

Section 2

Runtime



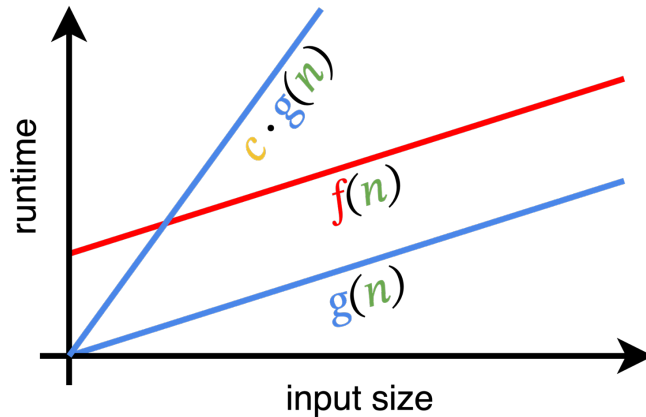
Big-Oh Review

Definition:

If $f: \mathbb{N} \rightarrow \mathbb{R}$ and $g: \mathbb{N} \rightarrow \mathbb{R}$ are two functions, then we say $f(n) \in O(g(n))$ provided there exists positive constants c and n_0 such that $f(n) \leq c \cdot g(n)$ for all values of $n \geq n_0$.

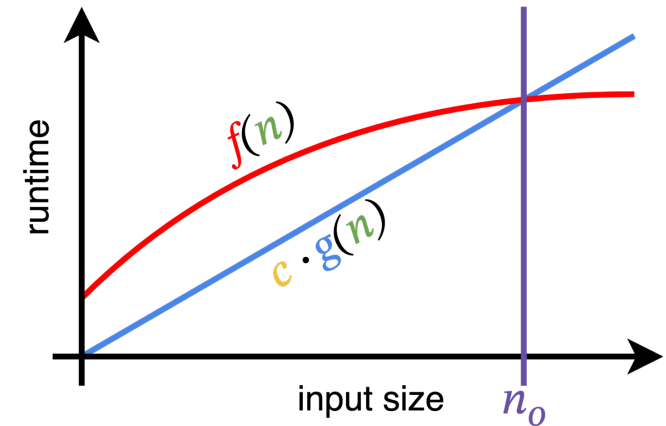
Why is c important?

Ensures that we focus on the *shape* of the runtime curve's *growth*.



Why is n_0 important?

Ensures that we focus on the input size n as it *goes to infinity* by allowing us to ignore small values.



Main ideas:

- In Big-O, we focus on the *growth* of the runtime as the input size n *goes to infinity*.
- Big-O represents an *upper bound* on the algorithm runtime. Not necessarily tight!

Big-Omega and Big-Theta

Definition of Ω :

If $f: \mathbb{N} \rightarrow \mathbb{R}$ and $g: \mathbb{N} \rightarrow \mathbb{R}$ are two functions, then we say $f(n) \in \Omega(g(n))$ provided there exists positive constants c and n_0 such that $f(n) \geq c \cdot g(n)$ for all values of $n \geq n_0$.

- *We also focus on the growth of the runtime as the input size n goes to infinity.*
- *Ω represents a lower bound on the algorithm runtime. Not necessarily tight either.*

Definition of Θ :

If $f: \mathbb{N} \rightarrow \mathbb{R}$ and $g: \mathbb{N} \rightarrow \mathbb{R}$ are two functions, then we say $f(n) \in \theta(g(n))$ provided that both $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

- *Θ represents a tight/exact bound on the algorithm runtime.*

Practice Worksheet Question 0

Suppose that $f(n) \in O(n)$. Indicate if each statement is guaranteed to be always true, guaranteed to be always false, or is sometimes true (and thus also sometimes false).

1. $f(n) \in O(n^2)$
2. $f(n) \in O(4n)$
3. $f(n) \in O(\log n)$
4. $f(n) \in O(n \log n)$
5. $f(n) \in O(1)$
6. $f(n) \in \Omega(n)$
7. $f(n) \in \Omega(\log n)$
8. $f(n) \in \Omega(n^2)$

Solutions Worksheet Question 0

Suppose that $f(n) \in O(n)$. Indicate if each statement is guaranteed to be always true, guaranteed to be always false, or is sometimes true (and thus also sometimes false).

