Sample Algorithm Analysis

(based on MIT Open Course)

May 24, 2014

1 Analysis of Recurrence

1.1 Master Method

Proof is done by recursive tree. Master method sketch proof is in Figure 1.

1.1.1 Example

Time analysis for Merge Sort.

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

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

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

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

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

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

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

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!

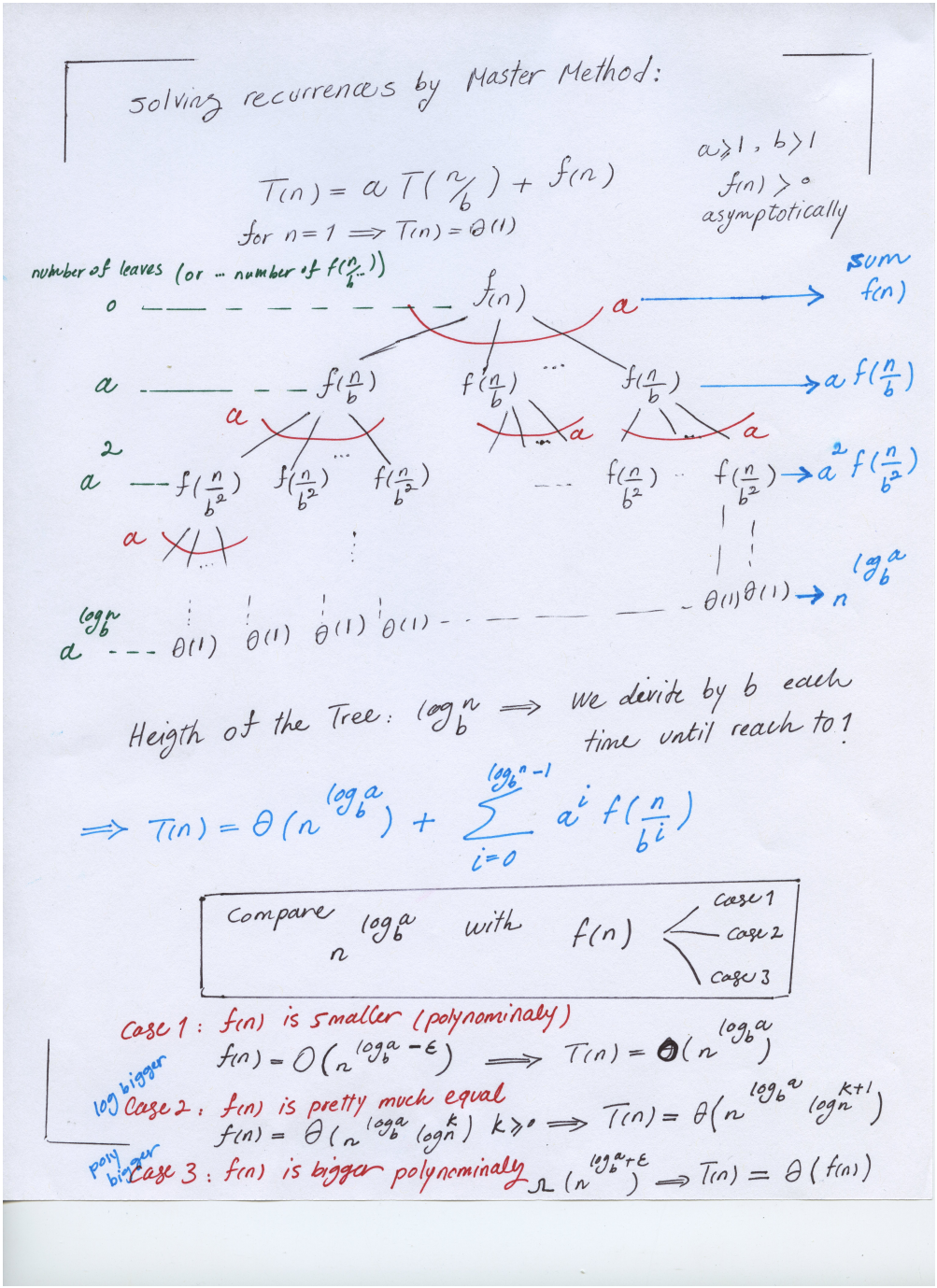

Figure 1: Sketch of recursion tree for Master method proof.

