Θ and Ω Notation

COMS10017 - Algorithms 1

Dr Christian Konrad

Limitations/Strengths of Big-O

O-notation: Upper Bound

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$
- For example, if the runtime is $O(n^2)$ then the actual runtime could also be in $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, ...

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$
- For example, if the runtime is $O(n^2)$ then the actual runtime could also be in $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, ...

This is a strong point:

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$
- For example, if the runtime is $O(n^2)$ then the actual runtime could also be in $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, ...

This is a strong point:

- Worst-case runtime: A runtime of $O(f(n))$ guarantees that the algorithm will not be slower, but may be faster

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$
- For example, if the runtime is $O(n^2)$ then the actual runtime could also be in $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, ...

This is a strong point:

- Worst-case runtime: A runtime of $O(f(n))$ guarantees that the algorithm will not be slower, but may be faster
- Example: FAST-PEAK-FINDING often faster than $5 \log n$

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$
- For example, if the runtime is $O(n^2)$ then the actual runtime could also be in $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, ...

This is a strong point:

- Worst-case runtime: A runtime of $O(f(n))$ guarantees that the algorithm will not be slower, but may be faster
- Example: FAST-PEAK-FINDING often faster than $5 \log n$

How to Avoid Ambiguities

- Θ-notation: Growth is precisely determined (up to constants)

Limitations/Strengths of Big-O

O-notation: Upper Bound

- Runtime $O(f(n))$: On any input of length n, the runtime is bounded by some function in $O(f(n))$
- For example, if the runtime is $O(n^2)$ then the actual runtime could also be in $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n\sqrt{n})$, ...

This is a strong point:

- Worst-case runtime: A runtime of $O(f(n))$ guarantees that the algorithm will not be slower, but may be faster
- Example: FAST-PEAK-FINDING often faster than $5 \log n$

How to Avoid Ambiguities

- Θ-notation: Growth is precisely determined (up to constants)
- Ω-notation: Gives us a lower bound (up to constants)

Θ-notation

"Theta"-notation:

Growth is precisely determined up to constants

Definition: Θ-notation ("Theta")

Let $g(n)$ be a function. Then $\Theta(g(n))$ is the set of functions:

$$\Theta(g(n)) = \{f(n) : \text{There exist positive constants } c_1, c_2 \text{ and } n_0 \text{ s.t. } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$$

Θ-notation

"Theta"-notation:

Growth is precisely determined up to constants

Definition: Θ-notation ("Theta")

Let $g(n)$ be a function. Then $\Theta(g(n))$ is the set of functions:

$$\Theta(g(n)) = \{f(n) : \text{There exist positive constants } c_1, c_2 \text{ and } n_0 \text{ s.t. } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$$

$f \in \Theta(g)$: "f is asymptotically sandwiched between constant multiples of g"

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Proof.

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Proof. Suppose that $f \in \Theta(g)$.

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Proof. Suppose that $f \in \Theta(g)$. To show that $g \in \Theta(f)$, we need to prove that there are positive constants C_1, C_2, N_0 such that

$$0 \leq C_1 f(n) \leq g(n) \leq C_2 f(n), \text{for all } n \geq N_0 \ . \quad (1)$$

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Proof. Suppose that $f \in \Theta(g)$. To show that $g \in \Theta(f)$, we need to prove that there are positive constants C_1, C_2, N_0 such that

$$0 \leq C_1 f(n) \leq g(n) \leq C_2 f(n), \text{for all } n \geq N_0 \ . \tag{1}$$

Since $f \in \Theta(g)$, there are positive constants c_1, c_2, n_0 s.t.

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \text{for all } n \geq n_0. \tag{2}$$

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Proof. Suppose that $f \in \Theta(g)$. To show that $g \in \Theta(f)$, we need to prove that there are positive constants C_1, C_2, N_0 such that

$$0 \le C_1 f(n) \le g(n) \le C_2 f(n), \text{for all } n \ge N_0 \ . \tag{1}$$

Since $f \in \Theta(g)$, there are positive constants c_1, c_2, n_0 s.t.

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n), \text{for all } n \ge n_0. \tag{2}$$

Setting $C_1 = \frac{1}{c_2}$, $C_2 = \frac{1}{c_1}$, $N_0 = n_0$, then (1) is equivalent to (2).

Symmetry of Θ

Lemma

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $g \in \Theta(f)$

Proof. Suppose that $f \in \Theta(g)$. To show that $g \in \Theta(f)$, we need to prove that there are positive constants C_1, C_2, N_0 such that

$$0 \leq C_1 f(n) \leq g(n) \leq C_2 f(n), \text{for all } n \geq N_0 \ . \tag{1}$$

Since $f \in \Theta(g)$, there are positive constants c_1, c_2, n_0 s.t.

