# CS 341
# Background Information

Collin Roberts

January 15, 2025

## Contents

# 1 Exponent Laws

1. $\log_b a = \frac{\ln a}{\ln b}$

   *Proof.* Let $x = \log_b a \Leftrightarrow b^x = a$. Then we have

$$\begin{aligned} b^x &= a \\ \ln(b^x) &= \ln a \\ x \ln b &= \ln a \\ x &= \frac{\ln a}{\ln b} \\ \log_b a &= \frac{\ln a}{\ln b}. \end{aligned}$$

□

# 2 Geometric Series

Reproduced from CS 240 Module 01, Slide 42:

$$\sum_{i=0}^{n-1} a\,r^i = \begin{cases} a\dfrac{r^n - 1}{r - 1} & \in \Theta(r^{n-1}) \quad \text{if } r > 1 \\ na & \in \Theta(n) \quad \text{if } r = 1 \\ a\dfrac{1 - r^n}{1 - r} & \in \Theta(1) \quad \text{if } 0 < r < 1. \end{cases}$$

# 3 Analysis

**Definition 3.1.** $f(n) \in O(g(n))$ *if there exist constants* $c > 0$ *and* $n_0 > 0$ *such that* $0 \leq f(n) \leq c \cdot g(n)$ *for all* $n \geq n_0$.

**Definition 3.2.** $f(n) \in \Omega(g(n))$ *if there exist constants* $c > 0$ *and* $n_0 > 0$ *such that* $0 \leq c \cdot g(n) \leq f(n)$ *for all* $n \geq n_0$.

**Remarks:**

1. $f(n) \in \Omega(g(n))$ if and only if $g(n) \in O(f(n))$ (just take the reciprocal of the constant, and the same $n_0$).

**Definition 3.3.** $f(n) \in \Theta(g(n))$ *if there exist constants* $c_1, c_2 > 0$ *and* $n_0 > 0$ *such that* $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ *for all* $n \geq n_0$.

**Remarks:**

1. $f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$ .

**Useful Facts:**

1. $\log_b(n) \in \Theta(\log n)$ for all $b > 1$. (Our convention will be that $\log n$ will mean $\log_2 n$.) Proved in CS 240 Lecture Notes and, more elegantly, in the Beidl book.

**Definition 3.4.** *$f(n) \in o(g(n))$ if for all constants $c > 0$, there exists $n_0 > 0$ such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.*

**Definition 3.5.** *$f(n) \in \omega(g(n))$ if $g(n) \in o(f(n))$.*

**Relationships between Order Notations**

1. $f(n) \in \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$
2. $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$
3. $f(n) \in o(g(n)) \Leftrightarrow g(n) \in \omega(f(n))$
4. $f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$
5. $f(n) \in o(g(n)) \Rightarrow f(n) \notin \Omega(g(n))$
6. $f(n) \in \omega(g(n)) \Rightarrow f(n) \in \Omega(g(n))$
7. $f(n) \in \omega(g(n)) \Rightarrow f(n) \notin O(g(n))$

**Algebra of Order Notations**

1. Identity Rule: $f(n) \in \Theta(f(n))$
2. Maximum Rules: Suppose that $f(n) > 0$ and $g(0) > 0$ for all $n \geq n_0$. Then
   (a) $O(f(n) + g(n)) = O(\max\{f(n), g(n)\})$.
   (b) $\Omega(f(n) + g(n)) = \Omega(\max\{f(n), g(n)\})$.
3. Transitivity:
   (a) If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$.
   (b) If $f(n) \in \Omega(g(n))$ and $g(n) \in \Omega(h(n))$, then $f(n) \in \Omega(h(n))$.

**Techniques of Order Notations**

1. Limit Rule: Suppose that $f(n) > 0$ and $g(n) > 0$ for all $n > n_0$. Suppose that

$$L = \lim_{n \to \infty} \frac{f(n)}{g(n)}, \text{ in particular, the limit exists.}$$

   Then

$$f(n) \in \begin{cases} o(g(n)) & \text{if} \quad L = 0 \\ \Theta(g(n)) & \text{if} \quad 0 < L < \infty \\ \omega(g(n)) & \text{if} \quad L = \infty. \end{cases}$$

   **Note:** sufficient, not necessary.

**Growth Rates**

1. If $f(n) \in \Theta(g(n))$, then the growth rates of $f(n)$ and $g(n)$ are the same.
2. If $f(n) \in o(g(n))$, then the growth rate of $f(n)$ is less than the growth rate of $g(n)$.
3. If $f(n) \in \omega(g(n))$, then the growth rate of $f(n)$ is greater than the growth rate of $g(n)$.

**Useful Facts:**

1. The growth rate of $\log n$ is less than the growth rate of $n$. Proved in CS 240 Lecture Notes.
2. The growth rate of $(\log n)^c$ is less than the growth rate of $n^d$, where $c > 0$ and $d > 0$ are arbitrary real numbers. Proved in CS 240 Lecture Notes.

**Complexity of Algorithms**

1. <u>Worst-case complexity of an algorithm</u> Add, if needed.
2. <u>Average-case complexity of an algorithm</u> Add, if needed.

