CSE 431/531: Algorithm Analysis and Design (Fall 2024)

Introduction III: Asymptotic Notation

Lecturer: Kelin Luo

Department of Computer Science and Engineering
University at Buffalo

Outline

1. Introduction
   - What is an Algorithm?
   - More Computation Problems
   - Asymptotic Analysis

2. Asymptotic Notations

Outline

Outline

Lecture Review

- Computational Problem: GCD, Sorting Problem
- Algorithms: Euclidean Alg, Insertion Sort Alg
- See correctness proof on Piazza-Resources-Lecture Notes
- Feel free to bring up more real-life problems on Piazza, and discuss potential algorithms.

Analyzing Running Time of Insertion Sort

- Q1: what is the size of input?

Analyzing Running Time of Insertion Sort

- Q1: what is the size of input?
- A1: Running time as the function of size

Analyzing Running Time of Insertion Sort

- Q1: what is the size of input?
- A1: Running time as the function of size
- possible definition of size :
  - Sorting problem: # integers,
  - Greatest common divisor: total length of two integers
  - Shortest path in a graph: # edges in graph

Analyzing Running Time of Insertion Sort

- Q1: what is the size of input?
- A1: Running time as the function of size
- possible definition of size :
  - Sorting problem: # integers,
  - Greatest common divisor: total length of two integers
  - Shortest path in a graph: # edges in graph
- Q2: Which input?
  - For the insertion sort algorithm: if input array is already sorted in ascending order, then algorithm runs much faster than when it is sorted in descending order.

Analyzing Running Time of Insertion Sort

- Q1: what is the size of input?
- A1: Running time as the function of size
- possible definition of size :
  - Sorting problem: # integers,
  - Greatest common divisor: total length of two integers
  - Shortest path in a graph: # edges in graph
- Q2: Which input?
  - For the insertion sort algorithm: if input array is already sorted in ascending order, then algorithm runs much faster than when it is sorted in descending order.
- A2: Worst-case analysis:
  - Running time for size n = worst running time over all possible arrays of length n

Analyzing Running Time of Insertion Sort

- Q3: How fast is the computer?
- Q4: Programming language?

Analyzing Running Time of Insertion Sort

- Q3: How fast is the computer?
- Q4: Programming language?
- A: They do not matter!

Analyzing Running Time of Insertion Sort

- Q3: How fast is the computer?
- Q4: Programming language?
- A: They do not matter!

Important idea: asymptotic analysis

- Focus on growth of running-time as a function, not any particular value.

Outline

Asymptotic Analysis: O-notation

Informal way to define O-notation:

- Ignoring lower order terms
- Ignoring leading constant

Asymptotic Analysis: O-notation

Informal way to define O-notation:

- Ignoring lower order terms
- Ignoring leading constant

- $3n^3 + 2n^2 - 18n + 1028 \Rightarrow 3n^3 \Rightarrow n^3$

Asymptotic Analysis: O-notation

Informal way to define O-notation:

- Ignoring lower order terms
- Ignoring leading constant

- $3n^3 + 2n^2 - 18n + 1028 \Rightarrow 3n^3 \Rightarrow n^3$
- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$

Asymptotic Analysis: O-notation

Informal way to define O-notation:

- Ignoring lower order terms
- Ignoring leading constant

- $3n^3 + 2n^2 - 18n + 1028 \Rightarrow 3n^3 \Rightarrow n^3$
- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$

- $n^2/100 - 3n + 10 \Rightarrow n^2/100 \Rightarrow n^2$

Asymptotic Analysis: O-notation

Informal way to define O-notation:

- Ignoring lower order terms
- Ignoring leading constant

- $3n^3 + 2n^2 - 18n + 1028 \Rightarrow 3n^3 \Rightarrow n^3$
- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$

- $n^2/100 - 3n + 10 \Rightarrow n^2/100 \Rightarrow n^2$
- $n^2/100 - 3n + 10 = O(n^2)$

Asymptotic Analysis: O-notation

- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$
- $n^2/100 - 3n + 10 = O(n^2)$

Asymptotic Analysis: O-notation

- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$
- $n^2/100 - 3n + 10 = O(n^2)$

O-notation allows us to ignore

- architecture of computer
- programming language
- how we measure the running time: seconds or # instructions?

