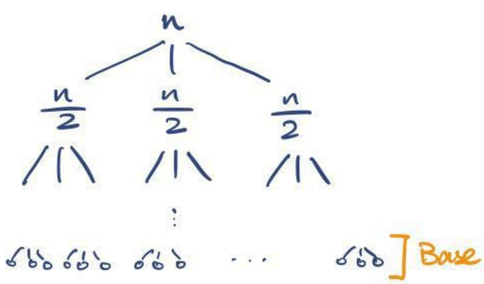
Lec 3 — Brick method balanced case & substitution method

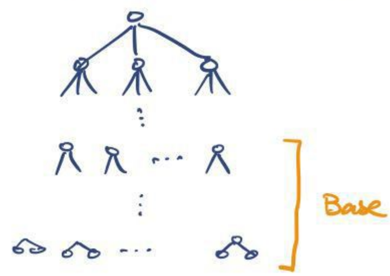
Brick method – more on leaf dominated case

Thm Suppose $\text{cost}(v) \leq \alpha \, \text{cost}(\text{children}(v))$ for all nodes v (↙ Sum of all children)
with input size greater than some n_0, $0 < \alpha < 1$.
Then the overall cost is $O(\text{cost}(\text{base of tree}))$

Ex. $W(n) = 3W(\frac{n}{2}) + n$



$$W(n) = \begin{cases} 3W(\frac{n}{2}) + n & n > 42 \\ 2W(n-1) + 1 & 1 < n \leq 42 \\ 1 & n \leq 1 \end{cases}$$

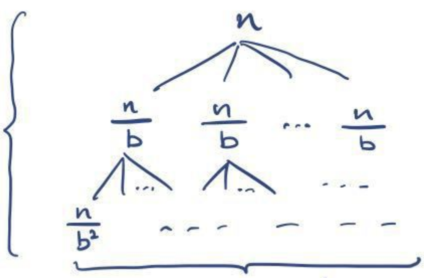


<u>Computing cost of base</u>

Usually … if leaves has $O(1)$ and root is leaves,
$\text{cost}(\text{base of tree}) = O(\#\text{ leaves})$

Ex. $W(n) = a\,W(\frac{n}{b}) + \cdots$

$N_{layers} = \log_b n$

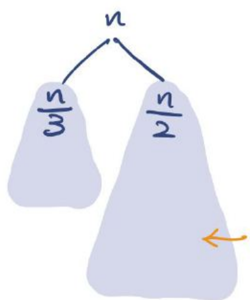


$$\begin{aligned} N_{leaves} &= a^{\log_b n} \\ &= n^{\log_b a} \\ &\in O(n^k) \end{aligned}$$

by $a^{\log_b c} = c^{\log_b a}$

But some cases are different…

Ex. $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$ …



← and leaves scattered

↑ But more things here

local cost: cost $(v_n) = \sqrt{n}$

cost (children (v_n)) $= \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}} = \sqrt{n}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}}\right)$

$= 1.284\sqrt{n}$

So increasing — leave dominated.

sometimes not the case

#leaves $L(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ L(\frac{n}{2}) + L(\frac{n}{3}) & \text{else} \end{cases}$

Substitution Method

— aka guess and check

guess: come up with some function

check: by induction

Ex. (cont) guess: $L(n) = n^b$ for some constant b

(BC) $L(n) = n^b = 1^b = 1$ for all b.

(IH) Assume for $0 \leq k < n$, $L(k) = k^b$.

(IS) $L(n) = L(\frac{n}{2}) + L(\frac{n}{3})$

$= (\frac{n}{2})^b + (\frac{n}{3})^b$ by IH

$= n^b \cdot \left(\frac{1}{2^b} + \frac{1}{3^b}\right)$

But if we want $n^b \cdot \left(\frac{1}{2^b} + \frac{1}{3^b}\right) = n^b$, we need:

$\frac{1}{2^b} + \frac{1}{3^b} = 1$

Wolfram alpha

$b \approx 0.788\ldots$

$\Rightarrow L(n) = n^{0.788\ldots}$

$W(n) \in O(n^{0.788\ldots})$ since leaves dominated

(if... say base case costs $O(m)$,

$W(n,m) \in O(mn^{0.788\ldots})$)

Brick method – balanced tree

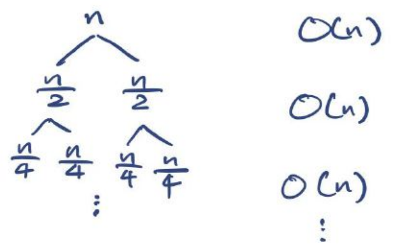
If work balanced, across levels:
└ asymptotically the same as imprecise definition

overall cost $\leq \max_{L_i}(\text{cost}(L_i)) \cdot \#\text{levels}$

highest cost level

Ex. Merge sort

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



$\#\text{level} \in O(\log n)$

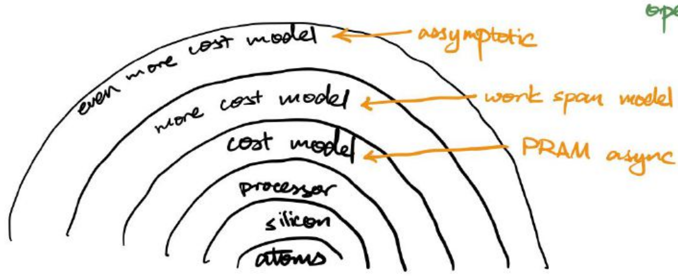
$\text{overall} \in O(n \log n)$

Note: not all recurrences fall in one of brick cases

Cost models

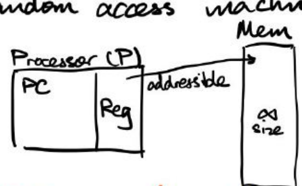
— another layer of abstraction,
the question of asymptotic what?

time?
operations?
⋮



Some types of models...

- Random access machine (RAM) model ← Good enough for writing better algorithm



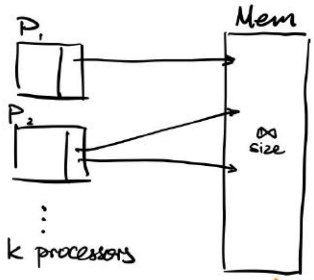
$O(1)$ instructions: read, write, add, multiply, jumps, conditionals ...

Sequential complexity in 122: #instructions on RAM model

Imperfection: read write may not be $O(1)$ ... (think cache)

- IO model: non-constant read/write cost

- RAM model but multiple processors



- P-RAM model : that but all processors run synchronously
  - P-RAM (W) : variant to allow write at same time
  - P-RAM (exclusive W) : -- disallow ---

Problems: how do we model and partition?
maybe possible, but messy to work with
also synchronization is costly to implement
↳ but asynchronous makes it even harder to program

- On top of async PRAM — Nested Parallel Work-Span Model

More like a language cost model than machine model

expressions		Work	Span
$e ::= x$	(var)	1	1
$\mid c$	(constant)	1	1
$\mid e_1 + e_2$		$W(e_1) + W(e_2) + 1$	$S(e_1) + S(e_2) + 1$
$\mid e_1 \parallel e_2$	(parallel)	$W(e_1) + W(e_2) + 1$	$\max(S(e_1), S(e_2)) + 1$
$\mid e_1 , e_2$	(sequential)	$W(e_1) + W(e_2) + 1$	$S(e_1) + S(e_2) + 1$