CSE 373: Data Structures and Algorithms

Asymptotic Analysis

Autumn 2018

Shrirang (Shri) Mare
shri@cs.washington.edu

Thanks to Kasey Champion, Ben Jones, Adam Blank, Michael Lee, Evan McCarty, Robbie Weber, Whitaker Brand, Zora Fung, Stuart Reges, Justin Hsia, Ruth Anderson, and many others for sample slides and materials ...

Code Analysis

How do we compare two piece of code? Lots of metrics we could pick!

- Time needed to run
- Memory used
- Number of network calls made
- Amount of data we save to the disk
- Specialized vs. generic
- Code reusability
- Security

(Some metrics are intangible and hard to measure those, e.g., security, code reusability)

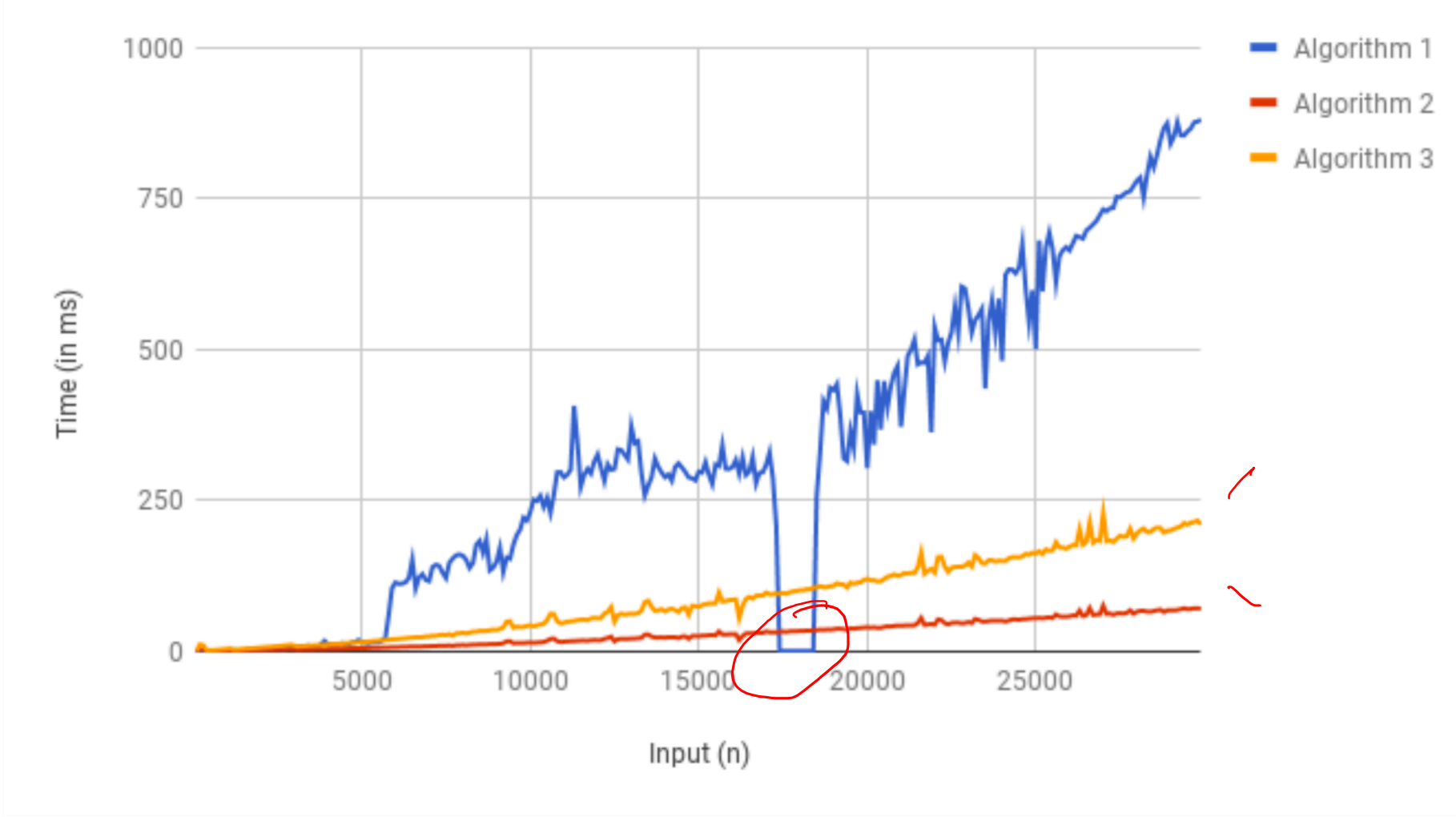
Today: Focus on comparing algorithms based on *how long it takes them to run in the worst case.*

Which of these algorithms is better?

Algorithm	Time (in ms)
Algorithm 1	1
Algorithm 2	30
Algorithm 3	100

This is a trick question. Why isn't this table enough to let us decide which algorithm is better?