- | | |
|------------------------------|-------------------|
| 1. $f(n) \in O(n^2)$ | 1. Always True |
| 2. $f(n) \in O(4n)$ | 2. Always True |
| 3. $f(n) \in O(\log n)$ | 3. Sometimes True |
| 4. $f(n) \in O(n \log n)$ | 4. Always True |
| 5. $f(n) \in O(1)$ | 5. Sometimes True |
| 6. $f(n) \in \Omega(n)$ | 6. Sometimes True |
| 7. $f(n) \in \Omega(\log n)$ | 7. Sometimes True |
| 8. $f(n) \in \Omega(n^2)$ | 8. Always False |

Worksheet Question 1

1. Prove that $f(n) \in O(g(n))$

a) $f(n) = 7n$, $g(n) = n/10$

b) $f(n) = 1000$, $g(n) = 3n^3$

c) $f(n) = 2^n$, $g(n) = 3^{2n}$

d) $f(n) = 7n^2 + 3n$, $g(n) = n^4$

e) $f(n) = n + 2n\log(n)$, $g(n) = n\log(n)$

How to Approach these Problems

When trying to prove something like $f(n) \in O(g(n))$

or $f(n) \in \Omega(g(n))$ you need to find a c and n_0 .

- The **proof**, or final solution, for the problem should simply declare the values of c and n_0 and should plug them in/explain why they make the inequality true.
- The proof should **not** explain how to *solve for* c and n_0 - that would be your own work.

Show: $10n + 100 \in O(n^2)$.

- **Technique:** find values $c > 0$ and $n_0 > 0$ such that $\forall n \geq n_0$ we have that $10n + 100 \leq c * n^2$.
- **Proof:** Let $c = 10$ and $n_0 = 6$. Show that $\forall n \geq 6, 10n + 100 \leq 10n^2$.

$$\equiv n + 10 \leq n^2$$

$$\equiv 10 \leq n^2 - n$$

$$\equiv 10 \leq n(n - 1)$$

This final expression is true because $n(n - 1)$ is strictly increasing and $6(6 - 1) > 10$

1a) Prove $f(n) = 7n, g(n) = \frac{n}{10}$ (Scratch Work)

We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $7n \leq c \frac{n}{10}$

$$7n \leq c \frac{n}{10}$$

$$70n \leq cn$$

$$70 \leq c$$

This is only ok because we said $n \geq n_0 > 0$

Meaning that this inequality holds for all values $n > 0$ so long as $c \geq 70$, so we can select $c = 70$ and $n_0 = 1$.

1a) Prove $f(n) = 7n, g(n) = \frac{n}{10}$ (Final Solution)

Let $c = 70$ and $n_0 = 1$, let $n \geq n_0$ be an arbitrary integer.
We start with the left-hand-side of the inequality. Observe that

$$7n = 70 \cdot \frac{n}{10} = c \cdot \frac{n}{10}$$

Thus $7n \leq c \cdot \frac{n}{10}$ for all $n \geq n_0$.

We thus have $7n \in O\left(\frac{n}{10}\right)$, by definition.

1b) Prove $f(n) = 1000, g(n) = 3n^2$ (Scratch Work)

We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $1000 \leq c \cdot 3n^2$.

Here we can select $c = 1$ and then make sure n_0 is large enough so $1000 \leq 3n^2$

If we select $n_0 = 20$ then $3n^2 = 1200$

Definition of O holds for $c = 1$ and $n_0 = 20$.

1b) Prove $f(n) = 1000, g(n) = 3n^2$ (Final Solution)

Let $c = 1$ and $n_0 = 20$. Let $n \geq n_0$ be an arbitrary integer. We start with the right-hand-side of the inequality.

Observe that $c3n^2 = 3n^2$ (as $c=1$).

Further, note that the smallest n could be is 20, where we have that $3n^2 = 3 \cdot 20^2 = 1200$.

When $n > 20$, we have $3n^2 > 3 \cdot 20^2 \geq 1200 > 1000$.

