Lecture 6: More Definitions, Modeling Complex Algorithms

CSE 373: Data Structures and Algorithms

Administrivia

Homework 1 due tonight

Homework 2 goes live tonight

Email list active!

- turn off discussion board notifications
- please do use discussion board

IDE Setup Office Hours today CSE Floor 2 breakout 3:30-7:30

Please fill out student background survey

Edge Cases

True or False: $10n^2 + 15n$ is $O(n^3)$

It's true – it fits the definition

$10n^2 \leq c \cdot n^3 \; when \; c = 10 \; for \; n \geq 1$

$15n \leq c \cdot n^3 \; when \; c = 15 \; for \; n \geq 1$

$10n^2 + 15n \leq 10n^3 + 15n^3 \leq 25n^3 \; for \; n \geq 1$

$10n^2 + 15n$ is $O(n^3)$ because $10n^2 + 15n \leq 25n^3 \; for \; n \geq 1$

Big-O is just an upper bound. It doesn't have to be a good upper bound

If we want the best upper bound, we'll ask you for a **tight** big-O bound.
$O(n^2)$ is the tight bound for this example.
It is (almost always) technically correct to say your code runs in time $O(n!)$.
DO NOT TRY TO PULL THIS TRICK ON AN EXAM. Or in an interview.

Why Are We Doing This?

You already intuitively understand what big-O means.

Who needs a formal definition anyway?

- We will.

Your intuitive definition and my intuitive definition might be different.

We're going to be making more subtle big-O statements in this class.

- We need a mathematical definition to be sure we're on the same page.

Once we have a mathematical definition, we can go back to intuitive thinking.

- But when a weird edge case, or subtle statement appears, we can figure out what's correct.

3 Minutes

Function comparison: exercise

$f(n) = n \leq g(n) = 5n + 3$? **True** – all linear functions are treated as equivalent

$f(n) = 5n + 3 \leq g(n) = n$? **True**

$f(n) = 5n + 3 \leq g(n) = 1$? **False**

$f(n) = 5n + 3 \leq g(n) = n^2$? **True** – quadratic will always dominate linear

$f(n) = n^2 + 3n + 2 \leq g(n) = n^3$? **True**

$f(n) = n^3 \leq g(n) = n^2 + 3n + 2$? **False**

O, Omega, Theta [oh my?]

Big-O is an **upper bound**
- My code takes at most this long to run

Big-Omega is a **lower bound**

Big-Omega

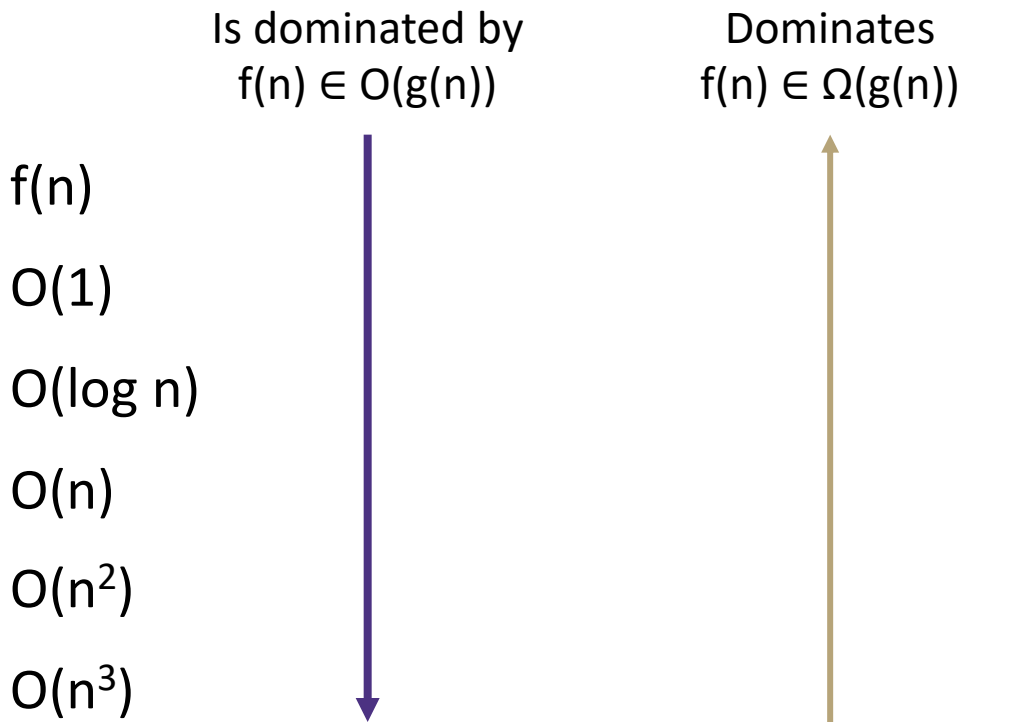
$f(n)$ is $\Omega(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$,
$f(n) \geq c \cdot g(n)$

