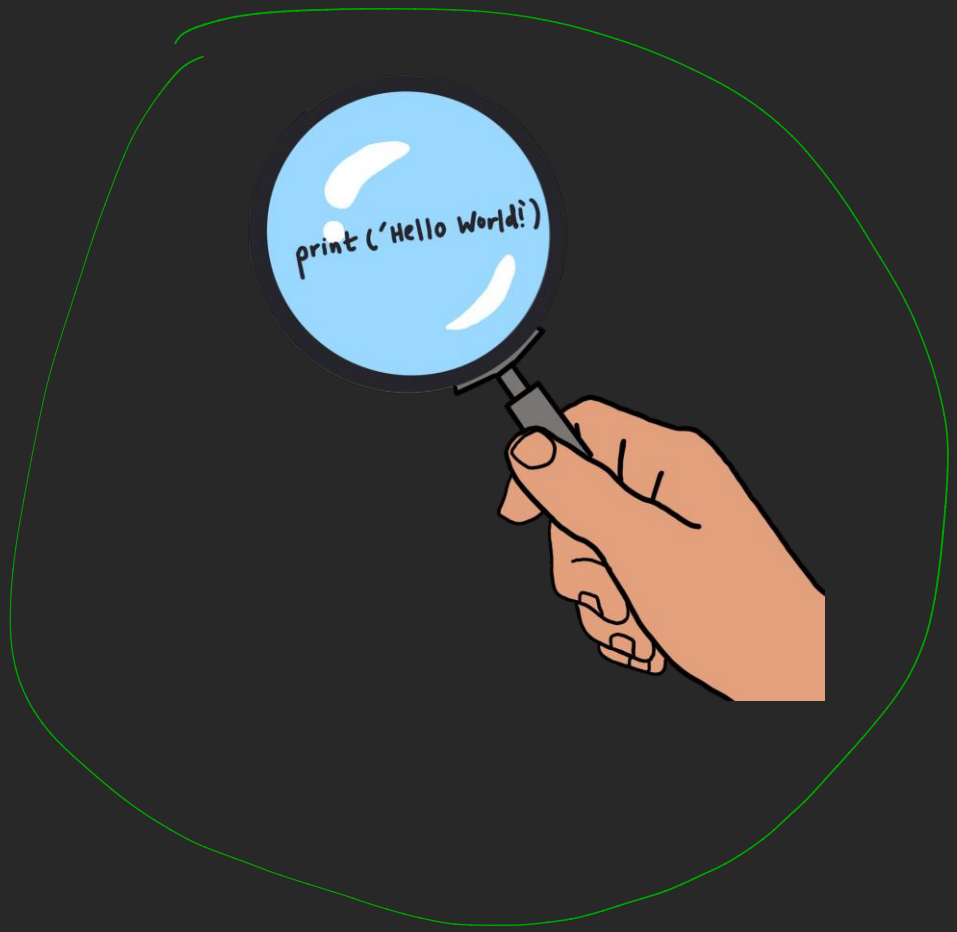
CSCI 3104
Lecture 3: Asymptotic Analysis and Loop Invariant