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \text{for all } n \geq n_0. \tag{2}$$

Setting $C_1 = \frac{1}{c_2}$, $C_2 = \frac{1}{c_1}$, $N_0 = n_0$, then (1) is equivalent to (2). □

Further Properties of Θ

More on Theta

Lemma (Relationship between Θ and Big-O)

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $f \in O(g)$ *and* $g \in O(f)$

Further Properties of Θ

More on Theta

Lemma (Relationship between Θ and Big-O)

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $f \in O(g)$ *and* $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Further Properties of Θ

More on Theta

Lemma (Relationship between Θ and Big-O)

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $f \in O(g)$ *and* $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Runtime of Algorithm in $\Theta(f(n))$?

Further Properties of Θ

More on Theta

Lemma (Relationship between Θ and Big-O)

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $f \in O(g)$ *and* $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Runtime of Algorithm in $\Theta(f(n))$?

- Only makes sense if the algorithm *always* requires $\Theta(f(n))$ steps, i.e., both the *best-case* and *worst-case* runtime are $\Theta(f(n))$

Further Properties of Θ

More on Theta

Lemma (Relationship between Θ and Big-O)

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $f \in O(g)$ *and* $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Runtime of Algorithm in $\Theta(f(n))$?

- Only makes sense if the algorithm *always* requires $\Theta(f(n))$ steps, i.e., both the *best-case* and *worst-case* runtime are $\Theta(f(n))$
- This is not the case in FAST-PEAK-FINDING

Further Properties of Θ

More on Theta

Lemma (Relationship between Θ and Big-O)

The following statements are equivalent:

1. $f \in \Theta(g)$
2. $f \in O(g)$ *and* $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Runtime of Algorithm in $\Theta(f(n))$?

- Only makes sense if the algorithm *always* requires $\Theta(f(n))$ steps, i.e., both the *best-case* and *worst-case* runtime are $\Theta(f(n))$
- This is not the case in FAST-PEAK-FINDING
- However, correct to say that the worst-case runtime of an algorithms is $\Theta(f(n))$

Ω-notation

Big Omega-Notation:

Definition: Ω-notation ("Big Omega")

Let $g(n)$ be a function. Then $\Omega(g(n))$ is the set of functions:

$$\Omega(g(n)) = \{f(n) : \text{There exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}$$

Ω-notation

Big Omega-Notation:

Definition: Ω-notation ("Big Omega")

Let $g(n)$ be a function. Then $\Omega(g(n))$ is the set of functions:

$$\Omega(g(n)) = \{f(n) : \text{There exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}$$

$f \in \Omega(g)$: *"f grows asymptotically at least as fast as g up to constants"*

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof.

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Examples: Big Omega

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Examples: Big Omega

- $10n^2 \in \Omega(n)$

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Examples: Big Omega

- $10n^2 \in \Omega(n)$
- $6^n \in \Omega(n^8)$

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Examples: Big Omega

- $10n^2 \in \Omega(n)$
- $6^n \in \Omega(n^8)$
- ... (reverse examples for $f \in O(g)$)

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Examples: Big Omega

- $10n^2 \in \Omega(n)$
- $6^n \in \Omega(n^8)$
- $\ldots$ (reverse examples for $f \in O(g)$)

Runtime of Algorithm in $\Omega(f)$?

Properties of Ω

Lemma

The following statements are equivalent:

1. $f \in \Omega(g)$
2. $g \in O(f)$

Proof. $\rightarrow$ Exercise.

Examples: Big Omega

- $10n^2 \in \Omega(n)$
- $6^n \in \Omega(n^8)$
- $\dots$ (reverse examples for $f \in O(g)$)

Runtime of Algorithm in $\Omega(f)$?

Only makes sense if best-case runtime is in $\Omega(f)$

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$
- $10n^2 + 1/n = 10n^2 + O(1)$

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$
- $10n^2 + 1/n = 10n^2 + O(1)$

Observe

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$
- $10n^2 + 1/n = 10n^2 + O(1)$

Observe

- Sloppy but very convenient

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$
- $10n^2 + 1/n = 10n^2 + O(1)$

Observe

- Sloppy but very convenient
- When using O, Θ, Ω in equations then details get lost

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$
- $10n^2 + 1/n = 10n^2 + O(1)$

Observe

- Sloppy but very convenient
- When using O, Θ, Ω in equations then details get lost
- This allows us to focus on the essential part of an equation

Using O, Ω, Θ in Equations

Notation

- O, Ω, Θ are often used in equations
- $\in$ is then replaced by $=$

Examples

- $4n^3 = O(n^3)$
- $n + 10 = n + O(1)$
- $10n^2 + 1/n = 10n^2 + O(1)$

Observe

- Sloppy but very convenient
- When using O, Θ, Ω in equations then details get lost
- This allows us to focus on the essential part of an equation
- Not reversible! E.g., $n + 10 = n + O(1)$ but $n + O(1) \neq n + 10$...