CS 170

DISCUSSION 1!

AGENDA

1. Intro + forms
2. Quick asymptotics review
3. Recurrence relations + Master theorem

ASYMPTOTICS

2 commonly used definitions :

1. - $f(n) = O(g(n))$ if, for $n \to \infty$,
   $f(n) \le c \cdot g(n)$, for a constant c.

   - $f(n) = \Omega(g(n))$ if $f(n) \ge c \cdot g(n)$ as $n \to \infty$

   - $f(n) = \Theta(g(n))$ if $f(n) = c \cdot g(n)$ as $n \to \infty$.

   you can think of O (big-O) as $\le$, Ω (big-Omega) as $\ge$, and Θ (big-Theta) as $=$ (on the asymptotic/ limit level).

2. Limit definition (not necessary! but sufficient).

   - If $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$, then $f(n) = O(g(n))$

   - If $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$, then $f(n) = \Omega(g(n))$

   - If $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$ for some $c > 0$, then $f(n) = \Theta(g(n))$.

$\boxed{n^2 + n}$ $\quad O(n^2)$.

$n \to \infty$

n^2

$$\begin{cases} n^2 + n \le c \cdot n^2 \\ \text{as } n \to \infty \end{cases}$$

PRACTICE :

$n^{100} + \boxed{1.01^n}$

For each of the following, state the order of growth using Θ notation, e.g. $f(n) = \Theta(n)$.

(i) $f(n) = 50$ $\quad \Theta(1)$

(ii) $f(n) = n^2 - 2n + 3$ $\quad \Theta(n^2)$

(iii) $f(n) = n + \cdots + 3 + 2 + 1$ $\quad 1 + \cdots n = \frac{n(n+1)}{2} \quad \frac{n^2+n}{2} \quad \Theta(n^2)$

(iv) $f(n) = n^{100} + 1.01^n$ $\quad \Theta(1.01^n)$

(v) $f(n) = n^{1.1} + n \log n$ $\quad \Theta(n^{1.1}) \quad n \cdot n^{1/10} \quad n \cdot \log n$

(vi) $f(n) = (g(n))^2$ where $g(n) = (\sqrt{n} + 5)^2$ $\quad (\sqrt{n})^2 = n \quad \Theta(n)$

$n^{10000} < 1.00001^n$ $\quad$ desmos

$f(n) = \Omega(n^{100})$

$f(n) = \Omega(1.01^n)$ " tighter"

$\{$ log < polynomial

FORMS

tinyurl.com/manan-week1-attendance

tinyurl.com/manan-mailing-list

RECURRENCE RELATIONS

Divide-and-conquer problems generally follow the following paradigm:

1. Split the problem up into smaller parts
2. Make a recursive calls to problems of size n/b
3. "glue" the subproblems together to provide solution of original problem.

This formulation lends itself to the canonical recurrence relation:

$$T(n) = a\,T(n/b) + O(n^d)$$

$O(n^2)$

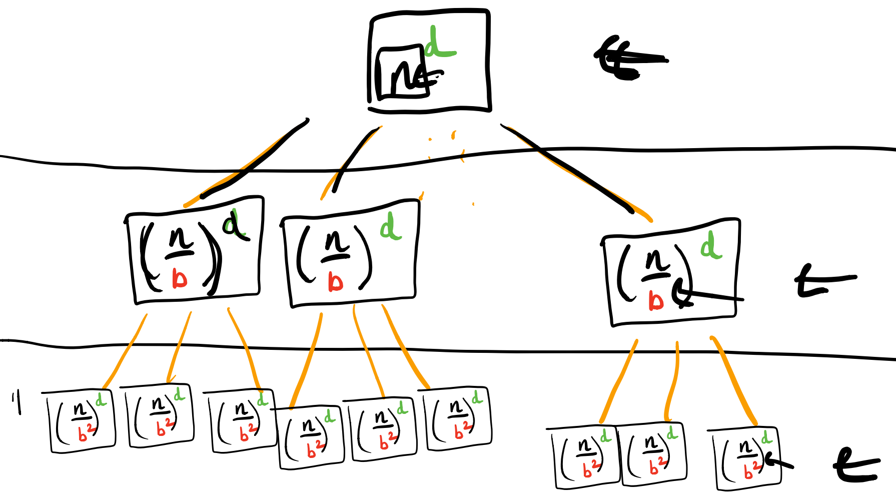
a ← branching factor

b ← factor by which subproblem size is reduced

d ← work done at each subproblem.

$O(n^d)$

$$\underline{T\left(\tfrac{n}{b}\right)} = \underline{a}\,T(n/b) + \boxed{O\left(\tfrac{n}{b}\right)^d} \leftarrow$$

Tree	Level	Nodes per level	Work per node per level
			
	0	$\underline{1}$ ×	$\underline{O(n^d)}$
	1	$\underline{a}$ ×	$O\left(\left(\frac{n}{b}\right)^d\right)$
	$\underline{2}$	$\underline{a^2}$ ×	$O\left(\left(\frac{n}{b^2}\right)^d\right)$

How many levels? Until subproblem size = 1, usually

$$\frac{n}{b^k} = 1 \qquad T(1)$$

$$n = b^k$$

$$k = \lfloor \log_b n \rfloor$$

<u>Total work</u> : Sum of work at each level :

$$T(n) = \underline{O(n^d)} \cdot \left(\underline{1} + \left(\frac{a}{b^d}\right) + \left(\frac{a}{b^d}\right)^2 + \dots + \left(\frac{a}{b^d}\right)^{\log_b n} \right)$$

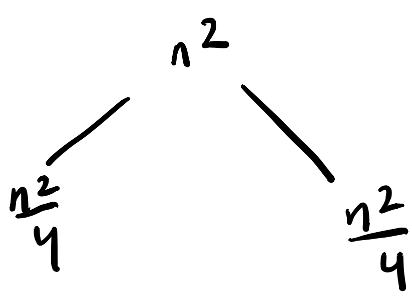
Key term : $\boxed{\frac{a}{b^d}}$

MASTER THEOREM

$$T(n) = a\,T(n/b) + O(n^d)$$

- If $\frac{a}{b^d} < 1$ ($d > \log_b a$), then $T(n) = O(n^d)$ (root heavy)
- If $\frac{a}{b^d} > 1$ ($d < \log_b a$), then $T(n) = O(n^{\log_b a})$ (leaf heavy)
- If $\frac{a}{b^d} = 1$ ($d = \log_b a$), then $T(n) = O(n^d \log n)$ (balanced)

$$T(n) = 2\,T(n/2) + O(n^2)$$



PRACTICE

(a) (i) $T(n) = 3T(n/4) + 4n$

(ii) $T(n) = 45T(n/3) + .1n^3$

$\log \log n$

$\left\lfloor \frac{1}{2^k} \right\rfloor =$

$n \to n^{1/2} \to n^{1/4} \to 1$

(b) $T(n) = 2T(\sqrt{n}) + 3$, and $T(2) = 3$.

Hint: Try repeatedly expanding the recurrence.

(can't use Master theorem!)