Caleb Escobedo

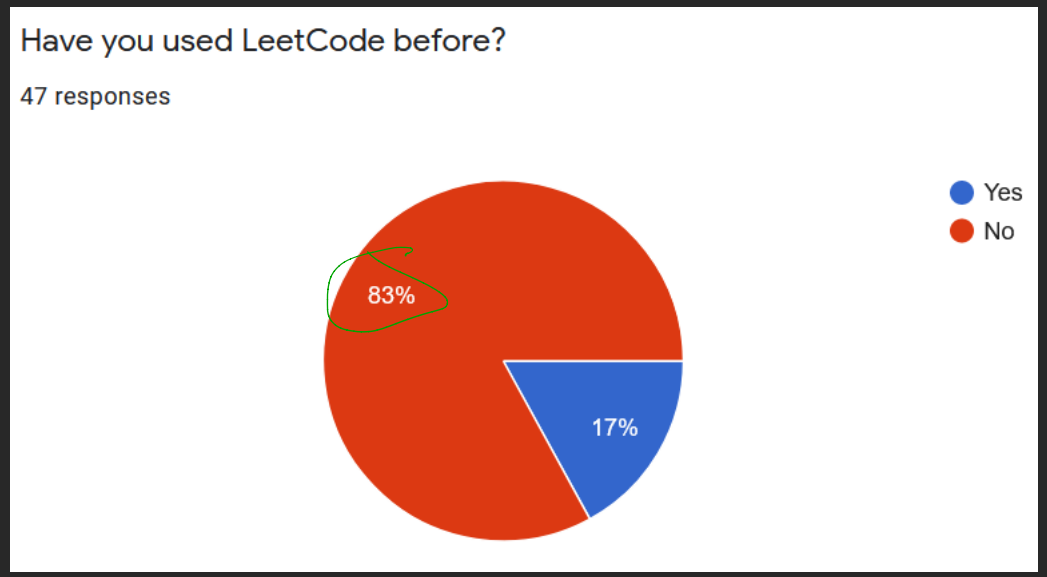
Caleb.Escobedo@colorado.edu

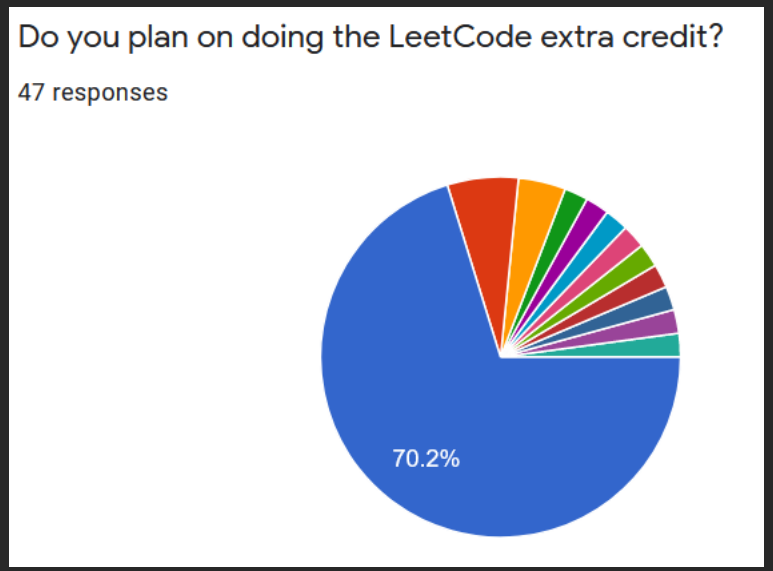
Lecture Outline

- Poll review/Admin
- Big-Θ Review
  - Sloppy thinking
- Asymptotic analysis in practice
  - L'Hopital's Rule
  - Examples 1-3
- Loop Invariant
  - Introduction
  - Insertion Sort
- Analyzing code
  - Simple example
  - Insertion Sort



Poll Review


Have you used LeetCode before?
47 responses
Yes
No
83%
17%


Do you plan on doing the LeetCode extra credit?
47 responses
70.2%

Administrative Update

Reading list is up!!

- Overleaf: download/upload **.tex** file
- Birthday not needed in HW name or first page
- Extra Credit
  - Make it pass all tests
  - Comment all of it in your own words
  - Try not to give the brute force solution

2, 1, 3, 4

1, 2, 3, 4

Lecture Outline

- ~~Poll review/Admin~~
- Big-Θ Review
  - Sloppy thinking
- Asymptotic analysis in practice
  - L'Hopital's Rule
  - Examples 1-3
- Loop Invariant
  - Introduction
  - Insertion Sort
- Analyzing code
  - Simple example
  - Insertion Sort

