Asymptotic Analysis and Recurrences

15-210 – Parallel and Sequential Data-structures and Algorithms

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

Asymptotic Analysis

Asymptotic Analysis: Motivation

Used throughout the curriculum:

- 15-122 Principles of Imperative Computation
- 15-251 Great Theoretical Ideas in Computer Science
- 15-150 Principles of Functional Programming
- 15-451 Algorithms

$$W(n) = 7n^2 + 3n \log n + 11\sqrt{n} + \frac{5}{\log n} + 2.72342142$$

Asymptotic Analysis: Motivation

Used throughout the curriculum:

- 15-122 Principles of Imperative Computation
- 15-251 Great Theoretical Ideas in Computer Science
- 15-150 Principles of Functional Programming
- 15-451 Algorithms

$$W(n) = 7n^2 + 3n \log n + 11\sqrt{n} + \frac{5}{\log n} + 2.72342142 \quad \in \quad \mathcal{O}(n^2)$$

Asymptotic Analysis: Motivation

Used throughout the curriculum:

- 15-122 Principles of Imperative Computation
- 15-251 Great Theoretical Ideas in Computer Science
- 15-150 Principles of Functional Programming
- 15-451 Algorithms

$$W(n) = 7n^2 + 3n\log n + 11\sqrt{n} + \frac{5}{\log n} + 2.72342142 \quad \in \quad \mathcal{O}(n^2)$$

Asymptotic analysis is a useful abstraction:

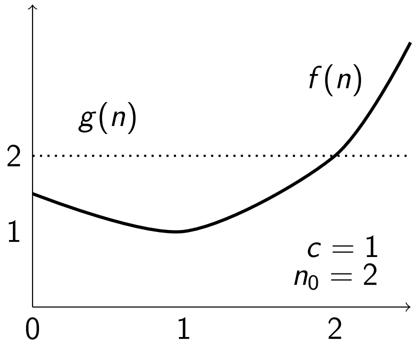
- Avoid details of the machine/model/compiler
- Avoid details of the algorithm
- Gives a way to compare algorithms in theory

we care about cost with large inputs

Asymptotic Analysis: Dominate

Definition

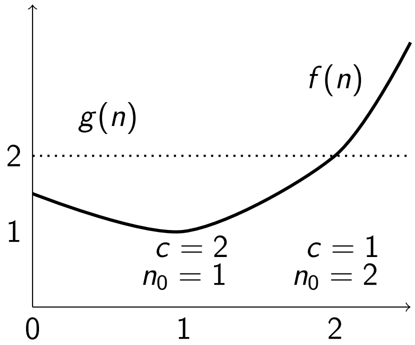
For two functions $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$



Asymptotic Analysis: Dominate

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$



Asymptotic Analysis: Dominate

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$

$f(n)$	$g(n)$
$2n$	n
n	$2n$
$n \log_2 n$	n
2^n	$2^{1.1n}$

Asymptotic Analysis: Dominate

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$

$f(n)$	$g(n)$
$2n$	n
n	$2n$
$n \log_2 n$	n
2^n	$2^{1.1n}$

c	n_0
1	0
2	0
1	2
×	×

Asymptotic Analysis: Big-$\mathcal{O}$, Big-Θ, and Big-Ω

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$

$$\mathcal{O}(f(n)) = \{g(n) \text{ s.t. } f(n) \text{ asymptotically dominates } g(n)\}$$

$$\Omega(f(n)) = \{g(n) \text{ s.t. } g(n) \text{ asymptotically dominates } f(n)\}$$

$$\Theta(f(n)) =$$

Asymptotic Analysis: Big-$\mathcal{O}$, Big-Θ, and Big-Ω

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$

$$\mathcal{O}(f(n)) = \{g(n) \text{ s.t. } f(n) \text{ asymptotically dominates } g(n)\}$$

$$\Omega(f(n)) = \{g(n) \text{ s.t. } g(n) \text{ asymptotically dominates } f(n)\}$$

