Week 1 Review: Induction

CS161 Winter 2026

Zoe Wefers

01-09-2026

Week 1 Review: Induction

CS161 Winter 2026

Zoe Wefers

01-09-2026

What is Induction?

Induction is a powerful and elegant proof technique for specific type of mathematical statements (ones true for all positive integers)

Induction naturally builds on our intuition to “try out examples” when presented with a new problem

How to construct an inductive proof?

To prove a statement by induction do the following …

1) State your problem in the form: **P_n for all positive integers n**

2) Demonstrate that **P_n holds for a small fixed n** (usually n=0 or n=1) (aka “base case”)

3) Assume that P_n is true for an arbitrary positive integer n (aka “inductive hypothesis)

4) Show that **P_n implies P_{n+1}** by invoking the induction hypothesis (“aka inductive step)

Let’s try an example together!

Prove that every power of of 13 can be written as the sum of two squares

Claim: Every power of 13 can be written as the sum of two squares

1) Problem Statement: For all positive integers n, there exists integers x and y such that $13^n = x^2 + y^2$

Claim: Every power of 13 can be written as the sum of two squares

1) Problem Statement: For all positive integers n, there exists integers x and y such that $13^n = x^2 + y^2$

2) Base Case (n=1): $13 = 4 + 9 = 2^2 + 3^2$

Claim: Every power of 13 can be written as the sum of two squares

1) Problem Statement: For all natural numbers n, there exists integers x and y such that $13^n = x^2 + y^2$

2) Base Case (n=1): $13 = 4 + 9 = 2^2 + 3^2$

3) Induction Hypothesis: Let n be a positive integers. Assume that $13^n = x^2 + y^2$ for some integers x and y

Claim: Every power of 13 can be written as the sum of two squares

1) Problem Statement: For all natural numbers n, there exists integers x and y such that $13^n = x^2 + y^2$

2) Base Case (n=1): $13 = 4 + 9 = 2^2 + 3^2$

3) Induction Hypothesis: Let n be a positive integers. Assume that $13^n = x^2 + y^2$ for some integers x and y

4) $13^{n+1} = 13 \times 13^n$

$= (2^2 + 3^2)13^n$

$= (2^2 + 3^2)(a^2 + b^2)$ for some integers a and b by the induction hypotheses

$= 2^2a^2 + 3^2b^2 + 3^2a^2 + 2^2b^2$

$= (2^2a^2 + 12ab + 3^2b^2) + (3^2a^2 - 12ab + 2^2b^2)$

$= (2a+3b)^2 + (3a - 2b)^2$ and now let $x = 2a + 3b$ and $y = 3a - 2b$ which are both integers

$= x^2 + y^2 \rightarrow$ QED!

Why does induction work?

If you prove the base case, the inductive hypothesis and the inductive proof can be iteratively applied to prove that the claim hold for all positive integers

Base Case = $\mathbf{P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow \ldots P_n \rightarrow P_{n+1} \rightarrow \ldots}$

And inductive proof is like an algorithm with a loop that proves the claim for each n

Why does induction work?

Say we proved the base case for P_0

Assume for sake of contradiction that claim P_n does not hold for some positive integer n.

Let n* be the smallest positive integer for such that P_{n^*} does not hold. Then P_{n^*-1} must hold.

But by the inductive step P_{n^*-1} implies P_{n^*}. Thus we have found a contradiction!

When does induction fail?

Claim: Everyone loves algorithms

Proof:

> **Problem Statement**: For all positive integers n, any group of n people, all people in the group love algorithms
>
> **Base Case**: Zoe loves algorithms!
>
> **Inductive Hypothesis**: Assume that any group of n people love algorithms
>
> **Inductive Step**: Say you have a group of n+1 people. Isolate the tallest n people in the group. By the IH they all love algorithms. Isolate the shortest n people. By the IH they all love algorithms. The union of these two subgroup contains all people in the group. Thus all people in the group love algorithms!

When does induction fail?

Claim: Everyone loves algorithms

Proof:

Problem Statement: For any group of n people, all people in the group love algorithms

Base Case: Zoe loves algorithms!

Inductive Hypothesis: Assume that any group of n people love algorithms

Inductive Step: Say you have a group of n+1 people. Isolate the tallest n people in the group. By the IH they all love algorithms. Isolate the shortest n people. By the IH they all love algorithms. Clearly, union of these two subgroup contains all people in the group. Thus all people in the group love algorithms!

Where are the errors in this proof?