Asymptotic Analysis: O-notation

- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$
- $n^2/100 - 3n + 10 = O(n^2)$

O-notation allows us to ignore

- architecture of computer
- programming language
- how we measure the running time: seconds or $\#$ instructions?
- to execute $a \leftarrow b + c$:
  - program 1 requires 10 instructions, or 10^{-8} seconds
  - program 2 requires 2 instructions, or 10^{-9} seconds

Asymptotic Analysis: O-notation

- $3n^3 + 2n^2 - 18n + 1028 = O(n^3)$
- $n^2/100 - 3n + 10 = O(n^2)$

O-notation allows us to ignore

- architecture of computer
- programming language
- how we measure the running time: seconds or # instructions?
- to execute $a \leftarrow b + c$:
  - program 1 requires 10 instructions, or 10^{-8} seconds
  - program 2 requires 2 instructions, or 10^{-9} seconds
  - they only change by a constant in the running time, which will be hidden by the $O(\cdot)$ notation

Asymptotic Analysis: O-notation

- Algorithm 1 runs in time $O(n^2)$
- Algorithm 2 runs in time $O(n)$

Asymptotic Analysis: O-notation

- Algorithm 1 runs in time $O(n^2)$
- Algorithm 2 runs in time $O(n)$
- Does not tell which algorithm is faster for a specific n!

Asymptotic Analysis: O-notation

- Algorithm 1 runs in time $O(n^2)$
- Algorithm 2 runs in time $O(n)$

- Does not tell which algorithm is faster for a specific n!
- Algorithm 2 will eventually beat algorithm 1 as n increases.

Asymptotic Analysis: O-notation

- Algorithm 1 runs in time $O(n^2)$
- Algorithm 2 runs in time $O(n)$

- Does not tell which algorithm is faster for a specific n!
- Algorithm 2 will eventually beat algorithm 1 as n increases.

- For Algorithm 1: if we increase n by a factor of 2, running time increases by a factor of 4

Asymptotic Analysis: O-notation

- Algorithm 1 runs in time $O(n^2)$
- Algorithm 2 runs in time $O(n)$

- Does not tell which algorithm is faster for a specific n!
- Algorithm 2 will eventually beat algorithm 1 as n increases.

- For Algorithm 1: if we increase n by a factor of 2, running time increases by a factor of 4
- For Algorithm 2: if we increase n by a factor of 2, running time increases by a factor of 2

Asymptotic Analysis of Insertion Sort

insertion-sort(A, n)

for $j \leftarrow 2$ to n **do**
    $key \leftarrow A[j]$
    $i \leftarrow j - 1$
    while $i > 0$ and $A[i] > key$ **do**
        $A[i+1] \leftarrow A[i]$
        $i \leftarrow i - 1$
    $A[i+1] \leftarrow key$

Asymptotic Analysis of Insertion Sort

insertion-sort(A, n)

for $j \leftarrow 2$ to n **do**
  $key \leftarrow A[j]$
  $i \leftarrow j - 1$
  while $i > 0$ and $A[i] > key$ **do**
    $A[i+1] \leftarrow A[i]$
    $i \leftarrow i - 1$
  $A[i+1] \leftarrow key$

- Worst-case running time for iteration j of the outer loop?

Asymptotic Analysis of Insertion Sort

insertion-sort(A, n)

```
for j ← 2 to n do
    key ← A[j]
    i ← j − 1
    while i > 0 and A[i] > key do
        A[i + 1] ← A[i]
        i ← i − 1
    A[i + 1] ← key
```

- Worst-case running time for iteration j of the outer loop?
  Answer: $O(j)$

Asymptotic Analysis of Insertion Sort

insertion-sort(A, n)

for $j \leftarrow 2$ to n **do**
  $key \leftarrow A[j]$
  $i \leftarrow j - 1$
  while $i > 0$ and $A[i] > key$ **do**
    $A[i+1] \leftarrow A[i]$
    $i \leftarrow i - 1$
  $A[i+1] \leftarrow key$

- Worst-case running time for iteration j of the outer loop?
  Answer: $O(j)$
- Total running time $= \sum_{j=2}^{n} O(j) = O(\sum_{j=2}^{n} j)$
  $= O(\frac{n(n+1)}{2} - 1) = O(n^2)$