$$\Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$$

Asymptotic Analysis: Big-$\mathcal{O}$, Big-Θ, and Big-Ω

Definition
For two functions $f, g : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ we say $f(n)$ asymptotically dominates $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$

$$\mathcal{O}(f(n)) = \{g(n) \text{ s.t. } f(n) \text{ asymptotically dominates } g(n)\}$$

$$\Omega(f(n)) = \{g(n) \text{ s.t. } g(n) \text{ asymptotically dominates } f(n)\}$$

$$\Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$$

$$o(f(n)) = \mathcal{O}(f(n)) \setminus \Theta(f(n))$$

$$\omega(f(n)) = \Omega(f(n)) \setminus \Theta(f(n))$$

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$
- $f(n)$ is $\mathcal{O}(n^2)$
- correct form: $f(n) \in \mathcal{O}(n^2)$

- $f(n) = g(n) + \mathcal{O}(n)$
- correct form: $f(n) \in g(n) + \mathcal{O}(n)$
- or equivalently $f(n) - g(n) \in \mathcal{O}(n)$

- $\mathcal{O}(n) = \mathcal{O}(n^2)$
- correct form: $\mathcal{O}(n) \subseteq \mathcal{O}(n^2)$

Proof that $\log(n!) = O(n \log n)$

Limit Theorem for Little-o and Little-ω

For positive functions f and g, the following are equivalent:

$$\begin{aligned} f(n) &= o(g(n)) \\ g(n) &= \omega(f(n)) \\ \lim_{n \to \infty} \frac{f(n)}{g(n)} &= 0 \end{aligned}$$

This is usually the easiest way to prove that one function is Little-o of another one.

Uses of the Limit Theorem (Exercises)

Use this theorem and l'Hôpital's rule to prove the following results:

$$n^k = o(\alpha^n) \quad \text{for any } k \text{ and any } \alpha > 1$$

In words this means: Any polynomial, no matter how big, is eventually dwarfed by any expontially growing function.

$$\log n = o(n^p) \quad \text{for any } p > 0$$

I.e. logs grow more slowly than any polynomial, even those of tiny degree.

Recurrences: Introduction

Recursive program with numeric values

Recurrences:

- base case(s) & recursive case(s)
- convenient for modeling costs
- derived from a recursive algorithm: abstract away details
- goal: find a closed form solution, at least asymptotically

Recurrences: Introduction

Recursive program with numeric values

Recurrences:

- base case(s) & recursive case(s)
- convenient for modeling costs
- derived from a recursive algorithm: abstract away details
- goal: find a closed form solution, at least asymptotically

Three methods to solve recurrences:

- Tree method
- Brick method
- Substitution method

Recurrences: Examples

$$F(n) = \begin{cases} n & \text{if } n \leq 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

Recurrences: Examples

$$F(n) = \begin{cases} n & \text{if } n \leq 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

$$= \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

with $\varphi = \frac{\sqrt{5}+1}{2}$

Recurrences: Examples

$$F(n) = \begin{cases} n & \text{if } n \leq 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

$$= \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

with $\varphi = \frac{\sqrt{5}+1}{2}$

$$\in \Theta(\varphi^n)$$

Recurrences: Examples

$$F(n) = \begin{cases} n & \text{if } n \leq 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases} = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

$$\text{with } \varphi = \frac{\sqrt{5}+1}{2}$$

$$\in \Theta(\varphi^n)$$

Recurrence for mergesort:

$$W(n) = \begin{array}{l} \text{if } (n \leq 1) \text{ then } c_1 \\ \text{else } 2W(\frac{n}{2}) + W_{\text{merge}}(n) + c_2 \end{array} \in \mathcal{O}(n \log_2 n)$$

Recurrences: Simplifications

First off all, since we're only doing asymptotic analysis we will assume that the value of the base case is is a constant denoted c_b.