When does induction fail?

Claim: Everyone loves algorithms

Proof:

Problem Statement: For any group of n people, all people in the group love algorithms

Base Case: Zoe loves algorithms! (Base case is not general! Should be for any group of 1)

Inductive Hypothesis: Assume that any group of n people love algorithms

Inductive Step: Say you have a group of n+1 people. Isolate the tallest n people in the group. By the IH they all love algorithms. Isolate the shortest n people. By the IH they all love algorithms. Clearly, union of these two subgroup contains all people in the group. Thus all people in the group love algorithms!

Where are the errors in this proof?

Let’s try another example!

When does induction fail?

Claim: Everyone has the same name

Proof:

> **Problem Statement**: For any group of n people, all people in the group have the same name
>
> **Base Case**: A single person has their own name
>
> **Inductive Hypothesis**: Assume that for any group of n people, everyone in the group has the same name
>
> **Inductive Step**: Say you have a group of n+1 people. Isolate the tallest n people in the group. By the IH they all have the same name. Isolate the shortest n people. By the IH they all have the same name. The second tallest person is in both groups. Thus everyone in each subgroup has the same name as the second tallest person. The union of the two subgroups contains the entire group. Thus everyone in the entire group has the same name (the name of the second tallest person).

When does induction fail?

Claim: Everyone has the same name

Proof:

Problem Statement: For any group of n people, all people in the group have the same name

Base Case: A single person has their own name

Inductive Hypothesis: Assume that for any group of n people, everyone in the group has the same name

Inductive Step: Say you have a group of n+1 people. Isolate the tallest n people in the group. By the IH they all have the same name. Isolate the shortest n people. By the IH they all have the same name. The second tallest person is in both groups. Thus everyone in each subgroup has the same name as the second tallest person. The union of the two subgroups contains the entire group. Thus everyone in the entire group has the same name (the name of the second tallest person).

Where are the errors in this proof?

When does induction fail?

Claim: Everyone has the same name

Proof:

> **Problem Statement**: For any group of n people, all people in the group have the same name
>
> **Base Case**: A single person has their own name
>
> **Inductive Hypothesis**: Assume that for any group of n people, everyone in the group has the same name
>
> **Inductive Step**: Say you have a group of n+1 people. Isolate the tallest n people in the group. By the IH they all have the same name. Isolate the shortest n people. By the IH they all have the same name. The second tallest person is in both groups. Thus everyone in each subgroup has the same name as the second tallest person. The union of the two subgroups contains the entire group. Thus everyone in the entire group has the same name (the name of the second tallest person).

IS implicitly assume that there are at least 3 people in the group. Base case is for n=1, but missing n=2. IS does not hold for n+1=2

More examples (increasing difficulty)

1) The sum of the n first odd numbers in n^2
2) The sum of integers from 1 to n is n(n-1)/2
3) The number of nodes in a complete binary tree of height n is 2^n-1
4) $2^{n+2} + 3^{2n+1}$ is divisible by 7 for all positive integers
5) All polygons can be “triangulated” with diagonals
6) Let finitely many lines divides a plane into regions. There is always a way to colors the regions with two colors in such a way that adjacent regions have different colors.

Solutions

4) Page 3 of https://math.dartmouth.edu/archive/m25f19/public_html/Documents/InductionPractice_solns.pdf

5) https://math.stackexchange.com/questions/1877253/triangulation-of-a-simple-polygon-elementary-proof

6) https://www.cut-the-knot.org/Curriculum/Geometry/TwoColorColoring.shtml

Weak Induction vs Strong Induction

Base Case: Show that P_n is true for small fixed n (usually n=0 or n=1)

IH: Assume that P_n is true for some positive integer n

IS: Prove that P_n implies P_{n+1}

Base Case: Show that P_n is true for small fixed n (usually n=0 or n=1)

IH: Show that P_k is true for all $0<k<n$ for some positive integer n

IS: Prove that the IH implies that P_{n+1} is also true

Weak vs Strong Induction

Strong Induction assume the statement is true at all steps from the base case to the n-th step

Weak induction only assumes that the statement is true at the n-th step

Weak and Strong Induction are logically equivalent, but sometimes it is simpler to prove something with strong induction

Let’s try an example of Strong Induction!

Prove that every integer can be written as a product of prime factors.

