsymbol{C}$ so that for all $n \geq 1$, $T(n) \leq \boldsymbol{C}n$
  - Aka, T(n) = O(n). (Technically we also need $0 \leq T(n)$ here…)

$$T(n) \leq n + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) \text{ for } n > 10.$$

Base case: $T(n) = 1$ when $1 \leq n \leq 10$

(Assume that $T(n) \geq 0$ for all n. Then,)

Step 3: Profit

(Aka, pretend we knew this all along).

Theorem: $T(n) = O(n)$

Proof:

- Inductive Hypothesis: $T(j) \leq \mathbf{10}j$ for all $1 \leq j \leq n$.
- Base case: $1 = T(j) \leq \mathbf{10}j$ for all $1 \leq j \leq 10$
- Inductive step:
  - Assume the IH holds for n=k-1.
  - $T(k) \leq k + T\left(\frac{k}{5}\right) + T\left(\frac{7k}{10}\right)$
    $\leq k + \mathbf{10} \cdot \left(\frac{k}{5}\right) + \mathbf{10} \cdot \left(\frac{7k}{10}\right)$
    $= k + 2k + 7k = \mathbf{10}k$
  - Thus IH holds for n=k.
- Conclusion:
  - For all $n \geq 1, T(n) \leq \mathbf{10}n$
  - (Also $0 \leq T(n)$ for all $n \geq 1$ since we assumed so.)
  - Aka, T(n) = O(n), using the definition with $n_0 = 1, c = 10$.

What have we learned?

- The substitution method can work when the master theorem doesn't.
  - For example with different-sized sub-problems.

- Step 1: generate a guess
  - Guess the rough estimate using back tracking.
- Step 2: try to prove that your guess is correct
  - You may have to leave some constants unspecified till the end – then see what they need to be for the proof to work!!
- Step 3: profit
  - Pretend you didn't do Steps 1 and 2 and write down a nice proof.

Acknowledgement

- Stanford University