Recurrences: Simplifications

First off all, since we're only doing asymptotic analysis we will assume that the value of the base case is is a constant denoted c_b.

Secondly many of the recurrences we want to solve involve integer parameters. For example, in the case of mergesort, we recurse on one part of size $\lceil n/2 \rceil$ and the other of size $\lfloor n/2 \rfloor$. But when we wrote the recurrence we just expressed this as $2W\left(\frac{n}{2}\right)$.

Recurrences: Simplifications

First off all, since we're only doing asymptotic analysis we will assume that the value of the base case is is a constant denoted c_b.

Secondly many of the recurrences we want to solve involve integer parameters. For example, in the case of mergesort, we recurse on one part of size $\lceil n/2 \rceil$ and the other of size $\lfloor n/2 \rfloor$. But when we wrote the recurrence we just expressed this as $2W\left(\frac{n}{2}\right)$.

We assert here without proof that this will not affect the asymptotic correceness of our analysis. Suffice it to say that this stems from the fact that for large n this change is miniscule, and that the realm of large n is where the preponderence of the recurrence is being computed.

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

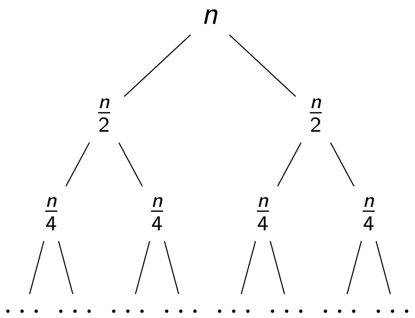
Substitution Method

Tree Method: Unfold Recurrence, Sum by Level

$$W(n) \;=\; 2W\left(\frac{n}{2}\right) + \mathcal{O}(n)$$

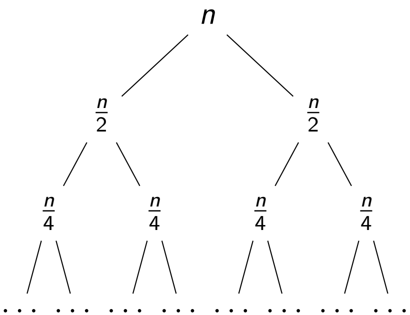
Tree Method: Unfold Recurrence, Sum by Level

$$\begin{aligned} W(n) &= 2W\left(\frac{n}{2}\right) + \mathcal{O}(n) \\ &= W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c1 \cdot n + c_2 \end{aligned}$$



Tree Method: Unfold Recurrence, Sum by Level

$$W(n) = 2W\left(\frac{n}{2}\right) + \mathcal{O}(n)$$
$$= W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c1 \cdot n + c_2$$



$$c_1 n + c_2$$

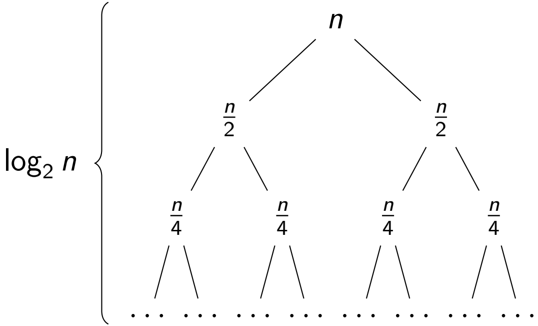
$$2(c_1 \tfrac{n}{2} + c_2) = c_1 n + 2c_2$$

$$4(c_1 \tfrac{n}{4} + c_2) = c_1 n + 4c_2$$

$$c_b \cdot n$$

Tree Method: Unfold Recurrence, Sum by Level

$$
\begin{aligned}
W(n) &= 2W\left(\frac{n}{2}\right) + \mathcal{O}(n) \\
&= W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c1 \cdot n + c_2
\end{aligned}
$$



