CS 305
Design and Analysis of Algorithms

09 / 19 / 2022

Instructor: Michael Eckmann

Today's Topics

- Questions/Comments?
- Logarithms
- Perfect a-ary tree work added up
- leads to T(n) formula where we have time at the leaves + time at the internal nodes

Recurrence Relations

- A divide and conquer algorithm that creates *a* subproblems each a factor of *1/b* the size of the original problem and takes *f(n)* amount of time to do the divide and combine steps.
- T(n) = a * T(n/b) + f(n)
    - Number of recursive calls is a
    - n/b is size of list in a recursive call
    - The time in a call is f(n)

Logarithms

FROM SECTION 3.3.

1. $b^{\log_b a} = a$

2. $\log_b(a \cdot c) = \log_b a + \log_b c$

3. $\log_b(a^x) = x \log_b a$ (follows from 2.)

4. $\log_b a = \dfrac{\log_c a}{\log_c b}$ for some other base c
for converting bases.

5. $\log_b\left(\frac{1}{a}\right) = \log_b 1 - \log_b a = -\log_b a$

Logarithms

- Let's go back and look at the a-ary tree and recall there are a^($\log_b n$) --- that is a to the power $\log_b n$
- Any other way to specify a^($\log_b n$) ?

Logarithms

- Let's go back and look at the a-ary tree and recall there are a^($\log_b n$) --- that is a to the power $\log_b n$
- Any other way to specify a^($\log_b n$) ?
- n^($\log_b a$)
- better because it is n to some constant power

Logarithms

FROM SECTION 3.3.

1. $b^{\log_b a} = a$

2. $\log_b(a \cdot c) = \log_b a + \log_b c$

3. $\log_b(a^x) = x \log_b a$ (follows from 2.)

4. $\log_b a = \dfrac{\log_c a}{\log_c b}$ for some other base c for converting bases.

5. $\log_b\left(\frac{1}{a}\right) = \log_b 1 - \log_b a = -\log_b a$

6. $a^{\log_b n} = n^{\log_b a}$

Master Method for Recurrences

- The Master Method can be used to solve most recurrences of the form: $T(n) = a * T(n/b) + f(n)$
- There are 3 cases to consider each based on a relationship of time spent at leaves vs. at the root.
- As an example, and to help us shortly, let's now count the number of nodes in a perfect tree, say a 3-ary tree.