Big-Θ (Theta) Tight Bound – Review

$2n = \Theta(n)$

1

$\Theta(g(n)) = \{f(n) :$ there exist positive constants c_1, c_2, and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0\}$.[1]

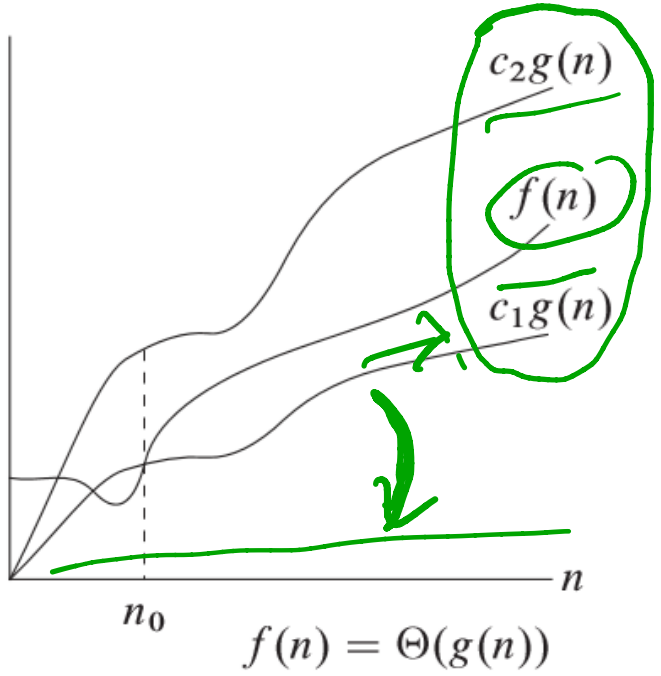
Θ = bounded above and below tight

O = " " " and can be tight

o = " " " " can not be tight

Ω = " below can be tight

ω = " " Can not be tight



Sloppy Thinking

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

$$2n = O(n^2) = O(n^3) = O(1.1^n) = O(1.11^n)$$
$$= O(2^n)$$

Lecture Outline

- ~~Poll review/Admin~~
- ~~Big-O Review~~
  - ~~Sloppy thinking~~
- Asymptotic analysis in practice
  - L'Hopital's Rule
  - Examples 1-3
- Loop Invariant
  - Introduction
  - Insertion Sort
- Analyzing code
  - Simple example
  - Insertion Sort

Asymptotic Analysis in Practice

- 1. Identify the constants n0 and c1 or c2 for Ω or O (or both for Θ), and demonstrate that the formal definitions are satisfied.
- 2. Apply L'Hopital's rule in the limit of $n \to \infty$, in order to obtain the correct functional relation.

L'Hopital's Rule

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{\partial}{\partial n} f(n)}{\frac{\partial}{\partial n} g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \Rightarrow f(n) = \Theta(g(n))$

2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f(n) = \Omega(g(n))$

3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = O(g(n))$

L'Hopital's Rule: Example 1

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{\partial}{\partial n} f(n)}{\frac{\partial}{\partial n} g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$$f(n) = n^2 + n - 1 \text{ and } g(n) = n^2:$$

$$\frac{n^2+n-1}{n^2} \rightarrow \frac{2n+1}{2n} \rightarrow \frac{2}{2} = 1$$

$$f(n) = \Theta(g(n))$$

$\frac{d}{dn} \log(n) = \frac{1}{n}$

L'Hopital's Rule: Example 2

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{\partial}{\partial n} f(n)}{\frac{\partial}{\partial n} g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$$\log(n)\left(\frac{d}{dn}(n)\right) + n\left(\frac{d}{dn}\log n\right)$$

$$\downarrow$$

$$\log(n) + n \cdot \frac{1}{n}$$

$$\log(n) + 1$$

$$f(n) = n \log n + n \text{ and } g(n) = n^2:$$

$$\frac{n\log n + n}{n^2} \rightarrow \lim_{n\to\infty} \frac{2 + \log(n)}{2n} = 0$$