**Definition 3.6.** *$f(n,m) \in O(g(n,m))$ if there exist constants $c > 0$ and $n_0 > 0, m_0 > 0$ such that $0 \leq f(n,m) \leq c \cdot g(n.m)$ for all $n \geq n_0$ **or** $m \geq m_0$ (i.e. finitely many exceptions).*

**Remarks:**

1. <u>Weaker Definition:</u> there exist constants $c > 0$ and $n_0 > 0, m_0 > 0$ such that $0 \leq f(n,m) \leq c \cdot g(n.m)$ for all $n \geq n_0$ **and** $m \geq m_0$.
2. It will not matter much which definition we use.

**Recursive Relations (See CS 240, Module 01)**

| Recursion | resolves to | example |
|---|---|---|
| $T(n) = T(n/2) + \Theta(1)$ | $T(n) \in \Theta(\log n)$ | Binary search |
| $T(n) = 2T(n/2) + \Theta(n)$ | $T(n) \in \Theta(n \log n)$ | Mergesort |
| $T(n) = 2T(n/2) + \Theta(\log n)$ | $T(n) \in \Theta(n)$ | Heapify ($\rightarrow$ later) |
| $T(n) = T(cn) + \Theta(n)$ for some $0 < c < 1$ | $T(n) \in \Theta(n)$ | Selection ($\rightarrow$ later) |
| $T(n) = 2T(n/4) + \Theta(1)$ | $T(n) \in \Theta(\sqrt{n})$ | Range Search ($\rightarrow$ later) |
| $T(n) = T(\sqrt{n}) + \Theta(1)$ | $T(n) \in \Theta(\log \log n)$ | Interpolation Search ($\rightarrow$ later) |

**MergeSort Reference:** See the Beidl book, CS 240E detailed analysis of MergeSort.

**Lemma 3.7.** *For any constant $r > 1$, $f(n) \in \Theta(r^{n-1})$ if and only if $f(n) \in \Theta(r^n)$.*

*Proof.*

1. Forward direction Assume $f(n) \in \Theta(r^{n-1})$.
   (a) Since $f(n) \in \Theta(r^{n-1})$, we have
      i. $c_1, n_1$ such that $f(n) \leq c_1 r^{n-1}$, for all $n \geq n_1$, and
      ii. $c_2, n_2$ such that $c_2 r^{n-1} \leq f(n)$, for all $n \geq n_2$.
   (b) From 1(a)i, $f(n) \leq \left(\frac{c_1}{r}\right) r^n$, for all $n \geq n_1$.
   (c) From 1(a)ii, $\left(\frac{c_2}{r}\right) r^n \leq f(n)$, for all $n \geq n_1$.
   (d) Hence $f(n) \in \Theta(r^n)$ as claimed.
2. Backward direction This is clear enough from the forward direction, I think.

□

# 4 Heaps

1. Refer to CS 240, Module 2, Section on Binary Heaps.
2. Quick Summary: A **heap** is a binary tree with the following properties:
   (a) **Structural Property:** All the levels of a heap are completely filled, except (possibly) for the last level. The filled items in the last level are left-justified.
   (b) **Heap Order Property:** For any node $i$, the key of the parent of $i$ is larger than or equal to key of $i$.

   The full name for this is **max-oriented binary heap**.

**Task:** Make this into a proper definition, when time permits.

# 5 Randomized Algorithms

1. Refer to CS 240, Module 3, Section on Randomized Algorithms.

# 6 Dictionary Using Ordered Linked List

1. Refer to CS 240, Module 4, Section on ADT Dictionary.
2. Quick Summary: Ordering the array improves search from $\Theta(n)$ to $\Theta(\log n)$, compared against the unordered option.

**Task:** Make this into a proper definition, when time permits.

## 7 AVL Trees

1. Refer to CS 240, Module 4, Section on AVL Trees.
2. Quick Summary: An **AVL** is a BST, with the additional balance property that the heights of the left and right subtrees can differ by at most 1.

**Task:** Make this into a proper definition, when time permits.

## 8 Tries

1. Refer to CS 240, Module 6, Section on Tries.
2. Quick Summary: A **Trie** is a radix tree (label each edge with the appropriate character).

**Task:** Make this into a proper definition, when time permits.

## 9 KD Trees

1. Refer to CS 240, Module 8, Section on KD Trees.
2. Quick Summary: A **KD Tree** is a binary tree, which has roughly half of its points in each subtree, at each level.

**Task:** Make this into a proper definition, when time permits.

## 10 Huffman Trees

1. Refer to CS 240, Module 10, Section on Huffman Trees.
2. Quick Summary: A **Huffman Tree** is a tree, to store an encoding, which will produce the minimum length of coded words, I think.

**Task:** Make this into a proper definition, when time permits.

## 11 Graphs

**Notation:** A **DAG** is a **directed acyclic graph**.
A **Hamiltonian Path** is a path that visits each vertex exactly once.
A **Hamiltonian Cycle** is a cycle that is also a Hamiltonian path.

## 12 Fibonacci Numbers

When time permits, copy/move the stuff from Lecture 07 about the Fibonacci numbers and the **Golden Ratio**.