

Asymptotic Analysis and Recurrences

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

big- O

and Ω , Θ , o and ω .

What is this?

```
fun f 0 = 0  
    | f 1 = 1  
    | f n = f(n - 1) + f(n - 2);
```

What is this?

```
fun f 0 = 0
    | f 1 = 1
    | f n = f(n - 1) + f(n - 2);
```

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

What is this?

```
fun f 0 = 0
    | f 1 = 1
    | f n = f(n - 1) + f(n - 2);
```

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

Recurrences: recursive functions of type $\mathbb{R} \rightarrow \mathbb{R}$

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

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

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

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

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

Asymptotic analysis is a useful abstraction:

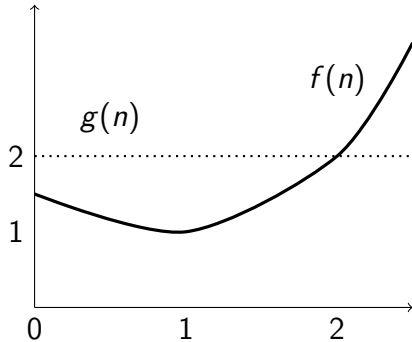
- Avoid details of the machine/model/compiler
- Avoid details of the algorithm
- Gives a way to compare algorithms in theory

we care about cost with large inputs

Asymptotic Analysis: Dominate

Definition

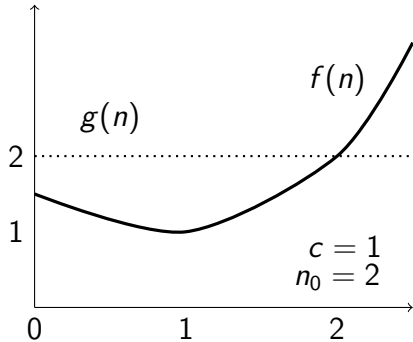
For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ we say $f(n)$ **asymptotically dominates** $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$



Asymptotic Analysis: Dominate

Definition

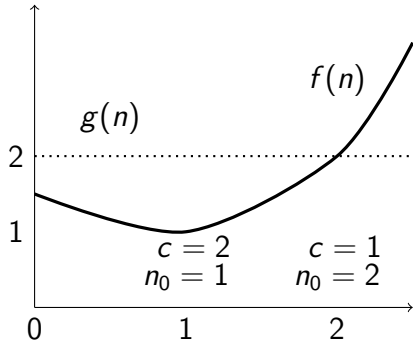
For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ we say $f(n)$ **asymptotically dominates** $g(n)$ if there exists positive constants c and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$



Asymptotic Analysis: Dominate

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ 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 $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ 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 $n \geq n_0$

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

Asymptotic Analysis: Dominate

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ 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 $n \geq n_0$

$f(n)$	$g(n)$	c	n_0
$2n$	n	1	0
n	$2n$		
$n \log_2 n$	n		
2^n	$2^{1.1n}$		

Asymptotic Analysis: Dominate

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ 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 $n \geq n_0$

$f(n)$	$g(n)$	c	n_0
$2n$	n	1	0
n	$2n$	2	0
$n \log_2 n$	n		
2^n	$2^{1.1n}$		

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 $n \geq n_0$

$f(n)$	$g(n)$	c	n_0
$2n$	n	1	0
n	$2n$	2	0
$n \log_2 n$	n	1	2
2^n	$2^{1.1n}$		

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 $n \geq n_0$

$f(n)$	$g(n)$	c	n_0
$2n$	n	1	0
n	$2n$	2	0
$n \log_2 n$	n	1	2
2^n	$2^{1.1n}$	×	×

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 $n \geq n_0$

Could write this as a relation $g(n) \leq_{ad} f(n)$.

Does it define a total order?

What about a partial order?

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 $n \geq n_0$

Could write this as a relation $g(n) \leq_{ad} f(n)$.

Does it define a total order?

What about a partial order?

No, and not quite.