Claim: Every integer can be written as a product of prime factors

1) Problem Statement: For all natural numbers $n>1$, $n=x_1 * x_2 * \dots * x_k$ where x_i is prime for all i

Claim: Every integer can be written as a product of prime factors

1) Problem Statement: For all natural numbers $n>1$, $n=x_1 * x_2 * \dots * x_k$ where x_i is prime for all i

2) Base Case (n=2): 2 is a product of one prime (itself)

Claim: Every integer can be written as a product of prime factors

1) Problem Statement: For all natural numbers $n>1$, $n=x_1 * x_2 * \dots * x_k$ where x_i is prime for all i

2) Base Case (n=2): 2 is a product of one prime (itself)

3) Induction Hypothesis: Let n be a positive integers. Assume that for all integers $2<k<=n$, k can be written as a product of prime numbers

Claim: Every integer can be written as a product of prime factors

1) Problem Statement: For all natural numbers $n>1$, $n=x_1 * x_2 * \dots * x_k$ where x_i is prime for all i

2) Base Case (n=2): 2 is a product of one prime (itself)

3) Induction Hypothesis: Let n be a positive integers. Assume that for all integers $2<k<=n$, k can be written as a product of prime numbers

4)

Case 1:

If n+1 is prime then

Case 2:

If n+1 is not prime then

Claim: Every integer can be written as a product of prime factors

1) Problem Statement: For all natural numbers $n>1$, $n=x_1 * x_2 * \dots * x_k$ where x_i is prime for all i

2) Base Case (n=2): 2 is a product of one prime (itself)

3) Induction Hypothesis: Let n be a positive integers. Assume that for all integers $2<k<=n$, k can be written as a product of prime numbers

4)

Case 1:

If n+1 is prime then it can be written a product of one prime (itself)

Case 2:

If n+1 is not prime then

Claim: Every integer can be written as a product of prime factors

1) Problem Statement: For all natural numbers n>1, $n=x_1 * x_2 * \dots * x_k$ where x_i is prime for all i

2) Base Case (n=2): 2 is a product of one prime (itself)

3) Induction Hypothesis: Let n be a positive integers. Assume that for all integers 2<k<=n, k can be written as a product of prime numbers

4)

Case 1:

If n+1 is prime then it can be written a product of one prime (itself)

Case 2:

If n+1 is not prime then it can be written as the product of two factors, a and b.

So n+1 = a*b. And we know that 2<a,b<n, so te IH applies to both a and b.

So $a=x_1 * x_2 * \dots * x_k$ and $b=y_1 * y_2 * \dots * y_m$ where x_i and y_j are prime for all i and j.

Then $n+1 = x_1 * x_2 * \dots * x_k * x_1 * x_2 * \dots * x_k$ which is a product of primes. Thus the IH hold for n+1

Big-Oh Notation!

In this class we will use…

- **Big-Oh notation!**
- Gives us a meaningful way to talk about the running time of an algorithm independent of programming language, computing platform, etc., without having to count all the operations.

Main idea:

Focus on how the runtime **scales** with n (the input size).

Some examples…

(Only pay attention to the largest function of n that appears.)

Number of operations	Asymptotic Running Time
$\frac{1}{10}e^n + 10n^2$	$O(e^n)$
$n^3 + 2n^2 + 7$	$O(n^3)$
$0.1\sqrt{n} - 10^9 n^{0.05}$	$O(\sqrt{n})$
$11\log(n) + 1$	$O(\log(n))$

We say this algorithm is "asymptotically faster" than the others.

Example Runtime

$$T(n) = 25n^2 + 5n + 7 \text{ ms}$$

The constant factor of 25 depends on the computing platform..

As n gets large, the lower-order terms don't really matter

$$= O(n^2)$$

pronounced “big-oh of …” or sometimes “oh of …”

Informal definition for O(…)

- Let $T(n)$, $g(n)$ be functions of positive integers.
  - Think of $T(n)$ as a runtime: positive and increasing in n.

- We say “$T(n)$ is $O\big(g(n)\big)$” if:

  for large enough n,

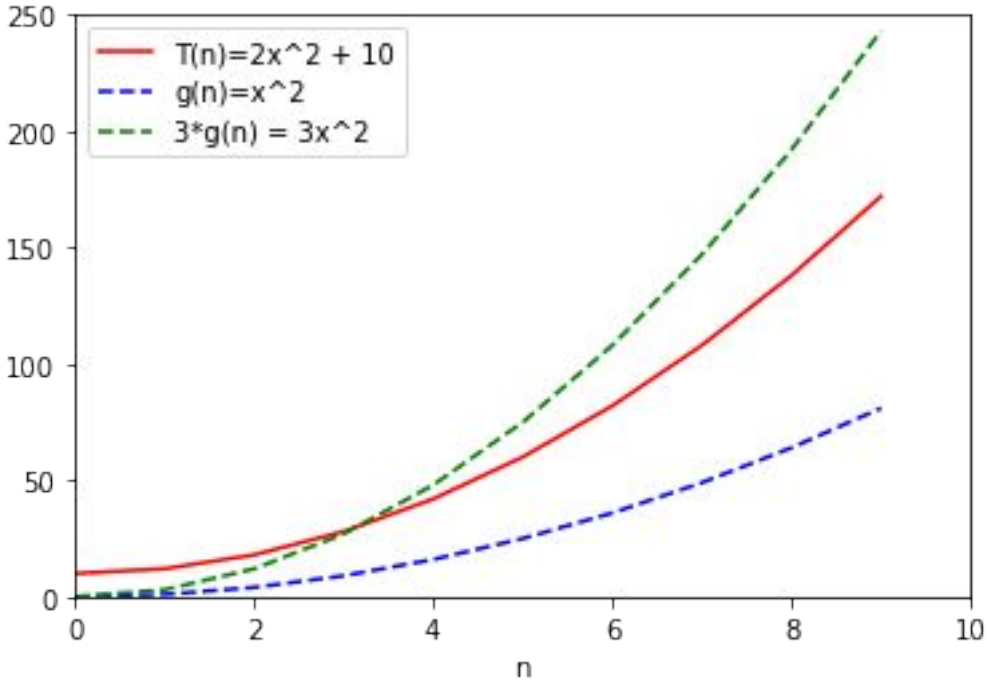
  $T(n)$ is at most some constant multiple of $g(n)$.

Here, “constant” means “some number that doesn’t depend on n.”

Example

$2n^2 + 10 = O(n^2)$

for large enough n,
$T(n)$ is at most some constant multiple of $g(n)$.



$3g(n) = 3n^2$

$T(n) = 2n^2 + 10$

$g(n) = n^2$

Formal definition of O(…)

- Let $T(n)$, $g(n)$ be functions of positive integers.
  - Think of $T(n)$ as a runtime: positive and increasing in n.

- Formally,

$$T(n) = O(g(n))$$

"If and only if" → $\Leftrightarrow$

"For all"

$$\exists c, n_0 > 0 \ \ s.t. \ \ \forall n \geq n_0,$$

$$T(n) \leq c \cdot g(n)$$

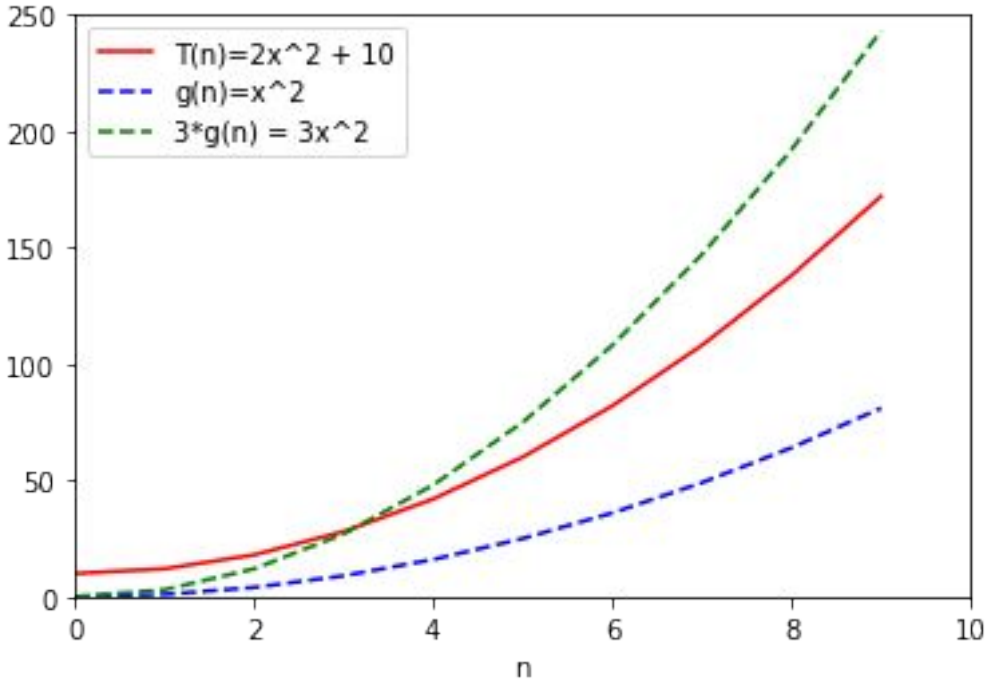
"There exists"