Which of these algorithms is better?



Comparing Algorithms

We want:

- To see **overall** trends as input increases
  - Considering a single data point isn't helpful
  - We really care about large inputs
- Final result is independent of incidental factors
  - (CPU speed, programming language, other programs running, etc.)
- Rigorously discover overall trends without resorting to testing
  - What if we miss worst-case input?
- A way to analyze before coding!

What Are We Counting?

Worst case analysis

- For a given input size, what's the running time for the worst state our data structure we can be in or the worst input we can give?

Best case analysis

- What is the number of steps for the best state of our structure and the best question?

Average case analysis

- How are we doing on average over all possible inputs/states of our data structure?
- Have to ask this question very carefully to get a meaningful answer

We usually do worst case analysis.

Asymptotic Analysis: Two step process

1. **Model** what we care about as a mathematical function
2. **Analyze** that function using asymptotic analysis

Modeling: What Are We Counting?

Consecutive statements
- Sum of time of each statement

Function calls
- Time of function's body

Conditionals
- Time of condition + max(if branch, else branch)

Loops
- Number of iterations x time of loop body

Modeling: Assumptions

Assume basic operations take the same constant amount of time.

What's a basic operation?

- Adding ints or doubles
- Assignment
- Incrementing a variable
- A return statement
- Accessing an array index or an object field

What's not a basic operation?

- Making a method call.

This is a LIE but it's a very useful lie.

Modeling Case Study

Goal: return 'true' if a sorted array of ints contains duplicates

Solution 1: compare each pair of elements

```
public boolean hasDuplicate1(int[] array) {
   for (int i = 0; i < array.length; i++) {
      for (int j = 0; j < array.length; j++) {
         if (i != j && array[i] == array[j]) {
            return true;
         }
      }
   }
   return false;
}
```

Solution 2: compare each consecutive pair of elements

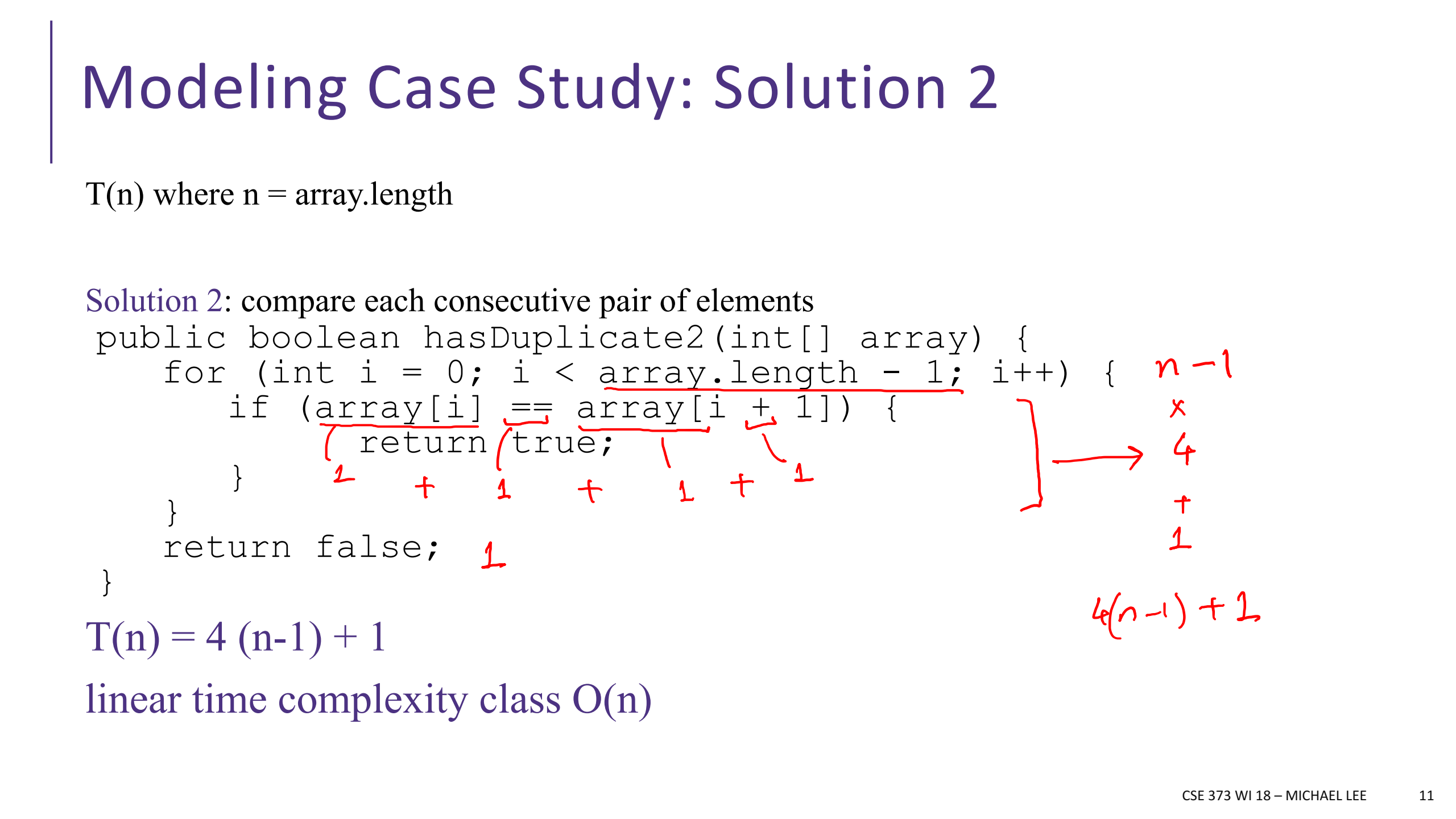
```
public boolean hasDuplicate2(int[] array) {
   for (int i = 0; i < array.length - 1; i++) {
      if (array[i] == array[i + 1]) {
            return true;
      }
   }
   return false;
}
```