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 $n \geq n_0$

If $f(n)$ is always growing (technically never shrinks, then asymptotically dominates implies there exists positive constants c_1 and c_2 such that

$$g(n) \leq c_1 f(n) + c_2$$

exercise, figure out the values

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

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 $n \geq n_0$

$$\mathbf{big-O} : \mathcal{O}(f(n)) = \{g(n) \mid f(n) \text{ asymptotically dominates } g(n)\}$$

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

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 $n \geq n_0$

$$\mathbf{big-O} : \mathcal{O}(f(n)) = \{g(n) \mid f(n) \text{ asymptotically dominates } g(n)\}$$

$$\mathbf{big-omega} : \Omega(f(n)) = \{g(n) \mid g(n) \text{ asymptotically dominates } f(n)\}$$

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

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 $n \geq n_0$

$$\mathbf{big-O} : \mathcal{O}(f(n)) = \{g(n) \mid f(n) \text{ asymptotically dominates } g(n)\}$$

$$\mathbf{big-omega} : \Omega(f(n)) = \{g(n) \mid g(n) \text{ asymptotically dominates } f(n)\}$$

$$\mathbf{big-theta} : \Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$$

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

Definition

For two functions $f, g : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ 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 $n \geq n_0$

$$\mathbf{big-O} : \mathcal{O}(f(n)) = \{g(n) \mid f(n) \text{ asymptotically dominates } g(n)\}$$

$$\mathbf{big-omega} : \Omega(f(n)) = \{g(n) \mid g(n) \text{ asymptotically dominates } f(n)\}$$

$$\mathbf{big-theta} : \Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$$

$$\mathbf{little-o} : o(f(n)) = \mathcal{O}(f(n)) \setminus \Theta(f(n))$$

$$\mathbf{little-omega} : \omega(f(n)) = \Omega(f(n)) \setminus \Theta(f(n))$$

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$ (does it typecheck?)

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$ (does it typecheck?) ❌
- correct form: $f(n) \in \mathcal{O}(n^2)$
- ok: $f(n)$ is $\mathcal{O}(n^2)$

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$ (does it typecheck?) **X**
- correct form: $f(n) \in \mathcal{O}(n^2)$
- ok: $f(n)$ is $\mathcal{O}(n^2)$

- ok shorthand: $f(n) \in g(n) + \mathcal{O}(n)$ (does it typecheck?)

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$ (does it typecheck?) **X**
- correct form: $f(n) \in \mathcal{O}(n^2)$
- ok: $f(n)$ is $\mathcal{O}(n^2)$

- ok shorthand: $f(n) \in g(n) + \mathcal{O}(n)$ (does it typecheck?)
- precise form $f(n) \in \{g(n) + h(n) : h(n) \in \mathcal{O}(n)\}$
- often substitute with: $f(n) \leq g(n) + c_1n + c_2$

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$ (does it typecheck?) ❌
- correct form: $f(n) \in \mathcal{O}(n^2)$
- ok: $f(n)$ is $\mathcal{O}(n^2)$

- ok shorthand: $f(n) \in g(n) + \mathcal{O}(n)$ (does it typecheck?)
- precise form $f(n) \in \{g(n) + h(n) : h(n) \in \mathcal{O}(n)\}$
- often substitute with: $f(n) \leq g(n) + c_1n + c_2$

- $\mathcal{O}(n) = \mathcal{O}(n^2)$ (is this true?)

Asymptotic Analysis: Conventions

- $f(n) = \mathcal{O}(n^2)$ (does it typecheck?) **X**
- correct form: $f(n) \in \mathcal{O}(n^2)$
- ok: $f(n)$ is $\mathcal{O}(n^2)$

- ok shorthand: $f(n) \in g(n) + \mathcal{O}(n)$ (does it typecheck?)
- precise form $f(n) \in \{g(n) + h(n) : h(n) \in \mathcal{O}(n)\}$
- often substitute with: $f(n) \leq g(n) + c_1n + c_2$

- $\mathcal{O}(n) = \mathcal{O}(n^2)$ (is this true?) **X**
- probably meant: $\mathcal{O}(n) \subseteq \mathcal{O}(n^2)$

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

Recurrences: Introduction

Recursive program with numeric values

Recurrences:

- base case(s) & recursive case(s)
- convenient for modeling costs
e.g. work, span, time, space
- often 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
e.g. work, span, time, space
- often 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

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

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

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

Recurrences: Examples

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

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

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

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

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

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

Recurrence for mergesort:

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

Recurrences: Examples

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

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

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

Recurrence for mergesort:

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

note: $\lg n$ is shorthand for $\log_2 n$

Recurrences: Simplifications

First off all, since we're only doing asymptotic analysis we will assume that the value of the base case is 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 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 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 correctness 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 preponderance of the recurrence is being computed.