Thus, we have $1000 \leq c \cdot 3n^2$, and so $1000 \in O(3n^2)$.

1c) $f(n) = 2^n, g(n) = 3^{2n}$ (Scratch Work)

We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $2^n \leq c \cdot 3^{2n}$.

$$2^n \leq c \cdot 3^{2n}$$
$$2^n \leq c \cdot 9^n$$

If we select $n_0 = 2$ and $c = 1$ then:

$$2^n \leq 1 \cdot 9^n$$

Certainly holds true $\forall n \geq 2$.

1c) Prove $f(n) = 2^n, g(n) = 3^{2n}$ (Final Solution)

Let $c = 1$ and $n_0 = 2$.

We wish to show $\forall n \geq n_0$, we have $2^n \leq c \cdot 3^{2n}$.

Since $c = 1$, our task is to show $2^n \leq 1 \cdot 3^{2n}$.

Note that $3^{2n} = 9^n$. Since $2 < 9$, we certainly have $2^n \leq 9^n$ for all $n \geq n_0$. Thus we have $2^n \leq 9^n = 3^{2n} = c \cdot 3^{2n}$ for all $n \geq n_0$, as required.

Therefore, $2^n \in O(3^{2n})$.

1d) $f(n) = 7n^2 + 3n, g(n) = n^4$ (Scratch Work)

We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $7n^2 + 3n \leq cn^4$

$$\begin{aligned} 7n^2 + 3n &\leq cn^4 \\ 7n^2 + 3n &\leq 7n^2 + 3n^2 \leq cn^4 && \text{As long as } n \geq 1 \\ 10n^2 &\leq c \cdot n^4 \end{aligned}$$

Because $10n^2 \leq c \cdot n^4$ for $c = 10$ and every $n \geq 1$, we can satisfy $f(n) \in O(g(n))$ for $c = 10$ and $n_0 = 1$

1d) Prove $f(n) = 7n^2 + 3n, g(n) = n^4$ (Final Solution)

Let $c = 10$ and $n_0 = 1$, and consider an arbitrary $n \geq n_0$.

We have $7n^2 \leq 7n^4$ (since $n \geq 1$) and $3n \leq 3n^4$ (since $n \geq 1$).

Adding these inequalities we have $7n^2 + 3n \leq 10n^4$, as required.

This approach is “by parts” as we show the runtime for each term separately. Read more on course webpage.

1 e) $f(n) = n + 2n\log_2 n$, $g(n) = n\log_2 n$ (Scratch Work)

We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $n + 2n\log_2 n \leq c \cdot n\log_2 n$:

$$\begin{aligned} n + 2n\log_2 n &\leq c \cdot n\log_2 n \\ n + 2n\log_2 n &\leq 3n\log_2 n \leq c \cdot n\log_2 n \quad \text{As long as } n \geq 2 \end{aligned}$$

Because $3n\log_2 n \leq c \cdot n\log_2 n$ for $c = 3$ and every $n \geq 1$,
and $n + 2n\log_2 n \leq 3n\log_2 n$ for every $n \geq 2$,
we can satisfy $f(n) \in O(g(n))$ for $c = 3$ and $n_0 = 2$

1e) Prove $f(n) = n + 2n\log_2 n$, $g(n) = n\log_2 n$ ⌈ Final Solution ⌋

Let $c = 3$ and $n_0 = 2$. For an arbitrary $n \geq n_0$, we have:

$$\begin{aligned} n &\leq n \log n && (\text{since } n \geq 2, \log n \geq 1) \\ 2n \log n &\leq 2n \log n \end{aligned}$$

Adding these inequalities, we have $n + 2n \log n \leq c \cdot n \log n$, as required. So we have $f(n) \in O(g(n))$.

This approach is the “squeeze method” as we squeeze another function between f and g . More on course webpage.

Worksheet problem 2. Big-Theta Proofs

For each of the following, prove that $f(n) \in \Theta(g)$:

a) $f(n) = 7n$, $g(n) = \frac{n}{10}$

b) $f(n) = n^3 + 10n$, $g(n) = 3n^3$

High Level Approach

When trying to prove a function $f(n) \in \Theta(g)$, you should show that $f(n) \in O(g)$ and $f(n) \in \Omega(g)$.

