

Sample Algorithm Analysis

(based on MIT Open Course)

May 23, 2014

1 Analysis of Recurrence

1.1 Master Method

Proof is done by recursive tree. Although I did not prove for each cases in Figure 1.

1.1.1 Example

Time analysis for Merge Sort.

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

It is Case 2 where $O(n) = \theta(n^{\log_2 2}) \rightarrow T(n) = \theta(n \log n)$

1.1.2 Example

Time analysis for

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

This is Case 1 where $f(n) < \theta(n^{\log_2 4}) \rightarrow T(n) = O(n^2)$

1.1.3 Example

Time analysis for

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

This is Case 3 where $f(n) > \theta(n^{\log_2 4}) \rightarrow T(n) = O(n^3)$

1.2 Recursion Tree

1.2.1 Example

Time analysis for

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + n^2$$

The analysis is shown in Figure 2.

1.3 Substitution Method

Here we use induction to solve recurrences.

1. Guess the answer form. Do not need to guess the constants.
2. Verify by induction.
3. You will get the constants for free!

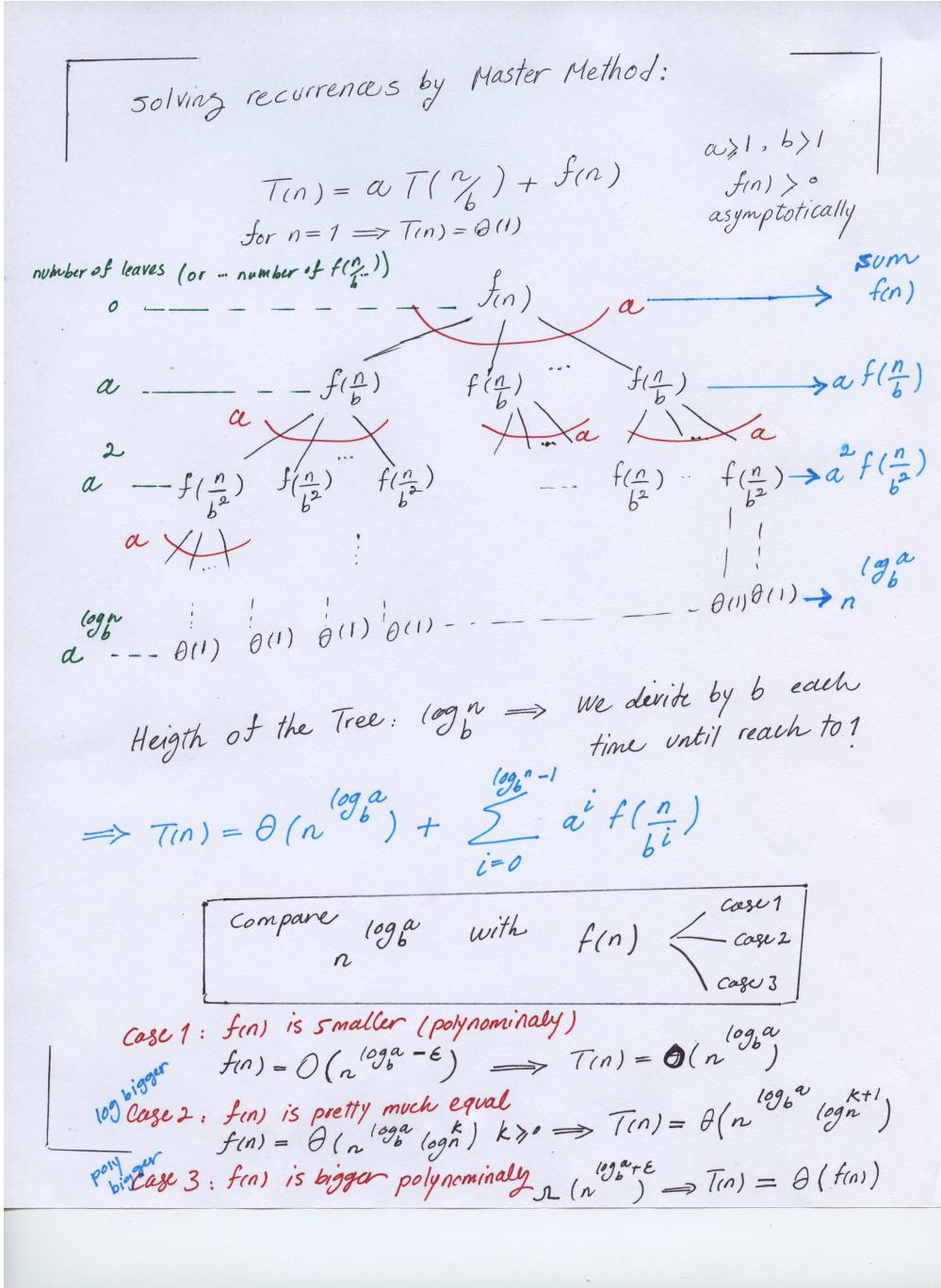


Figure 1: Sketch of recursion tree for Master method proof.

