CS 430 Introduction to Algorithms – Fall 2017
Recitation Class Note 2
Xiaolang Wang

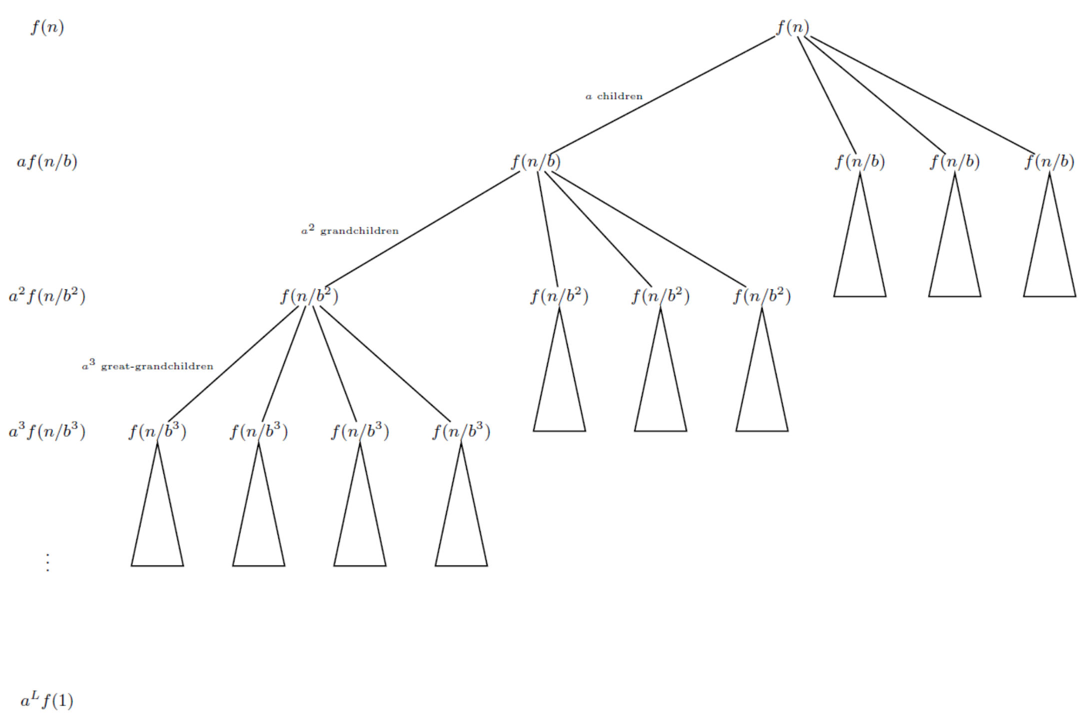
Solving Recurrences

1. **Techniques**
   - Divide and Conquer (recursion tree) – powerful
   - Master Method/Master Theorem – to solve “divide”
   - Annihilators – to solve “Minus”: $T(n) = T(n-a) + T(n-b) \ldots + f(n)$
   - Ultimate tech: Guess and Check!
2. **Master Theorem**

> **The Master Theorem.** The recurrence $T(n) = aT(n/b) + f(n)$ can be solved as follows.
> - If $a\,f(n/b)/f(n) < 1$, then $T(n) = \Theta(f(n))$.
> - If $a\,f(n/b)/f(n) > 1$, then $T(n) = \Theta(n^{\log_b a})$.
> - If $a\,f(n/b)/f(n) = 1$, then $T(n) = \Theta(f(n)\log_b n)$.
> - If none of these three cases apply, you're on your own.