- The proof, or final solution, for the problem consists of both a big-O and big-Omega proof
- Each sub-proof should simply declare values of c and n_0 and plug them in/explain why they make the inequality true.
- The proof should not explain how to solve for c and n_0 - that would be your own scratch work.

Problem Solving Techniques

Example: Show that $2n^2 + 5 \in \Theta(4n^2)$.

Technique: Demonstrate that $2n^2 + 5 \in O(4n^2)$ and $2n^2 + 5 \in \Omega(4n^2)$

Proof:

First we will show that $2n^2 + 5$ belongs to $O(4n^2)$:

Let $c = 1$ and $n_0 = 3$. We show that $\forall n > n_0$ we have that $2n^2 + 5 \leq c \cdot 4n^2$.

Start with the left-hand side of the inequality. $2n^2 + 5 \leq 3n^2$ because $n^2 > 5$ whenever $n \geq 3$. Since $3n^2 \leq 4n^2$ we can conclude $2n^2 + 5 \in O(4n^2)$.

Next we will show that $2n^2 + 5$ belongs to $\Omega(4n^2)$:

Let $c = \frac{1}{2}$ and $n_0 = 1$. We have that $2n^2 + 5 \geq c \cdot 4n^2$. We start with the left-hand side of the inequality. Observe that $2n^2 + 5 \geq \frac{1}{2} \cdot 4n^2$ for all positive values of n , therefore $2n^2 + 5 \in \Omega(4n^2)$

Since, $2n^2 + 5 \in O(4n^2)$ and $2n^2 + 5 \in \Omega(4n^2)$, it must be that $2n^2 + 5 \in \Theta(4n^2)$.

2a) $f(n) = 7n$, $g(n) = \frac{n}{10}$ (Scratch Work)

- $f(n) \in O(g(n))$ shown in problem 1(a).
- Now we show $f(n) \in \Omega(g(n))$
- We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $7n \geq c \frac{n}{10}$

$$7n \geq c \frac{n}{10}$$

$$70n \geq cn$$

$$70 \geq c$$

- Meaning that this inequality holds for all values of $n > 0$ so long as $c \leq 70$, so we can select $c = 70$ and $n_0 = 1$.

2a) Prove $f(n) = 7n$, $g(n) = \frac{n}{10}$ (Final Solution)

Again, we let $c = 70$ and $n_0 = 1$, let $n \geq n_0$ be an arbitrary integer. We start with the left-hand-side of the inequality. Observe that

$$7n = 70 \cdot \frac{n}{10} = c \cdot \frac{n}{10}$$

Thus $7n \geq c \cdot \frac{n}{10}$ for all $n \geq n_0$.

We thus have $7n \in \Omega\left(\frac{n}{10}\right)$, by definition. Since we have both O and Ω bounds, we conclude $7n \in \Theta\left(\frac{n}{10}\right)$.

2 b) $f(n) = n^3 + 10n, g(n) = 3n^3$. (Scratch)

To show Big-Oh:

- We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $n^3 + 10n \leq c \cdot 3n^3$.
- $n^2 + 10 \leq 2n^2$ when $n \geq 4$.
- Therefore $n^3 + 10n \leq 2n^3$ since n is positive.
- And $2n^3 \leq c \cdot 3n^3$ for $c \geq \frac{2}{3}$.
- By transitivity: $n^3 + 10n \leq c \cdot 3n^3$ for $n \geq 4, c \geq \frac{2}{3}$.
- We can select $c = 1$ and $n_0 = 4$.

2 b) $f(n) = n^3 + 10n, g(n) = 3n^3$. (Scratch)

To show Big-Omega:

- We need to find positive constants c and n_0 so that for all $n \geq n_0$ we have that $n^3 + 10n \geq c \cdot 3n^3$.
- Let's select $c = \frac{1}{3}$.
- $n^3 + 10n \geq n^3$ for all choices of n .
- Definition of Ω holds for $c = \frac{1}{3}$ and $n_0 = 1$.