Exercise: Runtime for Insertion Sort

insertion-sort(A, n)

```
for j ← 2 to n do
    key ← A[j]
    i ← j − 1
    while i > 0 and A[i] > key do
        A[i + 1] ← A[i]
        i ← i − 1
    A[i + 1] ← key
```

- The sequence A is given in ascending order.
  Total running time $= \sum_{j=2}^{n} O(1) = O(\sum_{j=2}^{n} 1) = O(n)$
- The sequence A is given in descending order.
  Total running time $= \sum_{j=2}^{n} O(j) = O(\sum_{j=2}^{n} j) = O(n^2)$

Computation Model

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time
- Basic operations such as addition, subtraction and multiplication take $O(1)$ time

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time
- Basic operations such as addition, subtraction and multiplication take $O(1)$ time
- Each integer (word) has $c \log n$ bits, $c \geq 1$ large enough
  - Reason: often we need to read the integer n and handle integers within range $[-n^c, n^c]$, it is convenient to assume this takes $O(1)$ time.

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time
- Basic operations such as addition, subtraction and multiplication take $O(1)$ time
- Each integer (word) has $c \log n$ bits, $c \geq 1$ large enough
  - Reason: often we need to read the integer n and handle integers within range $[-n^c, n^c]$, it is convenient to assume this takes $O(1)$ time.
- What is the precision of real numbers?

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time
- Basic operations such as addition, subtraction and multiplication take $O(1)$ time
- Each integer (word) has $c \log n$ bits, $c \geq 1$ large enough
  - Reason: often we need to read the integer n and handle integers within range $[-n^c, n^c]$, it is convenient to assume this takes $O(1)$ time.
- What is the precision of real numbers?
  Most of the time, we only consider integers.

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time
- Basic operations such as addition, subtraction and multiplication take $O(1)$ time
- Each integer (word) has $c \log n$ bits, $c \geq 1$ large enough
  - Reason: often we need to read the integer n and handle integers within range $[-n^c, n^c]$, it is convenient to assume this takes $O(1)$ time.
- What is the precision of real numbers?
  Most of the time, we only consider integers.
- Can we do better than insertion sort asymptotically?

Computation Model

- Random-Access Machine (RAM) model
  - reading and writing $A[j]$ takes $O(1)$ time
- Basic operations such as addition, subtraction and multiplication take $O(1)$ time
- Each integer (word) has $c \log n$ bits, $c \geq 1$ large enough
  - Reason: often we need to read the integer n and handle integers within range $[-n^c, n^c]$, it is convenient to assume this takes $O(1)$ time.
- What is the precision of real numbers?
  Most of the time, we only consider integers.
- Can we do better than insertion sort asymptotically?
- Yes: merge sort, quicksort and heap sort take $O(n \log n)$ time

Outline

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$ Yes

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$ Yes
- $2^n - n^{20}$

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$ Yes
- $2^n - n^{20}$ Yes

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$ Yes
- $2^n - n^{20}$ Yes
- $100n - n^2/10 + 50$?

Asymptotically Positive Functions

Def. $f : \mathbb{N} \rightarrow \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$ Yes
- $2^n - n^{20}$ Yes
- $100n - n^2/10 + 50$? No

Asymptotically Positive Functions

Def. $f : \mathbb{N} \to \mathbb{R}$ is an asymptotically positive function if:

- $\exists n_0 > 0$ such that $\forall n > n_0$ we have $f(n) > 0$

- In other words, $f(n)$ is positive for large enough n.
- $n^2 - n - 30$ Yes
- $2^n - n^{20}$ Yes
- $100n - n^2/10 + 50$? No
- We only consider asymptotically positive functions.

O-Notation: Asymptotic Upper Bound

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\}.$$

O-Notation: Asymptotic Upper Bound

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \le cg(n), \forall n \ge n_0\big\}.$$

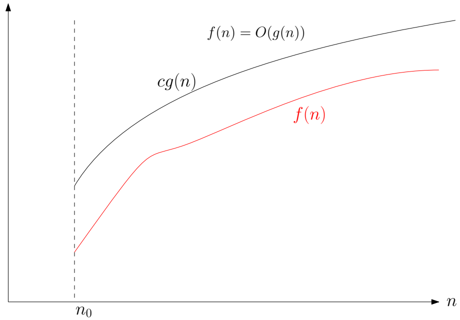
- In other words, $f(n) \in O(g(n))$ if $f(n) \le cg(n)$ for some $c > 0$ and every large enough n.