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

Tree



Tree in the real world

Tree



Tree in the real world



Tree in the computer science world

Tree Method: Unfold Recurrence, Sum by Level

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

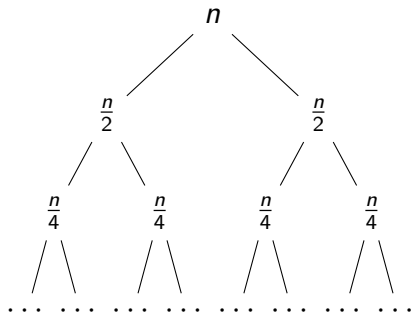
Mergesort Recurrence

Tree Method: Unfold Recurrence, Sum by Level

$$W(n) \leq \begin{cases} c_b & \text{if } n \leq 1 \\ W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c_1 \cdot n + c_2 & \text{otherwise} \end{cases}$$

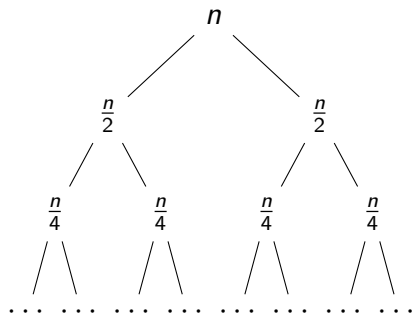
Tree Method: Unfold Recurrence, Sum by Level

$$W(n) \leq \begin{cases} c_b & \text{if } n \leq 1 \\ W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c_1 \cdot n + c_2 & \text{otherwise} \end{cases}$$



Tree Method: Unfold Recurrence, Sum by Level

$$W(n) \leq \begin{cases} c_b & \text{if } n \leq 1 \\ W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c_1 \cdot n + c_2 & \text{otherwise} \end{cases}$$



$$c_1 n + c_2$$

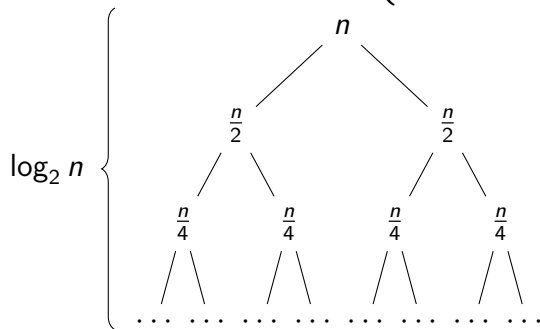
$$2\left(c_1 \frac{n}{2} + c_2\right) = c_1 n + 2c_2$$

$$4\left(c_1 \frac{n}{4} + c_2\right) = c_1 n + 4c_2$$

$$c_b \cdot n$$

Tree Method: Unfold Recurrence, Sum by Level

$$W(n) \leq \begin{cases} c_b & \text{if } n \leq 1 \\ W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c_1 \cdot n + c_2 & \text{otherwise} \end{cases}$$



$$c_1 n + c_2$$

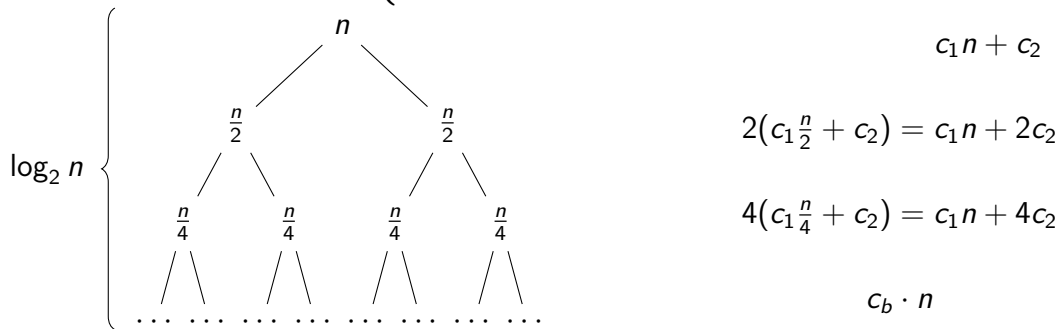
$$2\left(c_1 \frac{n}{2} + c_2\right) = c_1 n + 2c_2$$

$$4\left(c_1 \frac{n}{4} + c_2\right) = c_1 n + 4c_2$$

$$c_b \cdot n$$

Tree Method: Unfold Recurrence, Sum by Level

$$W(n) \leq \begin{cases} c_b & \text{if } n \leq 1 \\ W\left(\frac{n}{2}\right) + W\left(\frac{n}{2}\right) + c_1 \cdot n + c_2 & \text{otherwise} \end{cases}$$