Big Theta is **"equal to"**

Big-Theta

$f(n)$ is $\Theta(g(n))$ if
$f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

$$\Omega(f(n)) \leq f(n) == \theta(f(n)) \leq O(f(n))$$



Viewing O as a class

Sometimes you'll see big-O defined as a family or set of functions.

Big-O (alternative definition)

$O(g(n))$ is the set of all functions $f(n)$ such that there exist positive constants c, n_0 such that for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$

For that reason, some people write $f(n) \in O(g(n))$ where we wrote "$f(n)$ is $O(g(n))$".
Other people write "$f(n) = O(g(n))$" to mean the same thing.
The set of all functions that run in linear time (i.e. $O(n)$) is a "complexity class."
We never write $O(5n)$ instead of $O(n)$ – they're the same thing!

It's like writing $\frac{6}{2}$ instead of 3. It just looks weird.

Examples

$4n^2 \in \Omega(1)$

true

$4n^2 \in \Omega(n)$

true

$4n^2 \in \Omega(n^2)$

true

$4n^2 \in \Omega(n^3)$

false

$4n^2 \in \Omega(n^4)$

false

$4n^2 \in O(1)$

false

$4n^2 \in O(n)$

false

$4n^2 \in O(n^2)$

true

$4n^2 \in O(n^3)$

true

$4n^2 \in O(n^4)$

true

Big-O

$f(n) \in O(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$

Big-Omega

$f(n) \in \Omega(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$, $f(n) \geq c \cdot g(n)$

Big-Theta

$f(n) \in \Theta(g(n))$ if $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

3 Minutes

Practice

$5n + 3 \in O(n)$ True

$n \in O(5n + 3)$ True

$5n + 3 = O(n)$ True

$O(5n + 3) = O(n)$ True

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

$n^2 \in O(1)$ False

$n^2 \in O(n)$ False

$n^2 \in O(n^2)$ True

$n^2 \in O(n^3)$ True

$n^2 \in O(n^{100})$ True

Big-O

$f(n) \in O(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$,
$f(n) \leq c \cdot g(n)$

Big-Omega

$f(n) \in \Omega(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$,
$f(n) \geq c \cdot g(n)$

Big-Theta

$f(n) \in \Theta(g(n))$ if
$f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

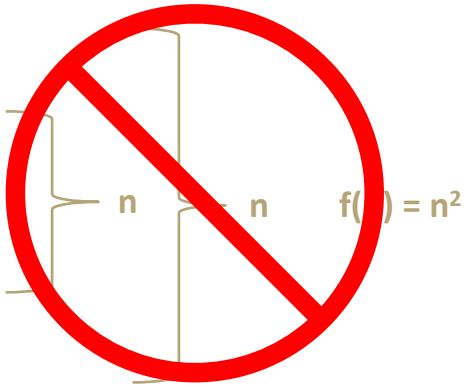
Modeling Complex Code

2 Minutes

Modeling Complex Loops

Write a mathematical model of the following code

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!");
    }
}
```



Keep an eye on loop bounds!

Modeling Complex Loops

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!"); +1
    }
}
```

0 + 1 + 2 + 3 +…+ i-1

n

T(n) = n (0 + 1 + 2 + 3 +…+ i-1)

How do we model this part?