2b) Prove $f(n) = n^3 + 10n, g(n) = 3n^3$ (Final Pt. 1)

First we will show $n^3 + 10n$ belongs to $O(3n^3)$.

Let $c = 1$ and $n_0 = 4$ and let $n \geq n_0$ be an arbitrary integer.

Observe $10 \leq n^2$ for $n \geq 4$. Multiply both sides by n .

$$10n \leq n^3$$

Add n^3 to both sides.

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

Thus, we have: $n^3 + 10n \leq 2n^3 \leq 1 \cdot 3n^3$. So $n^3 + 10n \in O(3n^3)$

2b) Prove $f(n) = n^3 + 10n, g(n) = 3n^3$ (Final Pt. 2)

Next, we show $n^3 + 10n \in \Omega(3n^3)$.

Consider $c = \frac{1}{3}$ and $n_0 = 1$.

For an arbitrary $n \geq n_0$, we have:

$$n^3 + 10n \geq \frac{1}{3} \cdot 3n^3 + 10n \geq \frac{1}{3} 3n^3 = c3n^3$$

We thus have $n^3 + 10n \in \Omega(3n^3)$.

Since we have matching O and Ω bounds, we conclude
 $n^3 + 10n \in \Theta(3n^3)$

Worksheet problems: Q3

```
1  int numUnique(String[] values) {
2      boolean[] visited = new boolean[values.length]
3      for (int i = 0; i < values.length; i++) {
4          visited[i] = false
5      }
6      int out = 0
7      for (int i = 0; i < values.length; i++) {
8          if (!visited[i]) {
9              out += 1
10             for (int j = i; j < values.length; j++)
11             {
12                 if (values[i].equals(values[j])) {
13                     visited[j] = true
14                 }
15             }
16         }
17     }
18     return out;
19 }
```

Get the $\Theta(\cdot)$ bound of each function below.

- a) $f(n)$ = worst case running time
- b) $g(n)$ = best case running time

We will construct equations for each function and then simplify them to get a closed form.

3a) $f(n)$ = worst case running time

```
1  int numUnique(String[] values) {
2      boolean[] visited = new boolean[values.length]
3      for (int i = 0; i < values.length; i++) {
4          visited[i] = false
5      }
6      int out = 0
7      for (int i = 0; i < values.length; i++) {
8          if (!visited[i]) {
9              out += 1
10             for (int j = i; j < values.length; j++)
11             {
12                 if (values[i].equals(values[j])) {
13                     visited[j] = true
14                 }
15             }
16         }
17     }
18     return out;
19 }
```

First function (lines 3-5) always runs once.

Worst case for lines 7-16 is if every value in array is unique.

This means for every iteration of the outer loop, the inner loop will iterate through the rest of the array.

Total number of iterations is $n + (n - 1) + (n - 2) + \dots + 1$.

There is a formula for this sum: $n(n + 1)/2$
 $f(n) = n + n(n + 1)/2 \Rightarrow n^2$.

This is quadratic running time

Now, we do something similar to Q2 proofs to prove that $n + n(n + 1)/2 \in \Theta(n^2)$.

3b) $g(n)$ = best case running time

```
1  int numUnique(String[] values) {
2      boolean[] visited = new boolean[values.length]
3      for (int i = 0; i < values.length; i++) {
4          visited[i] = false
5      }
6      int out = 0
7      for (int i = 0; i < values.length; i++) {
8          if (!visited[i]) {
9              out += 1
10             for (int j = i; j < values.length; j++)
11             {
12                 if (values[i].equals(values[j])) {
13                     visited[j] = true
14                 }
15             }
16         }
17     }
18     return out;
19 }
```

First function (lines 3-5) always runs once.

Best case for lines 7-16 is if every value in array is the same.

This means the inner loop will only run once: for the first iteration of the outer loop.

Then, it will not run because every index will be visited after the first time.

$$g(n) = n + n \Rightarrow n$$

Heaps!

- Review material on Heaps can be found on the course website under Home Page → Summaries
 - Review of the ADT and its uses
 - Associated methods
 - Building Heaps efficiently
- Please come to office hours and ask questions on the Ed Board
- Additionally, find [this video review for more contents on Heaps](#) from the course Panopto

Thanks!