$$c_1 n + c_2$$

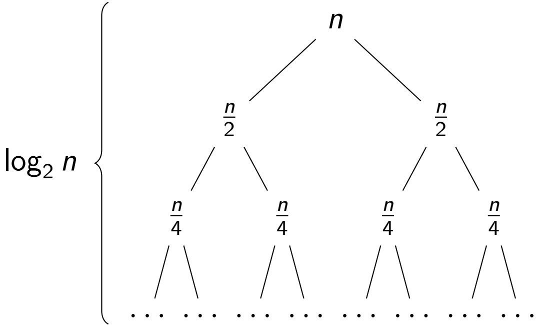
$$2(c_1 \tfrac{n}{2} + c_2) = c_1 n + 2c_2$$

$$4(c_1 \tfrac{n}{4} + c_2) = c_1 n + 4c_2$$

$$c_b \cdot n$$

Tree Method: Unfold Recurrence, Sum by Level

$$W(n) = 2W\left(\frac{n}{2}\right) + \mathcal{O}(n)$$
$$= W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c1 \cdot n + c_2$$



$$c_1 n + c_2$$

$$2(c_1 \tfrac{n}{2} + c_2) = c_1 n + 2c_2$$

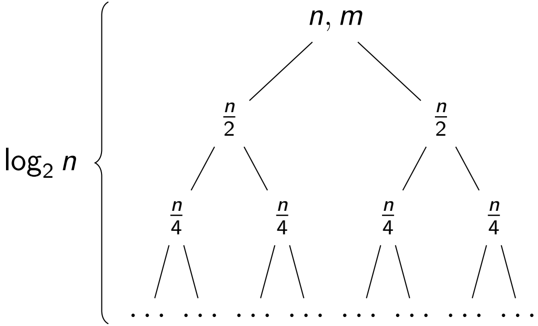
$$4(c_1 \tfrac{n}{4} + c_2) = c_1 n + 4c_2$$

$$c_b \cdot n$$

total cost is $c_1 n \log_2 n + c_2(n-1) + c_b n \in \mathcal{O}(n \log_2 n)$

Tree Method: Another Example

$$W(n) \quad = \quad W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + n^2$$



$$cost(L_0) = n^2$$

$$cost(L_1) = 2(\tfrac{n}{2})^2 = \frac{n^2}{2}$$

$$cost(L_2) = 4(\tfrac{n}{4})^2 = \frac{n^2}{4}$$

$$cost(L_d) = c_b \cdot n$$

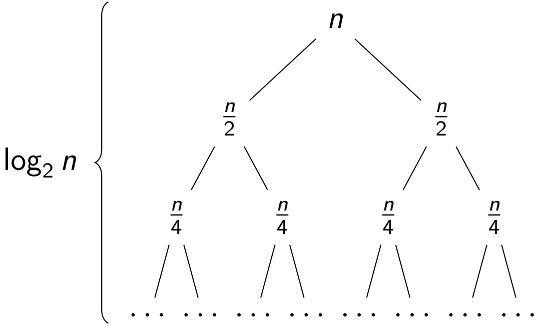
total cost is $n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \cdots < 2n^2 + c_b \cdot n \in \mathcal{O}(n^2)$