"such that"

Example

$2n^2 + 10 = O(n^2)$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



$3g(n) = 3n^2$

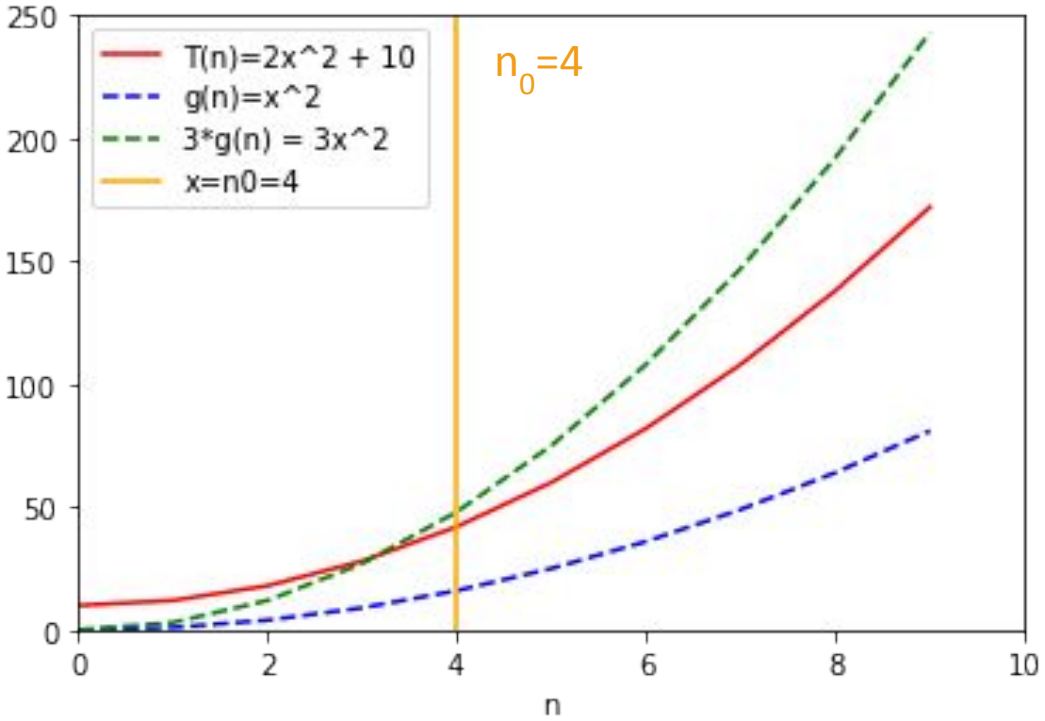
$T(n) = 2n^2 + 10$

$g(n) = n^2$

Example

$2n^2 + 10 = O(n^2)$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



$3g(n) = 3n^2$

$(c = 3)$

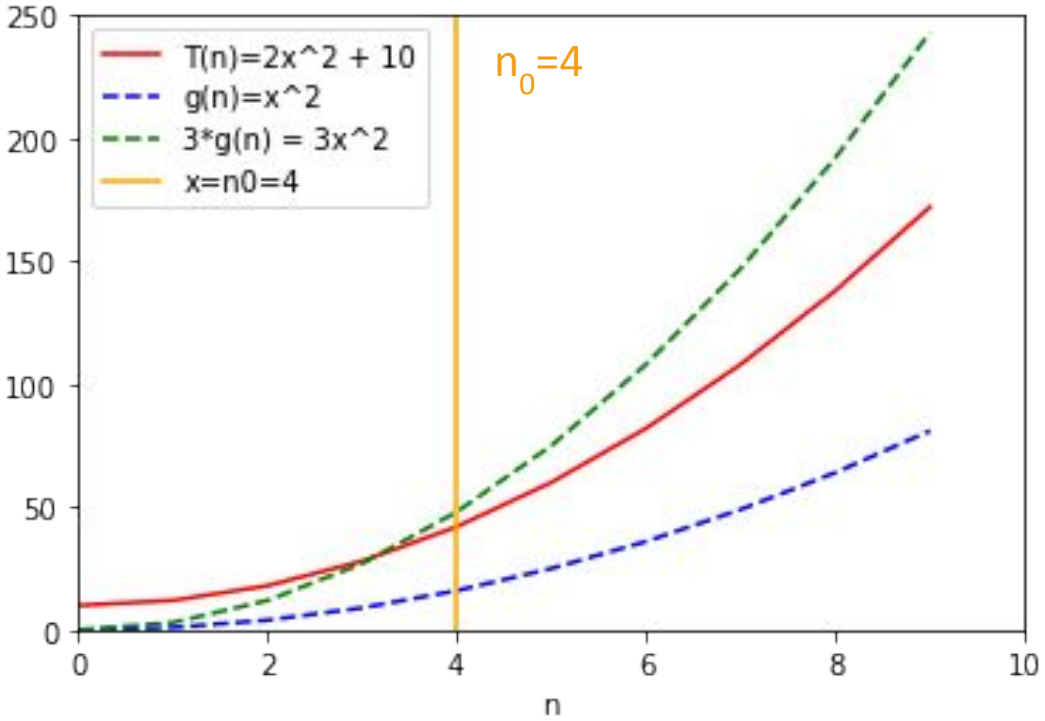
$T(n) = 2n^2 + 10$

$g(n) = n^2$

Example

$2n^2 + 10 = O(n^2)$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$