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

L'Hopital's Rule: Example 3

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{\partial}{\partial n} f(n)}{\frac{\partial}{\partial n} g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$x^{\log_x(y)} = y$

$x = e$

$f(n) = 2^n$ and $g(n) = 1.1^n$:

$\frac{2^n}{1.1^n} \to \frac{e^{\ln(2)\cdot n}}{e^{\ln(1.1)\cdot n}} \to e^{\ln(2)n - \ln(1.1)n}$

$\lim_{n\to\infty} e^{n(\ln(2) - \ln(1.1))} = \infty$

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

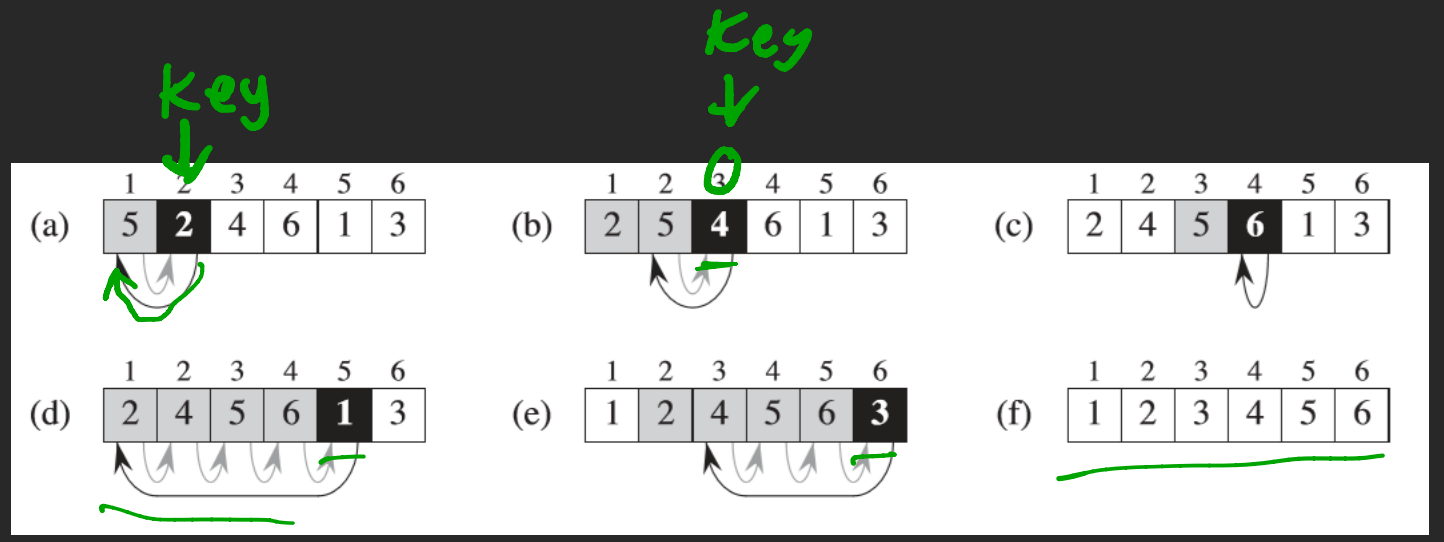
Identities!

1. $(x^y)^z = x^{yz}$
2. $x^y x^z = x^{y+z}$
3. $\log_x y = z \implies x^z = y$
4. $x^{\log_x y} = y$ by definition
5. $\log(xy) = \log x + \log y$
6. $\log(x^c) = c \log x$
7. $\log_c(x) = \log x / \log c$