Tree Method: Unfold Recurrence, Sum by Level

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

Tree Method: Unfold Recurrence, Sum by Level

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



$$c_1\sqrt{n} + c_2$$

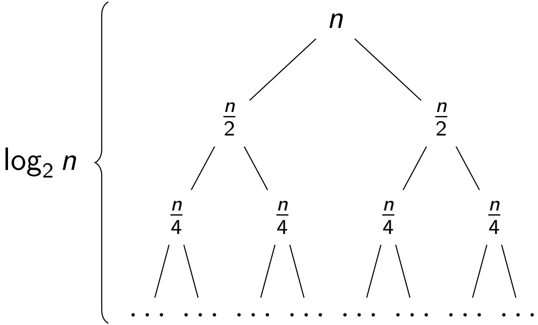
$$2c_1\sqrt{n/2} + 2c_2$$

$$4c_1\sqrt{n/4} + 4c_2$$

$$2^{\lg n}c_1\sqrt{n/2^{\lg n}} + 2^{\lg n}c_2$$

Tree Method: Unfold Recurrence, Sum by Level

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



$$c_1\sqrt{n} + c_2$$

$$2c_1\sqrt{n/2} + 2c_2$$

$$4c_1\sqrt{n/4} + 4c_2$$

$$2^{\lg n}c_1\sqrt{n/2^{\lg n}} + 2^{\lg n}c_2$$

total cost is $\mathcal{O}(n)$

Brick Method (An extension of the Tree Method): Introduction

Consider geometric series

$$S = \langle 1, \alpha, \alpha^2, \ldots, \alpha^n \rangle \qquad \text{with } \alpha \neq 1$$

$$\sum_{x \in S} x =$$

Brick Method (An extension of the Tree Method): Introduction

Consider geometric series

$$S = \langle 1, \alpha, \alpha^2, \ldots, \alpha^n \rangle \qquad \text{with } \alpha \neq 1$$

$$\sum_{x \in S} x = \frac{\alpha^{n+1} - 1}{\alpha - 1}$$

$$\text{For } \alpha > 1, \sum_{x \in S} x < \left(\frac{\alpha}{\alpha - 1} \right) \alpha^n$$

$$\text{For } \alpha < 1, \sum_{x \in S} x < \left(\frac{1}{1 - \alpha} \right)$$

Brick Method: Introduction

Consider recurrence tree, for any node v

- $C(v) =$ cost of v
- $D(v) =$ set of children of v

Root dominated:

- $C(v) \geq \alpha \sum_{u \in D(v)} C(u)$ for all v with $\alpha > 1$
- total cost is $\left(\frac{\alpha}{\alpha-1}\right) C(root) \in \mathcal{O}(C(root))$

Brick Method: Introduction

Consider recurrence tree, for any node v

- $C(v) =$ cost of v
- $D(v) =$ set of children of v

Leaf dominated:

- $\alpha C(v) \leq \sum_{u \in D(v)} C(u)$ for all v with $\alpha > 1$
- total cost is the cost of leaves $\in \mathcal{O}(C(leaves))$

Brick Method: Root Dominated Examples

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

- cost root: n^2
- cost children: $(\frac{n}{2})^2 + (\frac{n}{2})^2 = \frac{n^2}{2}$
- cost root $\geq$ 2 cost children $\Rightarrow$ root dominated: $\mathcal{O}(n^2)$
- applies at all nodes

Brick Method: Root Dominated Examples

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

- cost root: n^2
- cost children: $(\frac{n}{2})^2 + (\frac{n}{2})^2 = \frac{n^2}{2}$
- cost root $\geq$ 2 cost children $\Rightarrow$ root dominated: $\mathcal{O}(n^2)$
- applies at all nodes

$W(n) = W(\frac{n}{3}) + W(\frac{5n}{6}) + n^2$

- cost root: n^2
- cost children: $(\frac{n}{3})^2 + (\frac{5n}{6})^2 = \frac{n^2}{9} + \frac{25n^2}{36} = \frac{29n^2}{36}$
- cost root $\geq$ 2 cost children $\Rightarrow$ root dominated: $\mathcal{O}(n^2)$

Brick Method: Root Dominated Proof

$$S = \langle 1, \alpha, \alpha^2, \ldots, \alpha^n \rangle \qquad \text{with } \alpha \neq 1$$

$$\sum_{x \in S} x = \frac{\alpha^{n+1} - 1}{\alpha - 1}, \quad \sum_{x \in S} x < \frac{1}{1 - \alpha} \text{ with } 0 < \alpha < 1$$

Theorem

If $C(v) \geq \alpha \sum_{u \in D(v)} C(u)$ for all v with $\alpha > 1$, then the total cost is $\mathcal{O}(C(root))$.