$3g(n) = 3n^2$

$(c = 3)$

$T(n) = 2n^2 + 10$

$g(n) = n^2$

Formally:

- Choose c = 3
- Choose $n_0 = 4$
- Then:

$$\forall n \geq 4,$$

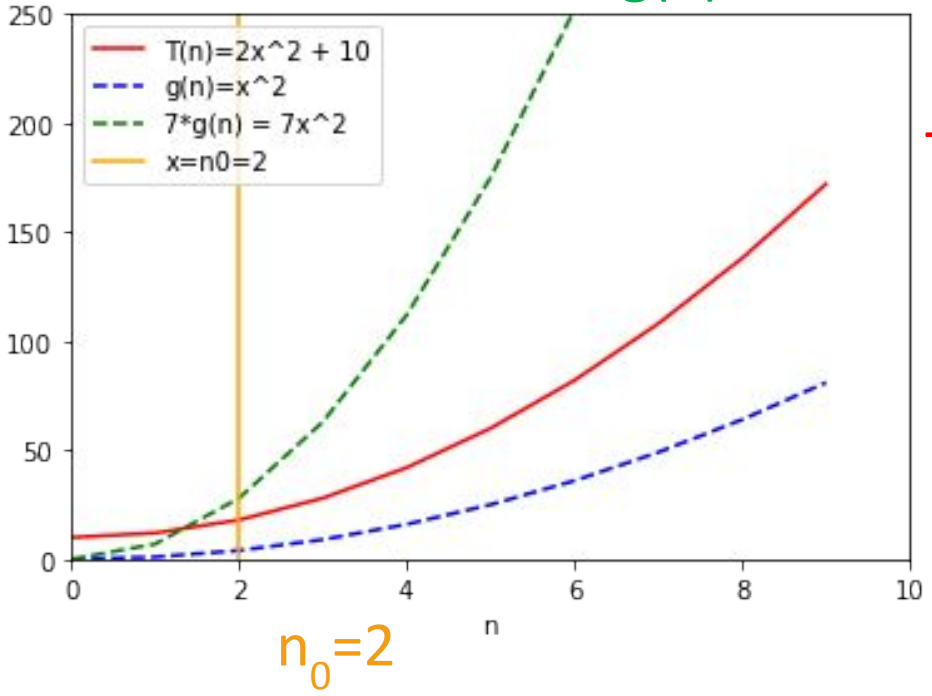
$$2n^2 + 10 \leq 3 \cdot n^2$$

Same example

$2n^2 + 10 = O(n^2)$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$

7g(n) = 7n^2



n_0=2

T(n) = 2n^2 + 10

g(n) = n^2

Formally:

- Choose c = 7
- Choose n_0 = 2
- Then:

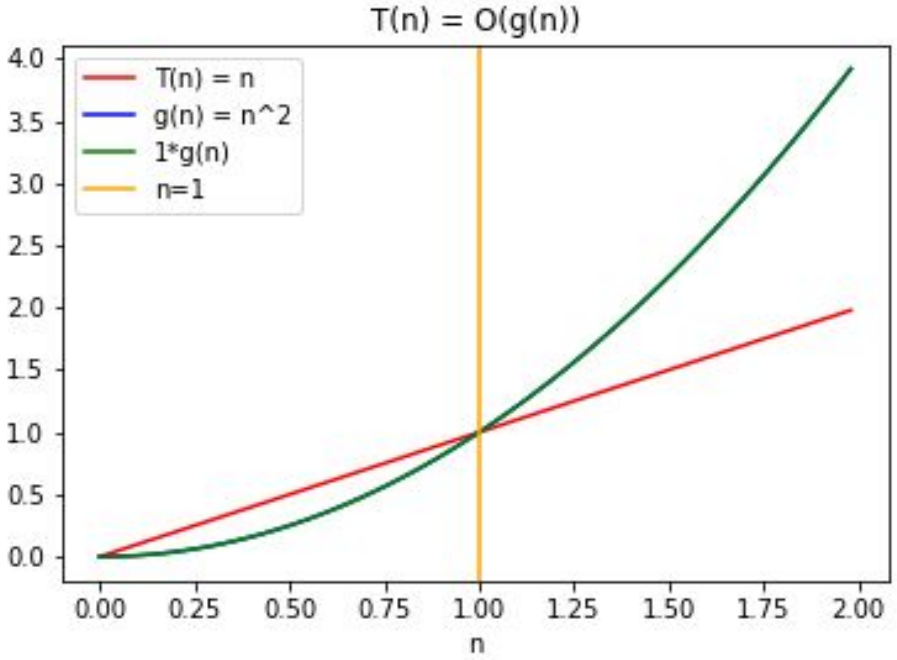
$$\forall n \geq 2,$$
$$2n^2 + 10 \leq 7 \cdot n^2$$

There is no "correct" choice of c and n_0

O(…) is an upper bound:

$$n = O(n^2)$$

$$T(n) = O(g(n)) \iff \exists c, n_0 > 0 \ \ s.t. \ \ \forall n \geq n_0, \ T(n) \leq c \cdot g(n)$$



g(n) = n^2

T(n) = n

- Choose c = 1
- Choose n_0 = 1
- Then

$$\forall n \geq 1,$$

$$n \leq n^2$$

pronounced “big-omega of …”

Ω(…) means a lower bound

- We say “$T(n)$ is $\Omega(g(n))$” if, for large enough n, $T(n)$ is at least as big as a constant multiple of $g(n)$.

- Formally,

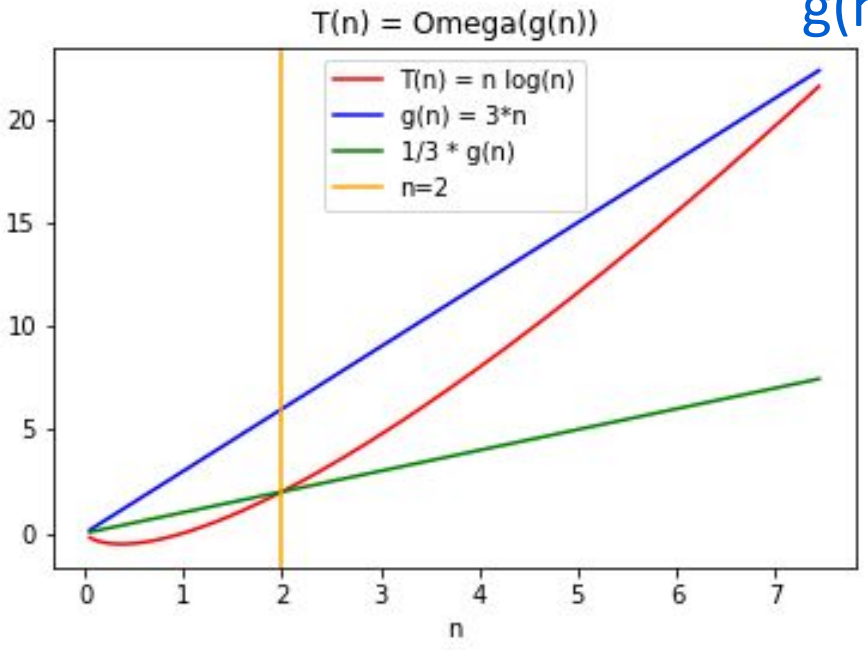
$$T(n) = \Omega(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$c \cdot g(n) \leq T(n)$$

Switched these!!