O-Notation: Asymptotic Upper Bound

O-Notation For a function $g(n)$,

$$O(g(n)) = \{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \leq cg(n)$ for some $c > 0$ and every large enough n.



O-Notation: Asymptotic Upper Bound

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \Big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \le cg(n), \forall n \ge n_0\Big\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \le cg(n)$ for some $c > 0$ and every large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$

O-Notation: Asymptotic Upper Bound

O-Notation For a function $g(n)$,

$$O(g(n)) = \{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \le cg(n), \forall n \ge n_0\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \le cg(n)$ for some $c > 0$ and every large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$

Proof.

Let $c = 4$ and $n_0 = 50$, for every $n > n_0 = 50$, we have,

$$3n^2 + 2n - c(n^2 - 10n) = 3n^2 + 2n - 4(n^2 - 10n)$$
$$= -n^2 + 42n \le 0.$$
$$3n^2 + 2n \le c(n^2 - 10n)$$

$\square$

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \leq cg(n)$ for some c and large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$

O-Notation For a function $g(n)$,

$$O(g(n)) = \Big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that}$$
$$f(n) \leq cg(n), \forall n \geq n_0\Big\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \leq cg(n)$ for some c and large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$
- $3n^2 + 2n \in O(n^3 - 5n^2)$

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \leq cg(n)$ for some c and large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$
- $3n^2 + 2n \in O(n^3 - 5n^2)$
- ? $n^{100} \in O(2^n)$

O-Notation For a function $g(n)$,

$$O(g(n)) = \Big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\Big\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \leq cg(n)$ for some c and large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$
- $3n^2 + 2n \in O(n^3 - 5n^2)$
- ? $n^{100} \in O(2^n)$
- ? $\sin n \in O(1/2)$

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \le cg(n), \forall n \ge n_0\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \le cg(n)$ for some c and large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$
- $3n^2 + 2n \in O(n^3 - 5n^2)$
- ? $n^{100} \in O(2^n)$
- ? $\sin n \in O(1/2)$
- ? $n^3 \notin O(10n^2)$

O-Notation For a function $g(n)$,

$$O(g(n)) = \Big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \le cg(n), \forall n \ge n_0\Big\}.$$

- In other words, $f(n) \in O(g(n))$ if $f(n) \le cg(n)$ for some c and large enough n.
- $3n^2 + 2n \in O(n^2 - 10n)$
- $3n^2 + 2n \in O(n^3 - 5n^2)$
- ? $n^{100} \in O(2^n)$
- ? $\sin n \in O(1/2)$
- ? $n^3 \notin O(10n^2)$

Asymptotic Notations	O	Ω	Θ
Comparison Relations	$\le$		

Conventions

- We use "$f(n) = O(g(n))$" to denote "$f(n) \in O(g(n))$"

Conventions

- We use "$f(n) = O(g(n))$" to denote "$f(n) \in O(g(n))$"
- $3n^2 + 2n = O(n^3 - 10n)$
- $3n^2 + 2n = O(n^2 + 5n)$
- $3n^2 + 2n = O(n^2)$

Conventions

- We use "$f(n) = O(g(n))$" to denote "$f(n) \in O(g(n))$"
- $3n^2 + 2n = O(n^3 - 10n)$
- $3n^2 + 2n = O(n^2 + 5n)$
- $3n^2 + 2n = O(n^2)$

"$=$" is asymmetric! Following equalities are wrong:

- $O(n^3 - 10n) = 3n^2 + 2n$
- $O(n^2 + 5n) = 3n^2 + 2n$
- $O(n^2) = 3n^2 + 2n$

Conventions

- We use "$f(n) = O(g(n))$" to denote "$f(n) \in O(g(n))$"
- $3n^2 + 2n = O(n^3 - 10n)$
- $3n^2 + 2n = O(n^2 + 5n)$
- $3n^2 + 2n = O(n^2)$

"$=$" is asymmetric! Following equalities are wrong:

