$\boxed{\text{Lec 2}}$ Asymptotic Analysis , Recurrances

Asymptotic Analysis

- useful abstraction
  - simplifies expression
  - avoid machine detail / programming lang
  - focus on details of algorithm

Def $f(n)$ asymptotically dominates $g(n)$ if $\exists c, n_0$ s.t.

$$g(n) \leq c \cdot f(n) \quad \forall n > n_0$$

Ex

	$f(n)$	$g(n)$	
	$2n$	n	Yes
	n	$2n$	Yes
	$n \lg n$	n	Yes
	2^n	$2^{1.1n}$	No

Notation
- $\lg$ means $\log_2$
- $O(f(n)) = \{ g \mid f \text{ asymptotically dominates } g \}$
- $\Omega(f(n)) = \{ g \mid g \text{ asymptotically dominates } f \}$
- $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$
- Notation abuses ... Really means ...
  - * $n = O(n^2)$ — $n \in O(n^2)$
  - * $f(n) = g(n) + O(n^2)$ — $f(n) \in \{ g(n) + h(n) \mid h(n) \in O(n^2) \}$
  - * $O(n) = O(n^2)$ — $O(n) \subseteq O(n^2)$
- $o(f(n)) = O(f(n)) \setminus \Theta(f(n))$
- $\omega(f(n)) = \Omega(f(n)) \setminus \Theta(f(n))$

Recurrences

- Base cases , recursive cases
- Modelling recurrent functions
- Harder to find closed form solution

```
msort(A) = if |A| ≤ 1 then A else
           let (L,R) = msort(A[0 ... |A|/2]),
                       msort(A[|A|/2 ... |A|])
           in merge (L,R) end
```

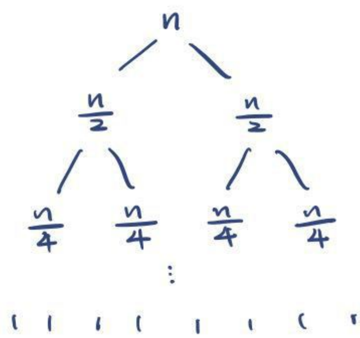
Convention : drop this trivial base case

$$W_{msort}(n) = \begin{cases} c_1 & \text{if } n \leq 1 \\ 2W(\frac{n}{2}) + W(n) + c_2 & \text{if } n > 1 \end{cases} \in O(n \lg n)$$

More abused notation: $W(n) = 2W(\frac{n}{2}) + O(n)$

Tree Method — Unfold level by level and sum them

msort



local costs

$O(n)$

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

$O(n)$

$\vdots$

$O(n)$

lg n levels

$$W(n) = \sum_{i=0}^{\lg n - 1} (c_1 n + 2^i c_2) \quad \ldots \in O(n \lg n)$$

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

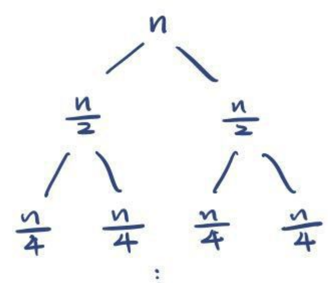
L_0

L_1

L_2

$\vdots$

L_i



$cost(L_1) = n^2$

$cost(L_2) = 2(\frac{n}{2})^2$

$cost(L_3) = 4(\frac{n}{4})^2$

$cost(L_i) = 2^i(\frac{n}{2^i})^2 = \frac{n^2}{2^i}$

Root dominated recurrances

$$W(n) = \underbrace{n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \frac{n^2}{8} + \cdots}_{\text{geometric decay}}$$
$$\in O(n^2)$$

other cases:
- leaf dominated
- balanced

Aside:

$1 + \alpha + \alpha^2 + \alpha^3 + \cdots$

$= \frac{1 - \alpha^{d+1}}{1 - \alpha}$ for $\alpha \neq 1$

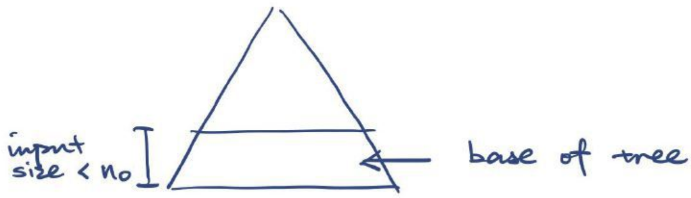
Brick Method

All children

Case 1: $cost(children(v)) \leq \alpha \, cost(v)$ $\forall$ node v and $\forall$ $0 < \alpha < 1$
then overall cost $\in O(cost(root))$

Proof: $cost(L_0) + \cdots + cost(L_d)$
$\leq cost(L_0) + \alpha \, cost(L_0) + \alpha^2 cost(L_0) + \alpha^d cost(L_0)$
$\leq \frac{1 - \alpha^{d+1}}{1 - \alpha} cost(L_0)$
$\leq \frac{1}{1 - \alpha} cost(L_0)$

Case 2: cost $(v) \leq \alpha$ cost (children (v)) for all nodes v with input size $> n_0$, $0 < \alpha < 1$.
then overall cost is $O($ cost (base of tree))



Suppose same input size for each level, then overall cost

$= \text{cost}(L_0) + \cdots + \text{cost}(L_{d-1}) + \text{cost}(L_d) + \text{cost}(\text{base of tree})$

$\leq \alpha^d \text{cost}(L_d) + \cdots + \alpha\, \text{cost}(L_d) + \text{cost}(L_d) + \text{cost}(\text{base of tree})$

Still need to compute this