Modeling Case Study: Solution 2

T(n) where n = array.length

Solution 2: compare each consecutive pair of elements

```
public boolean hasDuplicate2(int[] array) {
    for (int i = 0; i < array.length - 1; i++) {
        if (array[i] == array[i + 1]) {
            return true;
        }
    }
    return false;
}
```



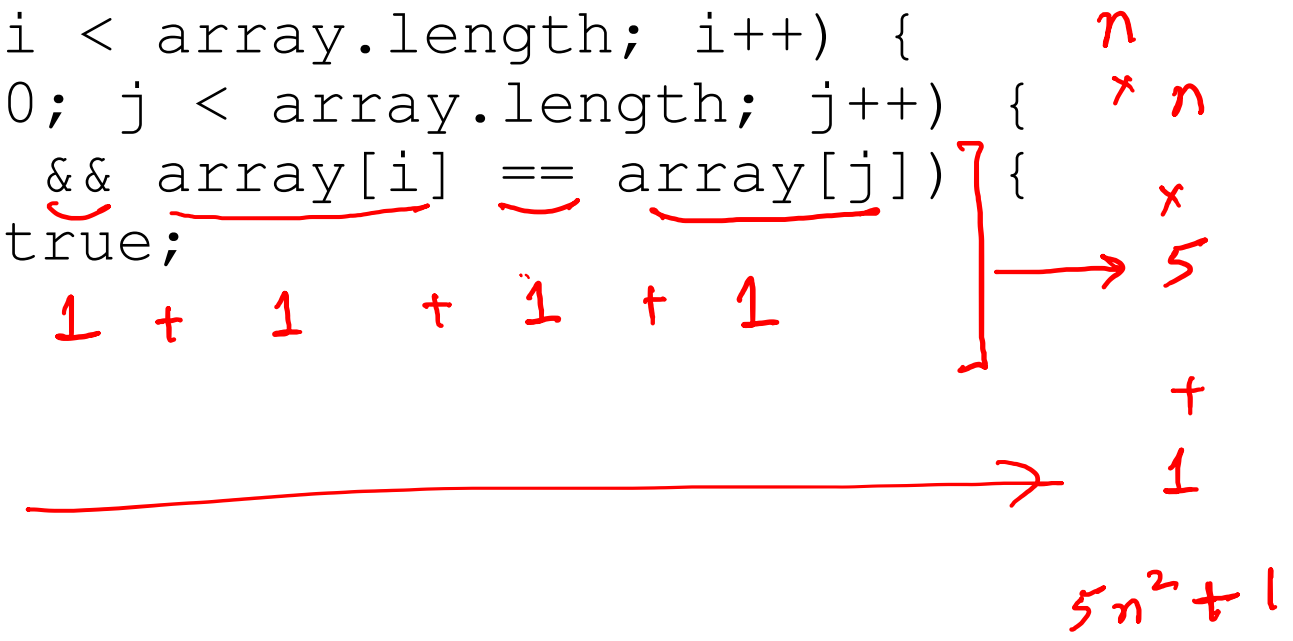
$T(n) = 4 (n-1) + 1$

linear time complexity class O(n)

Modeling Case Study: Solution 1

Solution 1: compare each consecutive pair of elements

```
public boolean hasDuplicate1(int[] array) {
    for (int i = 0; i < array.length; i++) {
        for (int j = 0; j < array.length; j++) {
            if (i != j && array[i] == array[j]) {
                return true;
            }
        }
    }
    return false;
}
```



$T(n) = 5\ n^2 + 1$

quadratic time complexity class $O(n^2)$

Asymptotic Analysis: Two step process

1. **Model** what we care about as a mathematical function

2. **Analyze** that function using asymptotic analysis
   - Specifically: have a way to compare two functions
   - Even more specifically: define a "less then or equal to" operator for functions