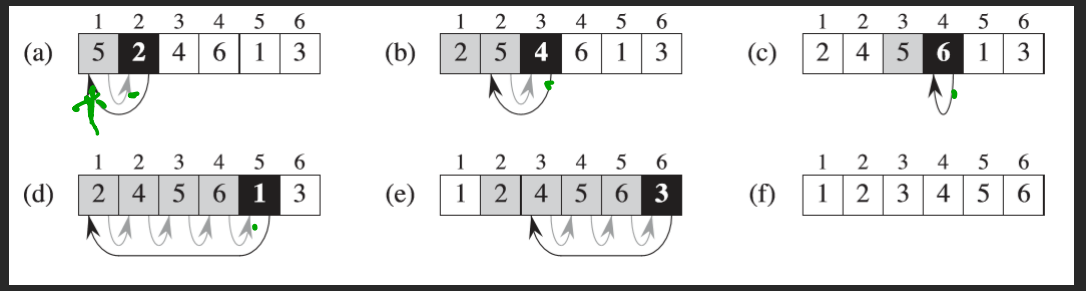
Lecture Outline

- ~~Poll review/Admin~~
- ~~Big-O Review~~
  - ~~Sloppy thinking~~
- ~~Asymptotic analysis in practice~~
  - ~~L'Hopital's Rule~~
  - ~~Examples 1-3~~
- Loop Invariant
  - Insertion Sort: Introduction
  - Example
- Analyzing code
  - Simple example
  - Insertion Sort

Insertion Sort!



Insertion Sort!




```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Loop Invariant: Insertion Sort

At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1 \ldots j-1]$ consists of the elements originally in $A[1 \ldots j-1]$, but in sorted order.

$A[1 \ldots j-1]$

1. **Initialization:** It is true prior to the first iteration of the loop.
2. **Maintenance:** If it is true before an iteration of the loop, it remains true before the next iteration.
3. **Termination:** When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Base Case
Inductive Step

New

Initialization: It is true prior to the first iteration of the loop.

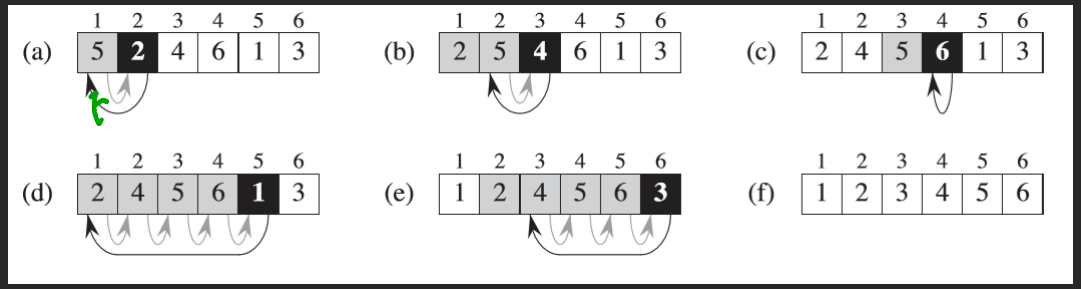
Loop Invariant: Initialization

At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1 \,..\, j-1]$ consists of the elements originally in $A[1 \,..\, j-1]$, but in sorted order.

$A[1 \cdots j-1]$

$A[1 \cdots 2-1]$

$A[1]$ ✓




```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Maintenance: If it is true before an iteration of the loop, it remains true before the next iteration.

Loop Invariant: Maintenance

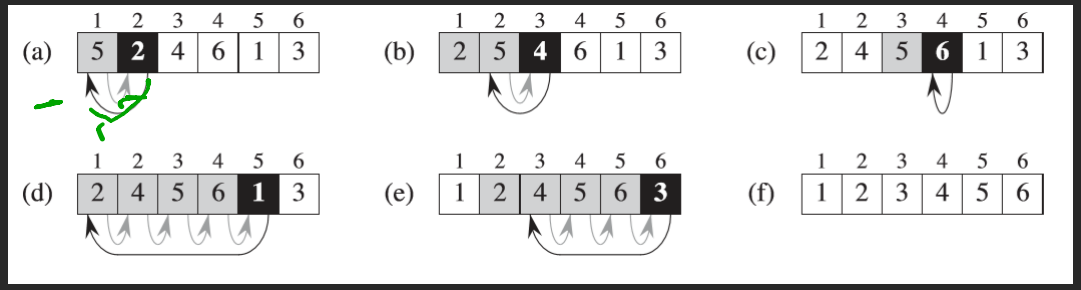
At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1 \,..\, j-1]$ consists of the elements originally in $A[1 \,..\, j-1]$, but in sorted order.