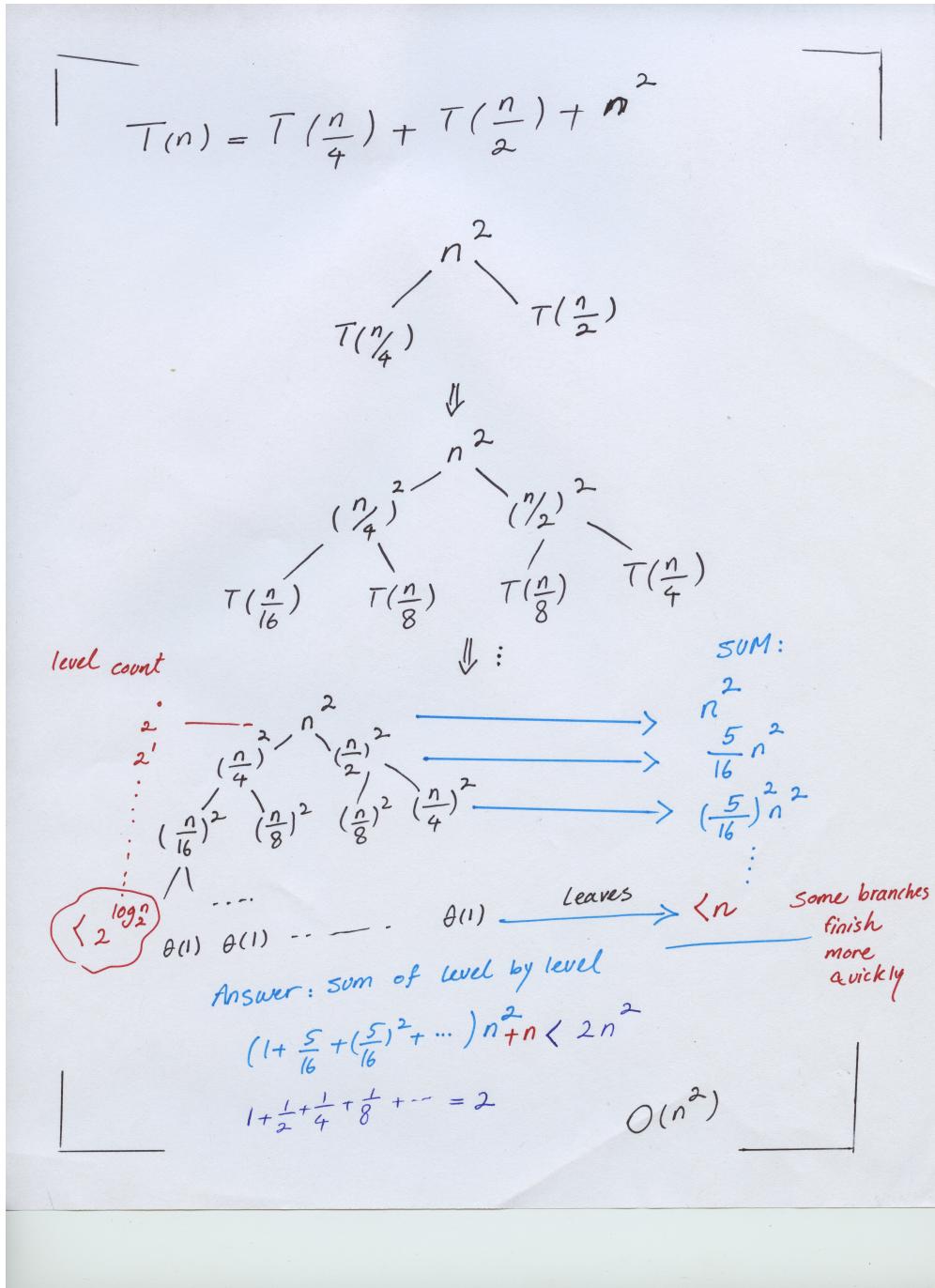


Figure 2: Recursion tree.

1.3.1 Example

Time analysis for

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

1. Lets see if it is $O(n^3)$
2. Base case: $T(1) = \theta(1) \leq c$ depending on $\theta(1)$: c should be sufficiently greater than $\theta(1)$

If $\forall k < n$:

$$T(k) = 4T\left(\frac{k}{2}\right) + k \leq ck^3,$$

We want to prove that for n:

$$T(n) = 4T\left(\frac{n}{2}\right) + n \leq cn^3$$

Proof:

$$\begin{aligned} \frac{n}{2} < n \rightarrow T\left(\frac{n}{2}\right) &\leq c\frac{n^3}{8} \\ \rightarrow 4T\left(\frac{n}{2}\right) + n &\leq c\frac{n^3}{2} + n \leq cn^3 - (c\frac{n^3}{2} - n) \\ \forall c \geq 2 : (c\frac{n^3}{2} - n) &> 0 \rightarrow \\ \forall c \geq 2 : 4T\left(\frac{n}{2}\right) + n &\leq cn^3 \end{aligned}$$

□

3. c should be sufficiently greater than $\theta(1)$ and also $c \geq 2$

But this is one upper bound. Now we want to prove it for another less upper bound:

1. Lets see if it is $O(n^2)$
2. Base case: $T(1) = \theta(1) \leq c_1 - c_2$ depending on $\theta(1)$: c_1 should be sufficiently greater than c_2

If $\forall k < n$:

$$T(k) = 4T\left(\frac{k}{2}\right) + k \leq c_1k^2 - c_2k$$

We want to prove that for n:

$$T(n) = 4T\left(\frac{n}{2}\right) + n \leq c_1n^2 - c_2n$$

Proof:

$$\begin{aligned} \frac{n}{2} < n \rightarrow T\left(\frac{n}{2}\right) &\leq c_1\frac{n^2}{4} - c_2\frac{n}{2} \\ \rightarrow 4T\left(\frac{n}{2}\right) + n &\leq c_1n^2 - 2c_2n + n \leq c_1n^2 - c_2n - ((c_2 - 1)n) \\ \forall c_2 \geq 1/2 : (c_2 - 1)n &> 0 \rightarrow \\ \forall c_2 \geq 1/2 : 4T\left(\frac{n}{2}\right) + n &\leq c_1n^2 - c_2n \end{aligned}$$

□

3. Depending on $\theta(1)$: c_1 should be sufficiently greater than c_2 and also $c_2 \geq 1/2$