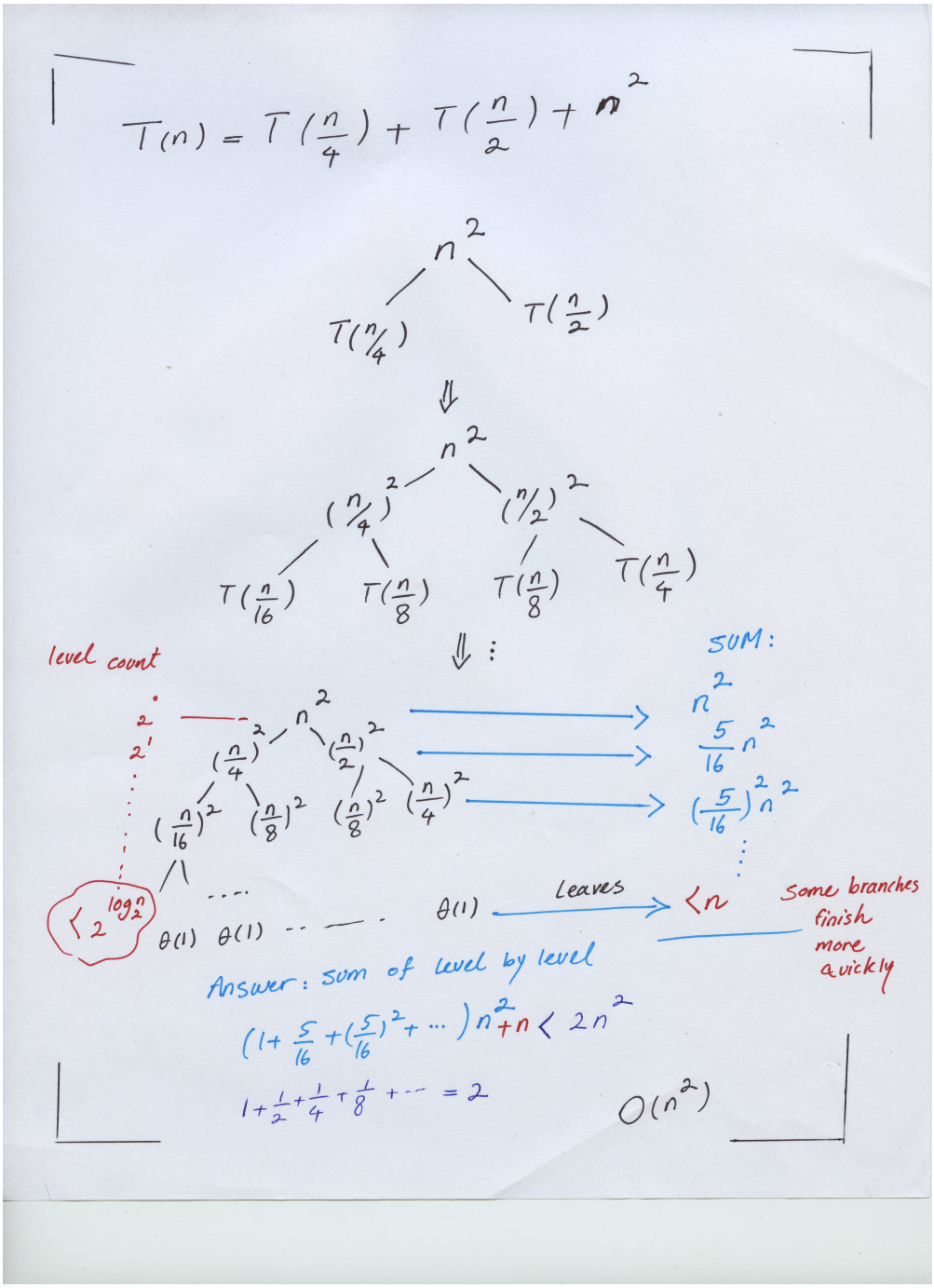

Figure 2: Recursion tree.