A[j]

A[j-1], A[j-2] ...

A[j] = key

A[1 ... j-1]




```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Termination: When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Loop Invariant: Termination

$j > A.len = n$

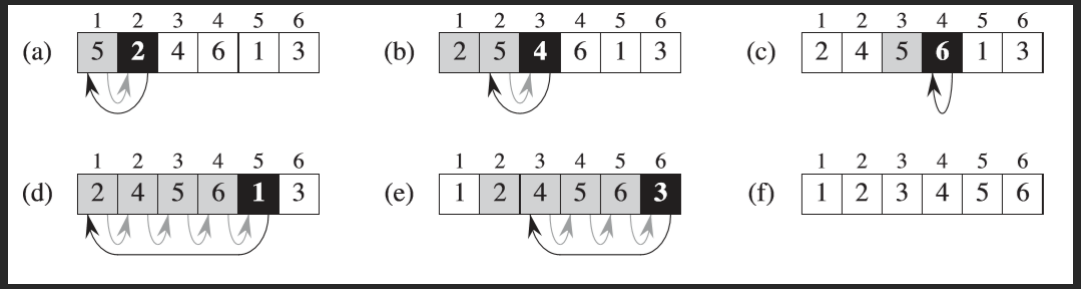
$j = n + 1$

$A[1 \ldots j-1]$

$A[1 \ldots n+1-1]$

$A[1 \ldots n]$

At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1 \ldots j-1]$ consists of the elements originally in $A[1 \ldots j-1]$, but in sorted order.




```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Lecture Outline

- ~~Poll review/Admin~~
- ~~Big-O Review~~
  - ~~Sloppy thinking~~
- ~~Asymptotic analysis in practice~~
  - ~~L'Hopital's Rule~~
  - ~~Examples 1-3~~
- ~~Loop Invariant~~
  - ~~Introduction~~
  - ~~Insertion Sort~~
- Analyzing code
  - Simple example
  - Insertion Sort

Code Analysis: Simple Example

```
computeSum(int n) {
    if n < 1 { return 0 } // catch negative values!
    acc = 0               // initialize the sum
    for i = 1; i<=n; i++ {  // add it up
        acc += i
    }
    return acc            // return the sum
}
```

C = 1
1
1
2
0
0
2
4n
1

$5 + 4n + 1 = O(n)$

Code Analysis: Insertion Sort

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

Code Analysis: Insertion Sort – Best Case

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n}(t_j - 1) + c_7 \sum_{j=2}^{n}(t_j - 1) + c_8(n-1) .$$

$t_j = 0 \ \forall j$

Code Analysis: Insertion Sort – Best Case

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n}(t_j - 1) + c_7 \sum_{j=2}^{n}(t_j - 1) + c_8(n-1) .$$

$t_j = 1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$
$$= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) .$$

$T(n) = an - b = O(n)$

Code Analysis: Insertion Sort – Worst Case

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] = key$	c_8	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\sum_{j=2}^{n} t_j + c_6\sum_{j=2}^{n}(t_j-1) + c_7\sum_{j=2}^{n}(t_j-1) + c_8(n-1)\,.$$

$t_j = j$

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

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

Code Analysis: Insertion Sort – Answer

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right) \\ &\quad + c_6 \left(\frac{n(n-1)}{2} \right) + c_7 \left(\frac{n(n-1)}{2} \right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

$T(n) = an^2 + bn + c$

Closing Thoughts

- Be patient!
- Look over insertion sort analysis a couple of times
- Read, Read, Read