

COMPSCI 311 Section 1: Introduction to Algorithms

Lecture 3: Big-Omega, Big-Theta, Running-Time Analysis

Dan Sheldon

University of Massachusetts

{February 9, 2025}

Clicker

Suppose f is $O(g)$. Which of the following is true?

- A. g is $O(f)$
- B. g is not $O(f)$
- C. g may be $O(f)$, depending on the particular functions f and g

Limitations of Big-O

- ▶ $10 \log(n)$ is $O(\log n)$, but also $O(n)$, $O(n^2)$, $O(n^3)$, ...
- ▶ $4n^2 + 10n + 100$ is $O(n^2)$, but also $O(n^3)$, $O(n^4)$, $O(n^5)$, ...

Big-Ω Motivation

Algorithm **foo**

```
for i= 1 to n do
  for j= 1 to n do
    do something...
```

Fact: run time is $O(n^3)$

Algorithm **bar**

```
for i= 1 to n do
  for j= 1 to n do
    for k= 1 to n do
      do something else..
```

Fact: run time is $O(n^3)$

Conclusion: **foo** and **bar** have the same asymptotic running time. [What is wrong?](#)

More Big- Ω Motivation

Algorithm **sum-product**

```
sum = 0
```

```
for  $i = 1$  to  $n$  do
```

```
    for  $j = i$  to  $n$  do
```

```
        sum +=  $A[i] * A[j]$ 
```

What is the running time of **sum-product**?

Easy to see it is $O(n^2)$. Could it be better? $O(n)$?

Big- Ω

Informally: T grows at least as fast as f

Definition: The function $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that

$$T(n) \geq cf(n) \text{ for all } n \geq n_0$$

f is an asymptotic lower bound for T

Big- Ω Examples

$$4n + 10 = \Omega(n)$$

$$\frac{1}{2}n^2 = \Omega(n^2)$$

Clicker

Claim $n - 10$ is $\Omega(n)$

To prove this we need to show that

$$n - 10 \geq cn \quad \text{for all } n \geq n_0$$

Clicker. What is the largest value of c below for which we can find some n_0 to make this statement true?

- A. $c = 0.5$
- B. $c = 0.99$
- C. $c = 2$
- D. $c = 20$

Big- Ω

Exercise: let $T(n)$ be the running time of **sum-product**. Show that $T(n)$ is $\Omega(n^2)$

Algorithm **sum-product**

```
sum = 0
```

```
for  $i = 1$  to  $n$  do
```

```
    for  $j = i$  to  $n$  do
```

```
        sum +=  $A[i] * A[j]$ 
```

Solution

Hard way

- ▶ Count exactly how many times the loop executes

$$1 + 2 + \dots + n = \frac{n(n+1)}{2} = \Omega(n^2)$$

Easy way

- ▶ Ignore all loop executions where $i > n/2$ or $j < n/2$
- ▶ The inner statement executes at least $(n/2)^2 = \Omega(n^2)$ times

Big- Θ

Definition: the function $T(n)$ is $\Theta(f(n))$ if it is both $O(f(n))$ and $\Omega(f(n))$.

f is an **asymptotically tight bound** of T

Example. $T(n) = 32n^2 + 17n + 1$

- ▶ $T(n)$ is $\Theta(n^2)$
- ▶ $T(n)$ is neither $\Theta(n)$ nor $\Theta(n^3)$

Big- Θ example

How do we correctly compare the running time of these algorithms?

Algorithm **foo**

```
for  $i = 1$  to  $n$  do
  for  $j = 1$  to  $n$  do
    do something...
```

Algorithm **bar**

```
for  $i = 1$  to  $n$  do
  for  $j = 1$  to  $n$  do
    for  $k = 1$  to  $n$  do
      do something else..
```

Answer: **foo** is $\Theta(n^2)$ and **bar** is $\Theta(n^3)$. They do not have the same asymptotic running time.

Additivity Revisited

Suppose f and g are two (non-negative) functions and f is $O(g)$

Old version: Then $f + g$ is $O(g)$

New version: Then $f + g$ is $\Theta(g)$

$$\underbrace{n^2}_g + \underbrace{42n + n \log n}_f \text{ is } \Theta(n^2)$$

Efficiency

When is an algorithm efficient?

Stable Matching Brute force: $\Omega(n!)$

Propose-and-Reject?: $O(n^2)$

We must have done something clever

Polynomial Time

Definition: an algorithm runs in **polynomial time** if its running time is $O(n^d)$ for some constant d

Polynomial Time: Examples

These are polynomial time:

$$f_1(n) = n$$

$$f_2(n) = 4n + 100$$

$$f_3(n) = n \log(n) + 2n + 20$$

$$f_4(n) = 0.01n^2$$

$$f_5(n) = n^2$$

$$f_6(n) = 20n^2 + 2n + 3$$

Not polynomial time:

$$f_7(n) = 2^n$$

$$f_8(n) = 3^n$$

$$f_9(n) = n!$$

Why Polynomial Time ?

Why is this a good definition of efficiency?

- ▶ Matches practice: almost all practically efficient algorithms have this property.
- ▶ Usually distinguishes a clever algorithm from a “brute force” approach.
- ▶ Refutable: gives us a way of saying an algorithm is not efficient, or that **no efficient algorithm exists**.

Bonus if Time: Clicker Fun

Clicker

Algorithm Print1(n)

for $i=1$ to n **do**

 print "X"

for $j=1$ to n **do**

 print "Y"

What is the output of this algorithm with $n = 4$? (ignore spaces)

- A. XYYY XYYY XYYY
- B. XXXX YYYY YYYY YYYY YYYY
- C. XYYYY XYYYY XYYYY XYYYY
- D. XYYYYY XYYYYY XYYYYY XYYYYY

Clicker

Algorithm Print1(n)

for $i=1$ to n **do**

 print "X"

for $j=1$ to n **do**

 print "Y"

What is the exact number of characters printed as a function of n ?

- A. n
- B. n^2
- C. $n^2 - n$
- D. $n^2 + n$

Clicker

Algorithm Print1(n)

```
for  $i=1$  to  $n$  do  
  print "X"  
  for  $j=1$  to  $n$  do  
    print "Y"
```

The running time is:

- A. $\Omega(\sqrt{n})$
- B. $\Theta(n^2)$
- C. $O(n^4)$
- D. all of the above

Clicker

Algorithm Print2(n)

for $i=1$ to n **do**

 print "X"

if $i == 1$ **then**

for $j=1$ to n **do**

 print "Y"

What is the output of this algorithm with $n = 4$? (ignore spaces)

- A. XXXX YYY YYY YYY YYY
- B. XXXXX XXXXX XXXXX XXXXX
- C. XXXXX X X X
- D. XXXXXX XXXXXX XXXXXX XXXXXX

Clicker

Algorithm Print2(n)

for $i=1$ to n **do**

 print "X"

if $i == 1$ **then**

for $j=1$ to n **do**

 print "Y"

What is the exact number of characters printed as a function of n ?

- A. n
- B. $2n$
- C. $n^2 - n$
- D. n^2

Clicker

Algorithm Print2(n)

for $i=1$ to n **do**

 print "X"

if $i == 1$ **then**

for $j=1$ to n **do**

 print "Y"

What is the tight running-time bound of the algorithm?

- A. $\Theta(\log n)$
- B. $\Theta(n)$
- C. $\Theta(n^2)$
- D. $\Theta(n^3)$