Summations!

1 + 2 + 3 + 4 +… + n = $\sum_{i=1}^{n} i$

Definition: Summation

$\sum_{i=a}^{b} f(i)$ = f(a) + f(a + 1) + f(a + 2) + … + f(b-2) + f(b-1) + f(b)

T(n) = $\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1$

What is the Big O?

Summation Identities

https://courses.cs.washington.edu/courses/cse373/19wi/resources/math/

Provided on the exams!

1. **Splitting a sum**

Rule:

$$\sum_{i=a}^{b}(x+y) = \sum_{i=a}^{b} x + \sum_{i=a}^{b} y$$

Example:

$$\sum_{i=1}^{10}(5+7) = 120 = 50 + 70 = \sum_{i=1}^{10} 5 + \sum_{i=1}^{10} 7$$

2. **Adjusting summation bounds**

Rule:

$$\sum_{i=a}^{b} f(x) = \sum_{i=0}^{b} f(x) - \sum_{i=0}^{a-1} f(x)$$

Example:

$$\sum_{i=5}^{8} i = 5 + 6 + 7 + 8 = \sum_{i=0}^{8} i - \sum_{i=0}^{4} i$$

3. **Factoring out a constant**

Rule:

$$\sum_{i=a}^{b} cf(i) = c\sum_{i=a}^{b} f(i)$$

Note: c is some constant.

Example:

$$\sum_{i=1}^{5} 10n = 10\sum_{i=1}^{5} n$$

And more!

Simplifying Summations

```
for (int i = 0; i < n; i++) {
   for (int j = 0; j < i; j++) {
      System.out.println("Hello!");
   }
}
```

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1$$

$$= \sum_{i=0}^{n-1} 1 \cdot i$$

Summation of a constant

$$\sum_{i=0}^{n-1} c = cn$$

$$= 1 \sum_{i=0}^{n-1} i$$

Factoring out a constant

$$\sum_{i=a}^{b} cf(i) = c \sum_{i=a}^{b} f(i)$$

$$= c \frac{n(n-1)}{2}$$

Gauss's Identity

$$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$

$$= \frac{c}{2} n^2 - \frac{c}{2} n = \mathbf{O(n^2)}$$

1 Minute

Modeling Recursion

Write a mathematical model of the following code

```
public int factorial(int n) {
    if (n == 0 || n == 1) {          +3
        return 1;                    +1
    } else {
        return n * factorial(n-1);
    }                    +1          +??????
}
```

recurrence!

$$T(n) = \begin{cases} 4 \; when \; n = 0,1 \\ T(n-1) \; otherwise \end{cases}$$

Writing a Recurrence

If the function runs recursively, our formula for the running time should probably be recursive as well.

- *Such a formula is called a **recurrence**.*

$$T(n) = \begin{cases} T(n-1) + 2 & \text{if } n > 1 \\ 1 & \text{otherwise} \end{cases}$$

What does this say?

- The input to T is the size of the input to the Length.
- If the input to T() is large, the running time depends on the recusive call.
- If not we can just use the base case.

Another example

```
public int Mystery(int n){
    if(n == 1) {                          +1
        return 1;                         +1
    } else {
        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                System.out.println("hi!");  +1
            }
        }
        return Mystery(n/2)
    }
}
```

1 * n

1 * n * n

$$T(n) = \begin{cases} 1 \ when \ n = 1 \\ T(n/2) + n^2 \text{ if n} > 1 \end{cases}$$