- $O(n^3 - 10n) = 3n^2 + 2n$
- $O(n^2 + 5n) = 3n^2 + 2n$
- $O(n^2) = 3n^2 + 2n$

- Analogy: Mike is a student. ~~A student is Mike.~~

Ω-Notation: Asymptotic Lower Bound

O-**Notation** For a function $g(n)$,

$$O(g(n)) = \Big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \le cg(n), \forall n \ge n_0\Big\}.$$

Ω-**Notation** For a function $g(n)$,

$$\Omega(g(n)) = \Big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \ge cg(n), \forall n \ge n_0\Big\}.$$

Ω-Notation: Asymptotic Lower Bound

O-Notation For a function $g(n)$,

$$O(g(n)) = \left\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \leq cg(n), \forall n \geq n_0\right\}.$$

Ω-Notation For a function $g(n)$,

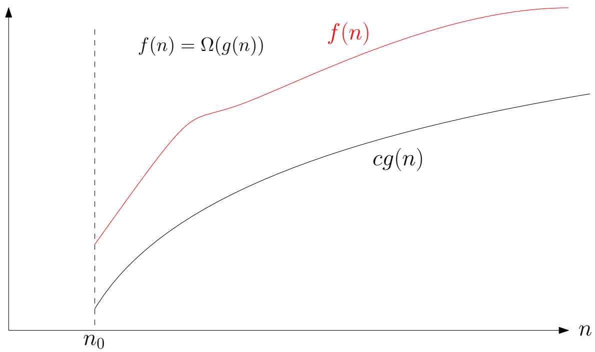
$$\Omega(g(n)) = \left\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \geq cg(n), \forall n \geq n_0\right\}.$$

- In other words, $f(n) \in \Omega(g(n))$ if $f(n) \geq cg(n)$ for some c and large enough n.

Ω-Notation: Asymptotic Lower Bound

Ω-Notation For a function $g(n)$,

$$\Omega(g(n)) = \big\{\text{function } f : \exists c > 0, n_0 > 0 \text{ such that } f(n) \geq cg(n), \forall n \geq n_0\big\}.$$



Ω-Notation: Asymptotic Lower Bound

- Again, we use "$=$" instead of $\in$.
  - $4n^2 = \Omega(n - 10)$
  - $3n^2 - n + 10 = \Omega(n^2 - 20)$

Ω-Notation: Asymptotic Lower Bound

- Again, we use "$=$" instead of $\in$.
  - $4n^2 = \Omega(n - 10)$
  - $3n^2 - n + 10 = \Omega(n^2 - 20)$

Asymptotic Notations	O	Ω	Θ
Comparison Relations	$\leq$	$\geq$	

Ω-Notation: Asymptotic Lower Bound

- Again, we use "$=$" instead of $\in$.
  - $4n^2 = \Omega(n - 10)$
  - $3n^2 - n + 10 = \Omega(n^2 - 20)$

Asymptotic Notations	O	Ω	Θ
Comparison Relations	$\leq$	$\geq$	

Theorem $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$.

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \big\{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\big\}.$$

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that}$$
$$c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}.$$

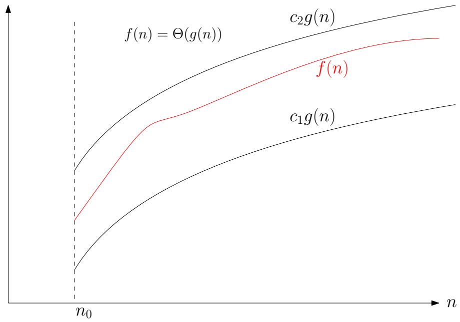
- $f(n) = \Theta(g(n))$, then for large enough n, we have "$f(n) \approx g(n)$".

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}.$$

- $f(n) = \Theta(g(n))$, then for large enough n, we have "$f(n) \approx g(n)$".



Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}.$$

- $3n^2 + 2n = \Theta(n^2 - 20n)$

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \big\{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\big\}.$$

- $3n^2 + 2n = \Theta(n^2 - 20n)$
- $2^{n/3+100} = \Theta(2^{n/3})$

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \big\{\text{function } f : \exists c_2 \ge c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \le f(n) \le c_2 g(n), \forall n \ge n_0\big\}.$$

- $3n^2 + 2n = \Theta(n^2 - 20n)$
- $2^{n/3+100} = \Theta(2^{n/3})$
- ? $2^{n/3+\sqrt{n}+100} = \Theta(2^{n/3})$

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}.$$

- $3n^2 + 2n = \Theta(n^2 - 20n)$
- $2^{n/3+100} = \Theta(2^{n/3})$
- ? $2^{n/3+\sqrt{n}+100} = \Theta(2^{n/3})$

Asymptotic Notations	O	Ω	Θ
Comparison Relations	$\leq$	$\geq$	$=$

Θ-Notation: Asymptotic Tight Bound

Θ-Notation For a function $g(n)$,

$$\Theta(g(n)) = \{\text{function } f : \exists c_2 \geq c_1 > 0, n_0 > 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}.$$

- $3n^2 + 2n = \Theta(n^2 - 20n)$
- $2^{n/3+100} = \Theta(2^{n/3})$
- ? $2^{n/3+\sqrt{n}+100} = \Theta(2^{n/3})$

Asymptotic Notations	O	Ω	Θ
Comparison Relations	$\leq$	$\geq$	$=$

Theorem $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$			
$3n - 50$	$n^2 - 7n$			
$n^2 - 100n$	$5n^2 + 30n$			
$\log_2 n$	$\log_{10} n$			
$\log^{10} n$	$n^{0.1}$			
2^n	$2^{n/2}$			
$\sqrt{n}$	$n^{\sin n}$			

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$			
$n^2 - 100n$	$5n^2 + 30n$			
$\log_2 n$	$\log_{10} n$			
$\log^{10} n$	$n^{0.1}$			
2^n	$2^{n/2}$			
$\sqrt{n}$	$n^{\sin n}$			

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$	Yes	No	No
$n^2 - 100n$	$5n^2 + 30n$			
$\log_2 n$	$\log_{10} n$			
$\log^{10} n$	$n^{0.1}$			
2^n	$2^{n/2}$			
$\sqrt{n}$	$n^{\sin n}$			

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$	Yes	No	No
$n^2 - 100n$	$5n^2 + 30n$	Yes	Yes	Yes
$\log_2 n$	$\log_{10} n$			
$\log^{10} n$	$n^{0.1}$			
2^n	$2^{n/2}$			
$\sqrt{n}$	$n^{\sin n}$			

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$	Yes	No	No
$n^2 - 100n$	$5n^2 + 30n$	Yes	Yes	Yes
$\log_2 n$	$\log_{10} n$	Yes	Yes	Yes
$\log^{10} n$	$n^{0.1}$			
2^n	$2^{n/2}$			
$\sqrt{n}$	$n^{\sin n}$			

We often use $\log n$ for $\log_2 n$. But for $O(\log n)$, the base is not important.

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$	Yes	No	No
$n^2 - 100n$	$5n^2 + 30n$	Yes	Yes	Yes
$\log_2 n$	$\log_{10} n$	Yes	Yes	Yes
$\log^{10} n$	$n^{0.1}$	Yes	No	No
2^n	$2^{n/2}$			
$\sqrt{n}$	$n^{\sin n}$			

We often use $\log n$ for $\log_2 n$. But for $O(\log n)$, the base is not important.

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$	Yes	No	No
$n^2 - 100n$	$5n^2 + 30n$	Yes	Yes	Yes
$\log_2 n$	$\log_{10} n$	Yes	Yes	Yes
$\log^{10} n$	$n^{0.1}$	Yes	No	No
2^n	$2^{n/2}$	No	Yes	No
$\sqrt{n}$	$n^{\sin n}$			

We often use $\log n$ for $\log_2 n$. But for $O(\log n)$, the base is not important.

Exercise

For each pair of functions f, g in the following table, indicate whether f is O, Ω or Θ of g.

f	g	O	Ω	Θ
$n^3 - 100n$	$5n^2 + 3n$	No	Yes	No
$3n - 50$	$n^2 - 7n$	Yes	No	No
$n^2 - 100n$	$5n^2 + 30n$	Yes	Yes	Yes
$\log_2 n$	$\log_{10} n$	Yes	Yes	Yes
$\log^{10} n$	$n^{0.1}$	Yes	No	No
2^n	$2^{n/2}$	No	Yes	No
$\sqrt{n}$	$n^{\sin n}$	No	No	No

We often use $\log n$ for $\log_2 n$. But for $O(\log n)$, the base is not important.