Example

$$n \log_2(n) = \Omega(3n)$$

$$T(n) = \Omega(g(n)) \iff \exists c, n_0 > 0 \ \ s.t. \ \ \forall n \geq n_0, \ c \cdot g(n) \leq T(n)$$



g(n) = 3n

T(n) = nlog(n)

g(n)/3 = n

- Choose c = 1/3
- Choose $n_0 = 2$
- Then

$$\forall n \geq 2,$$

$$\frac{3n}{3} \leq n \log_2(n)$$

pronounced “big-theta of …” or sometimes “theta of”

Θ(…) means both!

- We say “$T(n)$ is $\Theta(g(n))$” iff both:

$$T(n) = O(g(n))$$

and

$$T(n) = \Omega(g(n))$$

$$T(n) = O(g(n))$$
$$\Leftrightarrow$$
$$\exists c, n_0 > 0 \;\; s.t. \;\; \forall n \geq n_0,$$
$$T(n) \leq c \cdot g(n)$$

Non-Example:
n^2 is not $O(n)$

- Proof by contradiction:
- Suppose that $n^2 = O(n)$.
- Then there is some positive c and n_0 so that:

$$\forall n \geq n_0, \qquad n^2 \leq c \cdot n$$

- Divide both sides by n:

$$\forall n \geq n_0, \qquad n \leq c$$

- That's not true!!! What about, say, $n_0 + c + 1$?
  - Then $n \geq n_0$, but , $n > c$
- Contradiction!

Take-away from examples

- To prove T(n) = O(g(n)), you have to come up with c and n_0 so that the definition is satisfied.
- To prove T(n) is NOT O(g(n)), one way is **proof by contradiction**:
  - Suppose (to get a contradiction) that someone gives you a c and an n_0 so that the definition ***is*** satisfied.
  - Show that this someone must by lying to you by deriving a contradiction.

Practice

- $f(n) = n$ and $g(n) = n^2 - n$

  $f(n) =$ ___ $g(n)$

- $f(n) = 2^n$ and $g(n) = n^2$

  $f(n) =$ ___ $g(n)$

- $f(n) = 8n$ and $g(n) = n \log n$

  $f(n) =$ ___ $g(n)$

Practice

- $f(n) = n$ and $g(n) = n^2 - n$

  $f(n) = O(g(n))$ n grows slower than n^2

- $f(n) = 2^n$ and $g(n) = n^2$

  $f(n) = \Omega(g(n))$ polynomial functions are slower than exponential functions

Practice

- $f(n) = 8n$ and $g(n) = n \log n$

  $f(n) = O(g(n))$

  with $c = 8$, $n_0 = 2$, $8n \leq 8n \log n$

  $1 \leq \log n, \ \forall n \geq 2$

  $f(n) = O(g(n))$

  $\lim_{n \to \infty} 8n / n \log n$

  $= \lim_{n \to \infty} 8 / \log n$

  $= 0$

State order of growth in Θ notation

- $f(n) = 50$
- $f(n) = n + \ldots + 3 + 2 + 1$
- $f(n) = (g(n))^2$ where $g(n) = \sqrt{n} + 5$

State order of growth in Θ notation

- $f(n) = 50$

  $f(n) = \Theta(1)$
- $f(n) = n + \ldots + 3 + 2 + 1$

  $f(n) = n(n+1)/2 = (n^2 + n)/2 = \Theta(n^2)$
- $f(n) = (g(n))^2$ where $g(n) = \sqrt{n} + 5$

  $f(n) = (\sqrt{n} + 5)^2 = n + 10\sqrt{n} + 25 = \Theta(n)$

Summary of Definitions

$f(n) = O(g(n))$ if there exists a $c > 0$ where after large enough n, $f(n) \leq c * g(n)$

Asymptotically f grows as most as much as g

$f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$

Asymptotically, f grows at least as much as g

$f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$

Asymptotically, f and g grow the same

Important Takeaways

- If $d > c$, $n^c = O(n^d)$ but $n^c \neq \Omega(n^d)$
- Asymptotic notation only cares about the highest growing terms: e.g. $n^2 + n = \Omega(n^2)$
- Asymptotic notation does not care about leading constants: e.g. $50n = \Theta(n)$
- Any exponential with base > 1 grows more than any polynomial
- The base of the exponential matters: e.g. $3^n = O(4^n)$ but $3^n \neq \Omega(4^n)$

Any questions?

More Practice! (If time)

Assume that f(n), g(n), and h(n) are all non-negative functions of integers.

1. If $f(n) = O(g(n))$, then $f(n) * h(n) = O(g(n) * h(n))$

$f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$

More Practice! (if time)