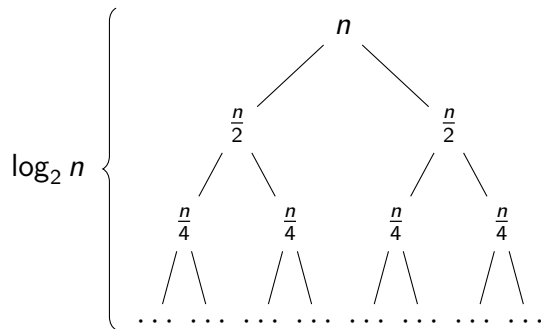
total cost is $c_1 n \log_2 n + c_2(n - 1) + c_b n \in \mathcal{O}(n \log_2 n)$

Tree Method: Another Example

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

Tree Method: Another Example

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



$$\text{cost}(L_0) = n^2$$

$$\text{cost}(L_1) = 2\left(\frac{n}{2}\right)^2 = \frac{n^2}{2}$$

$$\text{cost}(L_2) = 4\left(\frac{n}{4}\right)^2 = \frac{n^2}{4}$$

$$\text{cost}(L_d) = c_b \cdot n$$

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

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

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

The width of a brick is the work done at a tree node.

A brick from Home Depot:



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

The width of a brick is the work done at a tree node.

A brick from Home Depot:



Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$

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

The width of a brick is the work done at a tree node.

A brick from Home Depot:



Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot: 

Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot: 

Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot: 

Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot:



Root Dominated

Leaf Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$

e.g. $W(n) = 2W(\frac{n}{2}) + O(1)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot: 

Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



Leaf Dominated

e.g. $W(n) = 2W(\frac{n}{2}) + O(1)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot: 

Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



Leaf Dominated

e.g. $W(n) = 2W(\frac{n}{2}) + O(1)$



Balanced

e.g. $W(n) = 2W(n/2) + O(n)$



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

The width of a brick is the work done at a tree node.

A brick from Home Depot:



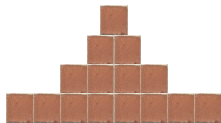
Root Dominated

e.g. $W(n) = 2W(n/2) + O(n^2)$



Leaf Dominated

e.g. $W(n) = 2W(\frac{n}{2}) + O(1)$



Balanced

e.g. $W(n) = 2W(n/2) + O(n)$



Often cost increases or decreases “geometrically”.

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

Consider geometric series

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

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

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

Consider geometric series

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

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

For $\alpha > 1$, $\sum_{x \in S} x < \left(\frac{\alpha}{\alpha - 1} \right) \alpha^n \in \boxed{O(\alpha^n)}$, dominated by last term

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

Consider geometric series

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

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

$$\text{For } \alpha > 1, \quad \sum_{x \in S} x < \left(\frac{\alpha}{\alpha - 1} \right) \alpha^n \in \boxed{O(\alpha^n)}, \quad \text{dominated by last term}$$

$$\text{For } \alpha < 1, \quad \sum_{x \in S} x < \left(\frac{1}{1 - \alpha} \right) \in \boxed{O(1)}, \quad \text{dominated by first term}$$

Brick Method: Root Dominated

Consider recurrence tree, for any node v

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

Root dominated:

- there exists a constant $\alpha > 1$
such that for all recursive calls from parent p :

$$C(p) \geq \alpha \sum_{c \in D(p)} C(c)$$

- total cost is $\left(\frac{\alpha}{\alpha-1}\right) C(\text{root}) \in \mathcal{O}(C(\text{root}))$

Shrinking



Brick Method: Root Dominated Examples

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

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

Brick Method: Root Dominated Examples

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

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

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

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

Brick Method: Root Dominated Proof

$$S = \langle 1, \alpha, \alpha^2, \dots, \alpha^n \rangle \quad \text{with } 0 < \alpha < 1$$

$$\sum_{x \in S} x < \frac{1}{1 - \alpha}$$

Theorem

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

Brick Method: Root Dominated Proof

$$S = \langle 1, \alpha, \alpha^2, \dots, \alpha^n \rangle \quad \text{with } 0 < \alpha < 1$$

$$\sum_{x \in S} x < \frac{1}{1 - \alpha}$$

Theorem

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

Proof.