Brick Method: Root Dominated Proof

$$S = \langle 1, \alpha, \alpha^2, \ldots, \alpha^n \rangle \qquad \text{with } \alpha \neq 1$$

$$\sum_{x \in S} x = \frac{\alpha^{n+1} - 1}{\alpha - 1}, \quad \sum_{x \in S} x < \frac{1}{1-\alpha} \text{ with } 0 < \alpha < 1$$

Theorem

If $C(v) \geq \alpha \sum_{u \in D(v)} C(u)$ for all v with $\alpha > 1$, then the total cost is $\mathcal{O}(C(root))$.

Proof.

$$\begin{aligned} \text{total C} &= C(L_0) + C(L_1) + \cdots + C(L_d) \\ &= C(L_0)(1 + 1/\alpha + \cdots + 1/\alpha^d) \\ &\leq C(L_0)\left(\frac{1}{1 - 1/\alpha}\right) = C(L_0)\left(\frac{\alpha}{\alpha - 1}\right) \in \mathcal{O}(C(L_0)) \end{aligned}$$

Brick Method: Leaf Dominated Examples

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

- cost root: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \sqrt{2}\sqrt{n}$
- cost of leaves: $2^{\log_2 n} = n$
- α cost node $\leq$ cost children $\Rightarrow$ leaf dominated: $\mathcal{O}(n)$

Brick Method: Leaf Dominated Examples

$$W(n) = W(\tfrac{n}{2}) + W(\tfrac{n}{2}) + \sqrt{n}$$

- cost root: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \sqrt{2}\sqrt{n}$
- cost of leaves: $2^{\log_2 n} = n$
- α cost node $\leq$ cost children $\Rightarrow$ leaf dominated: $\mathcal{O}(n)$

$$W(n) = W(\tfrac{n}{2}) + W(\tfrac{n}{2}) + W(\tfrac{n}{2}) + \sqrt{n}$$

- cost node: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \frac{3}{\sqrt{2}}\sqrt{n}$
- cost of leaves: $3^{\log_2 n} = n^{\log_2 3}$
- α cost node $\leq$ cost children $\Rightarrow$ leaf dominated: $\mathcal{O}(n^{\log_2 3})$

Brick Method: Leaf Dominated Example

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

- cost node: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \frac{3}{\sqrt{2}}\sqrt{n}$
- cost of leaves: $3^{\log_2 n} = n^{\log_2 3}$
- α cost node $\leq$ cost children $\Rightarrow$ leaf dominated: $\mathcal{O}(n^{\log_2 3})$

Brick Method: Leaf Dominated Proof

$$S = \langle 1, \alpha, \alpha^2, \ldots, \alpha^n \rangle \qquad \text{with } \alpha \neq 1$$

$$\sum_{x \in S} x = \frac{\alpha^{n+1} - 1}{\alpha - 1}, \quad \sum_{x \in S} x < \frac{1}{1 - \alpha} \text{ with } 0 < \alpha < 1$$

Theorem

If $C(v) \leq \frac{1}{\alpha} \cdot \sum_{u \in D(v)} C(u)$ for all v with $\alpha > 1$, then the total cost is $\mathcal{O}(C(leaves))$.

Brick Method: Leaf Dominated Proof

$$S = \langle 1, \alpha, \alpha^2, \dots, \alpha^n \rangle \qquad \text{with } \alpha \neq 1$$

$$\sum_{x \in S} x = \frac{\alpha^{n+1} - 1}{\alpha - 1}, \quad \sum_{x \in S} x < \frac{1}{1 - \alpha} \text{ with } 0 < \alpha < 1$$

Theorem

If $C(v) \leq \frac{1}{\alpha} \cdot \sum_{u \in D(v)} C(u)$ for all v with $\alpha > 1$, then the total cost is $\mathcal{O}(C(leaves))$.

Proof.