1.3.1 Example

Time analysis for

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

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 $\forall k < n$:
   $$T(k) = 4T(\frac{k}{2}) + k \leq ck^3,$$
   We want to prove that for n:
   $$T(n) = 4T(\frac{n}{2}) + n \leq cn^3$$
   Proof:
   $\frac{n}{2} < n \rightarrow T(\frac{n}{2}) \leq c\frac{n^3}{8}$
   $\rightarrow 4T(\frac{n}{2}) + n \leq c\frac{n^3}{2} + n \leq cn^3 - (c\frac{n^3}{2} - n)$
   $\forall c \geq 2 : (c\frac{n^3}{2} - n) > 0 \rightarrow$
   $\forall c \geq 2 : 4T(\frac{n}{2}) + n \leq cn^3$
   □
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 $\forall k < n$:
   $$T(k) = 4T(\frac{k}{2}) + k \leq c_1k^2 - c_2k$$
   We want to prove that for n:
   $$T(n) = 4T(\frac{n}{2}) + n \leq c_1n^2 - c_2n$$
   Proof:
   $\frac{n}{2} < n \rightarrow T(\frac{n}{2}) \leq c_1\frac{n^2}{4} - c_2\frac{n}{2}$
   $\rightarrow 4T(\frac{n}{2}) + n \leq c_1n^2 - 2c_2n + n \leq c_1n^2 - c_2n - ((c_2 - 1)n)$
   $\forall c_2 \geq 1/2 : (c_2 - 1)n > 0 \rightarrow$
   $\forall c_2 \geq 1/2 : 4T(\frac{n}{2}) + n \leq c_1n^2 - c_2n$
   □
3. Depending on $\theta(1)$: c_1 should be sufficiently greater than c_2 and also $c_2 \geq 1/2$

1.3.2 Example

Time analysis for

$$T(n) = T(\frac{n}{5}) + T(\frac{4n}{5}) + \theta(n)$$

1. Figure 3 shows the recursion tree. I just guessed the answer, but I am not too sure. So we will use substitution method to prove it.

1. Lets see if it is $O(nlogn)$
2. Base case: $T(1) = \theta(1)d_1$ should be sufficiently greater than $\theta(1)$

   Note we never use θ or O in our induction proof.

   If $\forall k < n$:

   $$T(k) = T(\frac{k}{5}) + T(\frac{4k}{5}) + c_1 k \leq d_1 k \log k$$

   We want to prove that for n:

   $$T(n) = T(\frac{n}{5}) + T(\frac{4n}{5}) + c_1 n \leq d_1 n \log n$$

   Proof:

   $\frac{n}{5} < n \rightarrow T(\frac{n}{5}) \leq \frac{d_1 n \log n/5}{5}$, and

   $\frac{4n}{5} < n \rightarrow T(\frac{4n}{5}) \leq \frac{4d_1 n \log 4n/5}{5}$

   $\rightarrow T(n) \leq \frac{d_1 n}{5}(\log n - \log 5) + \frac{4d_1 n}{5}(\log n - \log 5/4) + c_1 n$

   $\rightarrow T(n) \leq d_1 n \log n - n(\frac{d_1}{5} \log 5 + \frac{4d_1}{5} \log 5/4 - c_1)$

   $(\frac{d_1}{5} \log 5 + \frac{4d_1}{5} \log 5/4 - c_1) \geq 0 \rightarrow d_1 > 5c_1(...)$