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

Brick Method: Leaf Dominated

Consider recurrence tree, for any node v

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

Leaf dominated:

- there exists constants:
 - $\alpha > 1$
 - n_0

such that for all recursive calls from parent p

with $n > n_0$,

$$\alpha C(p) \leq \sum_{c \in D(p)} C(c)$$

- total cost is the cost of leaves $\in \mathcal{O}(C(\text{leaves}))$
The “leaves” are all nodes (problems) of size $n \leq n_0$

Growing



Brick Method: Leaf Dominated Examples

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

■ cost node: $\sqrt{n}$

■ cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \sqrt{2}\sqrt{n}$

Brick Method: Leaf Dominated Examples

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

- cost node: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \sqrt{2}\sqrt{n}$
- with $\alpha = \sqrt{2}$ and for all $n > n_0 = 0$, cost children $\geq \alpha \times$ cost parent
 $\Rightarrow$ **leaf dominated** ✓

Brick Method: Leaf Dominated Examples

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

- cost node: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \sqrt{2}\sqrt{n}$
- with $\alpha = \sqrt{2}$ and for all $n > n_0 = 0$, cost children $\geq \alpha \times$ cost parent
 $\Rightarrow$ **leaf dominated** ✓
- number of leaves: $(\text{fanout})^{\text{depth}} = 2^{\log_2 n} = n$
- total cost of leaves: $O(n)$ (constant cost each)
- $W(n) = O(n)$

Brick Method: Leaf Dominated Examples

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

- cost node: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \frac{3}{\sqrt{2}}\sqrt{n}$
- with $\alpha = \frac{3}{\sqrt{2}}$ and for all $n > n_0 = 0$, cost children $\geq \alpha \times$ cost parent
 $\Rightarrow$ **leaf dominated** ✓

Brick Method: Leaf Dominated Examples

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

- cost node: $\sqrt{n}$
- cost children: $\sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{2}} = \frac{3}{\sqrt{2}}\sqrt{n}$
- with $\alpha = \frac{3}{\sqrt{2}}$ and for all $n > n_0 = 0$, cost children $\geq \alpha \times$ cost parent
 $\Rightarrow$ **leaf dominated** ✓
- number of leaves: $3^{\lg n} = n^{\lg 3}$
- total cost of leaves: $O(n^{\lg 3})$ (constant cost each)
- $W(n) = O(n^{\lg 3})$

Brick Method: Leaf Dominated Examples

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

- cost node: $\lg n$
- cost children: $\lg \frac{n}{2} + \lg \frac{n}{2} = 2(\lg n - 1)$

Brick Method: Leaf Dominated Examples

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

- cost node: $\lg n$
- cost children: $\lg \frac{n}{2} + \lg \frac{n}{2} = 2(\lg n - 1)$

	$W(\text{parent})$	$W(\text{children})$
n_0	$\lg n$	$2(\lg n - 1)$
2	1	0
4	2	2
8	3	4
16	4	6

Brick Method: Leaf Dominated Examples

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

- cost node: $\lg n$
- cost children: $\lg \frac{n}{2} + \lg \frac{n}{2} = 2(\lg n - 1)$
- with $\alpha = 4/3$ and for all $n > n_0 = 8$,
cost children $\geq \alpha \times$ cost parent
 $\Rightarrow$ **leaf dominated** ✓

Brick Method: Leaf Dominated Examples

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

- cost node: $\lg n$
- cost children: $\lg \frac{n}{2} + \lg \frac{n}{2} = 2(\lg n - 1)$
- with $\alpha = 4/3$ and for all $n > n_0 = 8$,
cost children $\geq \alpha \times$ cost parent
 $\Rightarrow$ **leaf dominated** ✓
- number of leaves: $2^{\lg n/8} = O(n)$
- $W(n) = O(n)$

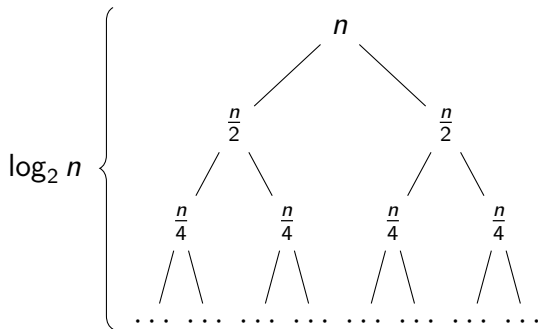
	$W(\text{parent})$	$W(\text{children})$
n_0	$\lg n$	$2(\lg n - 1)$
2	1	0
4	2	2
8	3	4
16	4	6

Brick Method: Balanced

The costs of each level are **approximately the same**

Neither leaf nor roof dominated. Could grow or shrink, but not geometrically.