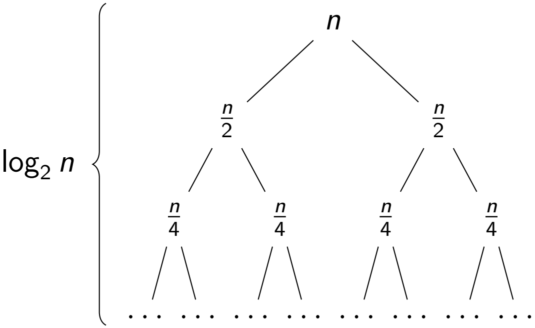
$$\begin{aligned}
\text{total cost} &= C(L_0) + C(L_1) + \cdots + C(L_d) \\
&\leq 1/\alpha^d \cdot C(L_d) + 1/\alpha^{d-1} \cdot C(L_d) + \cdots + C(L_d) \\
&= C(L_d)(1 + 1/\alpha + \cdots + 1/\alpha^d) \\
&\leq C(L_d)\left(\frac{\alpha}{\alpha - 1}\right) \in \mathcal{O}(C(L_d))
\end{aligned}$$

Brick Method: Balanced

The costs of each level are approximately the same

Neither leaf nor roof dominated

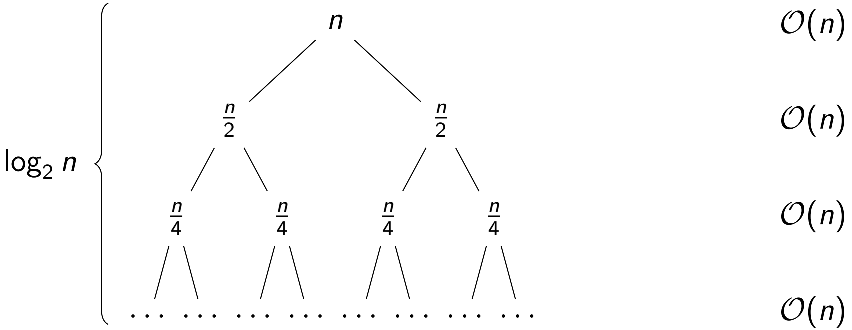
For example in mergesort: $W(n) = 2W(\frac{n}{2}) + \mathcal{O}(n)$



Brick Method: Balanced

The costs of each level are approximately the same
Neither leaf nor roof dominated

For example in mergesort: $W(n) = 2W(\frac{n}{2}) + \mathcal{O}(n)$



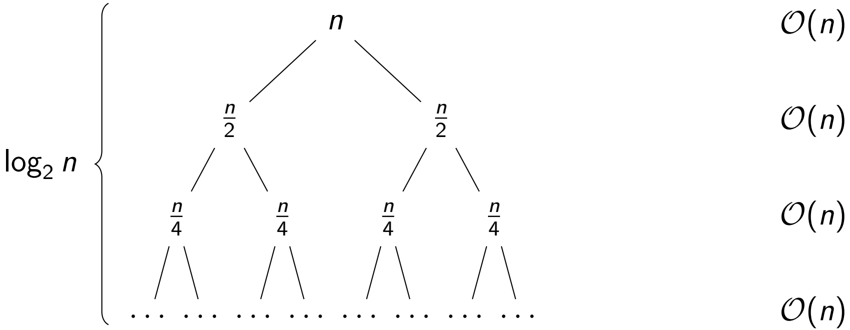
total cost is

Brick Method: Balanced

The costs of each level are approximately the same

Neither leaf nor roof dominated

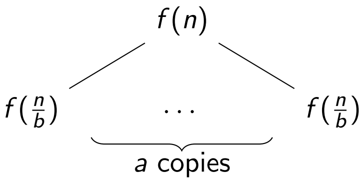
For example in mergesort: $W(n) = 2W(\frac{n}{2}) + \mathcal{O}(n)$



total cost is $\log_2 n \cdot \mathcal{O}(n) = \mathcal{O}(n \log_2 n)$

Brick Method "Masterform"

$$W(n) = a \cdot W\left(\frac{n}{b}\right) + f(n)$$



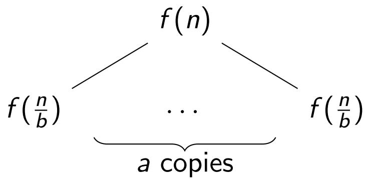
compare: $f(n) : a \cdot f\left(\frac{n}{b}\right)$