   □

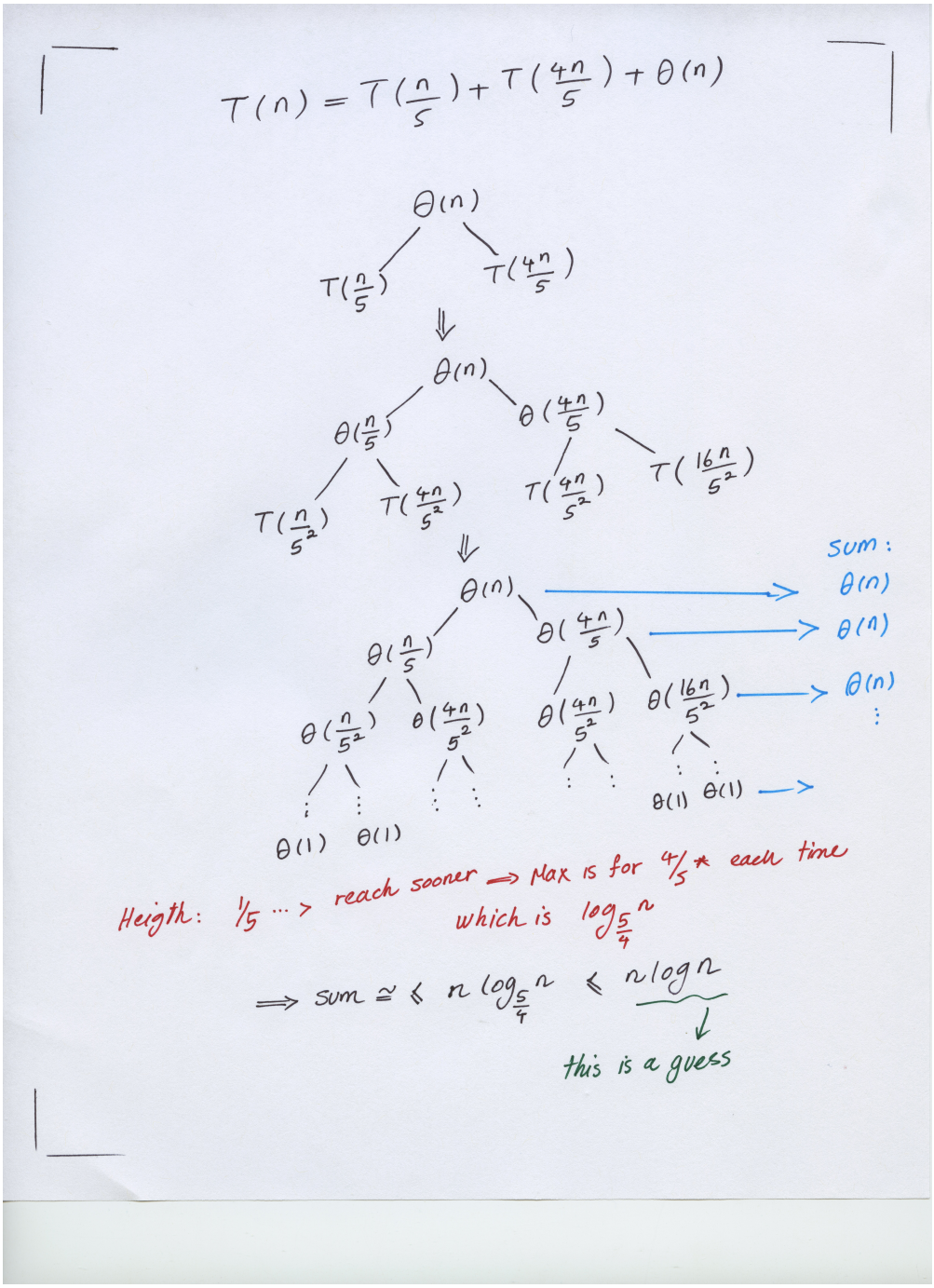

Figure 3: Recursion tree to guess the answer.