   - It works when f(n) is polynomial
   - We use divide and conquer to prove it’s correct

3. **Using Divide and Conquer to prove Master Theorem**



   - Draw the recursion tree for $T(n) \;=\; f(n) + a\,T(n/b) \;+ f(n)$, the tree’s depth is $\log_b n$
   - $T(n) = f(n) + a\,f\left(\frac{n}{b}\right) + a^2\,f\left(\frac{n}{b^2}\right) + \cdots a^i\,f\left(\frac{n}{b^i}\right) + \cdots a^H\,f\left(\frac{n}{b^H}\right)$

- $H = \log_b n$, because on the last layer $\frac{n}{b^H} = 1$
- We consider this as a geometric series: because $f(n)$ is polynomial, and $f(n) = \Theta(n^k) \rightarrow common\ ratio = \Theta\left(\frac{a}{b^k}\right)$
- If $a\frac{f\left(\frac{n}{b}\right)}{f(n)} < 1$, or the common ratio is between 0 and 1; f (n) will be the largest term, so $\mathrm{T(n)} = \Theta(f(n))$.
- If $a\frac{f\left(\frac{n}{b}\right)}{f(n)} > 1$, or the common ratio is greater than 1, and the last term becomes the largest. So $\mathrm{T(n)} = \Theta(a^{\log_b n})$
- If $a\frac{f\left(\frac{n}{b}\right)}{f(n)} = 1$, we have a lot of f(n) here, so $\mathrm{T(n)} = \Theta(\log_b n\ f(n))$

4. Examples

a) Merge Sort $T(n) = 2T\left(\frac{n}{2}\right) + n$
   - $f(n)$ is polynomial, we can use Master
   - $a = 2, b = 2 \rightarrow \Theta(n \lg n)$

b) $T(n) = 4T\left(\frac{n}{2}\right) + n \lg n$
   - $f(n)$ is not polynomial, and $a\ f\left(\frac{n}{b}\right) = 2n \lg n - 2n$
   - We can say $2\ n \lg n > 2n \lg n - 2n > c_1\ n \lg n\ , c_1 > 1$ when n is really large.
   - Thus, we can use the second case in Master $\rightarrow \Theta\left(4^{\lg n}\right) = \Theta(n^2)$

c) $T(n) = T\left(\frac{3n}{4}\right) + T\left(\frac{n}{4}\right) + n$
   - Divide and Conquer
   - (Draw the tree) And we can see each complete level sum up to n.
   - Shortest branch has depth $\log_4 n$, longest has depth $\log_{\frac{4}{3}} n$.
   - $n \log_4 n \le T(n) \le n \log_{\frac{4}{3}} n \rightarrow \Theta(?)$

d) $T(n) = 2T\left(\frac{n}{2}\right) + n/\lg n$
   - $f(n)$ is not polynomial, and $a\ f\left(\frac{n}{b}\right) = \frac{n}{\lg n - 1}$
   - We can't say it's geometric, so let's try Divide and conquer
   - We can see each layer sum up to $\frac{n}{\lg n - i}$ for the i^{th} level
   - $H < \lg n - 1\ (\approx \lg n - 2)$
   - So it will sum up to $\Theta(?)$

e) Hanoi Tower $T(n) = 2T(n-1) + 1$
   - Divide and Conquer gives us $\Theta(2^n - 1)$
   - Talk a little bit about annihilator if there's time

f) $T(n) = \sqrt{n}\ \times T\left(\sqrt{n}\right) + n$
   - Let's guess! And check the upper bound and lower bound.
   - Check whether $b \times\ g(n) \le T(n) \le a \times\ g(n)$
   - Try $\Theta(n^2)$, first let's check if it's okay for upper bound:

Insert $T(n) = O(n^2)$, we have $T(n) \leq \sqrt{n} \times a \times n + n = a \times n\sqrt{n} + n \leq a \times n^2$. When n is large enough, we can find this constant a for sure, so n^2 is good for an upper bound.
Then let's check if it's okay for lower bound:
Insert $T(n) = \Omega(n^2)$, we have $T(n) \geq \sqrt{n} \times b \times n + n = b \times n\sqrt{n} + n \ngeq b \times n^2$. When n is large enough, we can't find this constant b for sure, so n^2 is not good for a lower bound, which means we need something grows more slowly.

- Let's try $\Theta(n)$:
$$T(n) \leq \sqrt{n} \times a \times \sqrt{n} + n = (a+1) \times n \nleq a \times n$$
We can see $\Theta(n)$ grows too slowly so need something faster.
- Maybe $\Theta(n \log n)$? Keep narrowing it down!