- $>$ root dominated
- $<$ leaf dominated
- $=$ balanced

$\#$ leaves $= a^{\log_b n} = n^{\log_b a}$

Brick Method "Masterform"

$$W(n) = a \cdot W\left(\frac{n}{b}\right) + f(n)$$



compare: $f(n) : a \cdot f\left(\frac{n}{b}\right)$

$>$ root dominated

$<$ leaf dominated

$=$ balanced

$\#$ leaves $= a^{\log_b n} = n^{\log_b a}$

The techniques described in this lecture allow you to derive the result of the "Master Theorem" whenever necessary.

Substitution Method: "Guess and Check"

Computing can be tricky if tree is unbalanced

- $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$

Substitution Method: "Guess and Check"

Computing can be tricky if tree is unbalanced

- $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$
- This recurrence is leaf-dominated as $\sqrt{n} < \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}}$
- How many leaf nodes?

Substitution Method: "Guess and Check"

Computing can be tricky if tree is unbalanced

- $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$
- This recurrence is leaf-dominated as $\sqrt{n} < \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}}$
- How many leaf nodes? New recurrence: $L(n) = L(\frac{n}{2}) + L(\frac{n}{3})$

Substitution Method: "Guess and Check"

Computing can be tricky if tree is unbalanced

- $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$
- This recurrence is leaf-dominated as $\sqrt{n} < \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}}$
- How many leaf nodes? New recurrence: $L(n) = L(\frac{n}{2}) + L(\frac{n}{3})$

The substitution method consists of two steps

- (educated) guess: good luck... intuition
- check: proof by induction

Our guess: $L(n) = n^b$ for some b

- base case: $L(1) = 1 = 1^b$
- induction: $n^b = \left(\frac{n}{2}\right)^b + \left(\frac{n}{3}\right)^b$

Substitution Method: "Guess and Check"

Computing can be tricky if tree is unbalanced

- $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$
- This recurrence is leaf-dominated as $\sqrt{n} < \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}}$
- How many leaf nodes? New recurrence: $L(n) = L(\frac{n}{2}) + L(\frac{n}{3})$

The substitution method consists of two steps

- (educated) guess: good luck... intuition
- check: proof by induction

Our guess: $L(n) = n^b$ for some b

- base case: $L(1) = 1 = 1^b$
- induction: $n^b = \left(\frac{n}{2}\right)^b + \left(\frac{n}{3}\right)^b$
- after simplification (dividing by n^b): $1 = \left(\frac{1}{2}\right)^b + \left(\frac{1}{3}\right)^b$

Substitution Method: "Guess and Check"

Computing can be tricky if tree is unbalanced

- $W(n) = W(\frac{n}{2}) + W(\frac{n}{3}) + \sqrt{n}$
- This recurrence is leaf-dominated as $\sqrt{n} < \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{3}}$
- How many leaf nodes? New recurrence: $L(n) = L(\frac{n}{2}) + L(\frac{n}{3})$

The substitution method consists of two steps

- (educated) guess: good luck... intuition
- check: proof by induction

Our guess: $L(n) = n^b$ for some b

- base case: $L(1) = 1 = 1^b$
- induction: $n^b = \left(\frac{n}{2}\right)^b + \left(\frac{n}{3}\right)^b$
- after simplification (dividing by n^b): $1 = \left(\frac{1}{2}\right)^b + \left(\frac{1}{3}\right)^b$
- solution: $b \approx .788$, so $L(n) \approx n^{.788}$ and $W(n) \in \mathcal{O}(n^{.788})$

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method