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

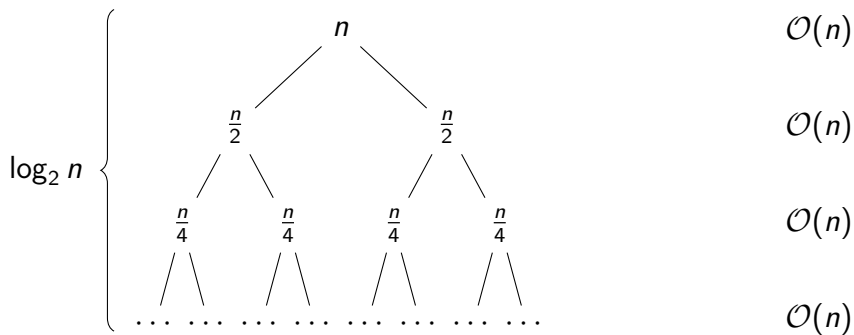


Brick Method: Balanced

The costs of each level are **approximately the same**

Neither leaf nor root dominated. Could grow or shrink, but not geometrically.

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



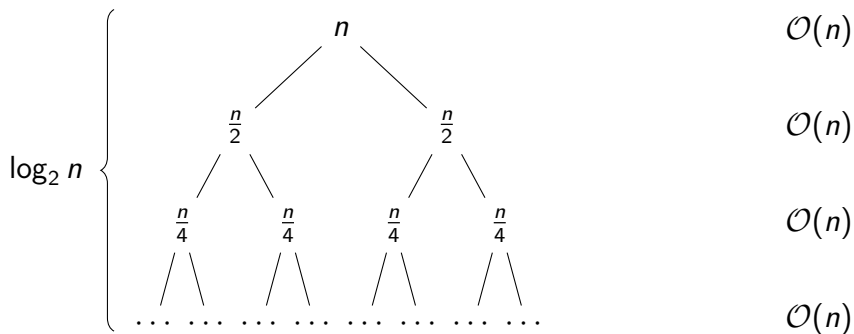
total cost is

Brick Method: Balanced

The costs of each level are **approximately the same**

Neither leaf nor roof dominated. Could grow or shrink, but not geometrically.

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 (Summary)

- **Root Dominated** : shrinks geometrically,

$$W(n) = O(W(\text{root}))$$

Very easy.

- **Leaf Dominated** : grows geometrically,

$$W(n) = O(W(\text{leaves}))$$

A bit tricky.

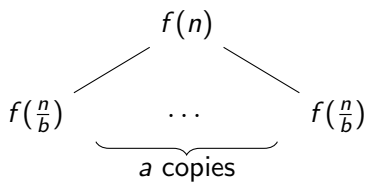
- Need to count the number of leaves.
- Is the cost of each leaf constant? If so $W(n) = O(W(\text{number of leaves}))$

- **Balanced** : neither of above, then cost can always be bounded by

$$\# \text{ of levels} \times \max_{l \in \text{levels}} W(l)$$

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

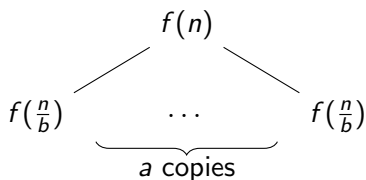
$<$ leaf dominated

$=$ balanced

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

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

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

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

Asymptotic Analysis

Recurrences

Tree Method

Brick Method

Substitution Method

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

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

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: $n^b = (\frac{n}{2})^b + (\frac{n}{3})^b$
- after simplification (dividing by n^b): $1 = (\frac{1}{2})^b + (\frac{1}{3})^b$

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: $n^b = (\frac{n}{2})^b + (\frac{n}{3})^b$
- after simplification (dividing by n^b): $1 = (\frac{1}{2})^b + (\frac{1}{3})^b$
- 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