Lecture 31:

INTRODUCTION TO ALGORITHMS

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Announcements

- Additional office hours this week:

  Wednesday (today) from 2 to 4

  Friday from 1:30 to 3

  (Friday session can be an exam post-mortem if you want, vote on Slack)

- See Slack for additional tutorials and notes on:
  - DIY buttons using `graphics.py`
  - making your own timer using the `time` module
  - working with `pygame`

Outline

- Algorithmic thinking activity
- Sorting algorithms:
  - Insertion Sort
  - Selection Sort
  - Bubble Sort
- Comparing algorithms
  - on specific examples
  - more generally
- Lab

10 volunteers, please!

Outline

- ✓Algorithmic thinking activity
- Sorting algorithms:
  - Insertion Sort
  - Selection Sort
  - Bubble Sort
- Comparing algorithms
  - on specific examples
  - more generally
- Lab

Debrief

What **similarities / differences** did you notice between the three approaches?

Discussion

How do you know which algorithm is **better**?

1. Consider a specific example

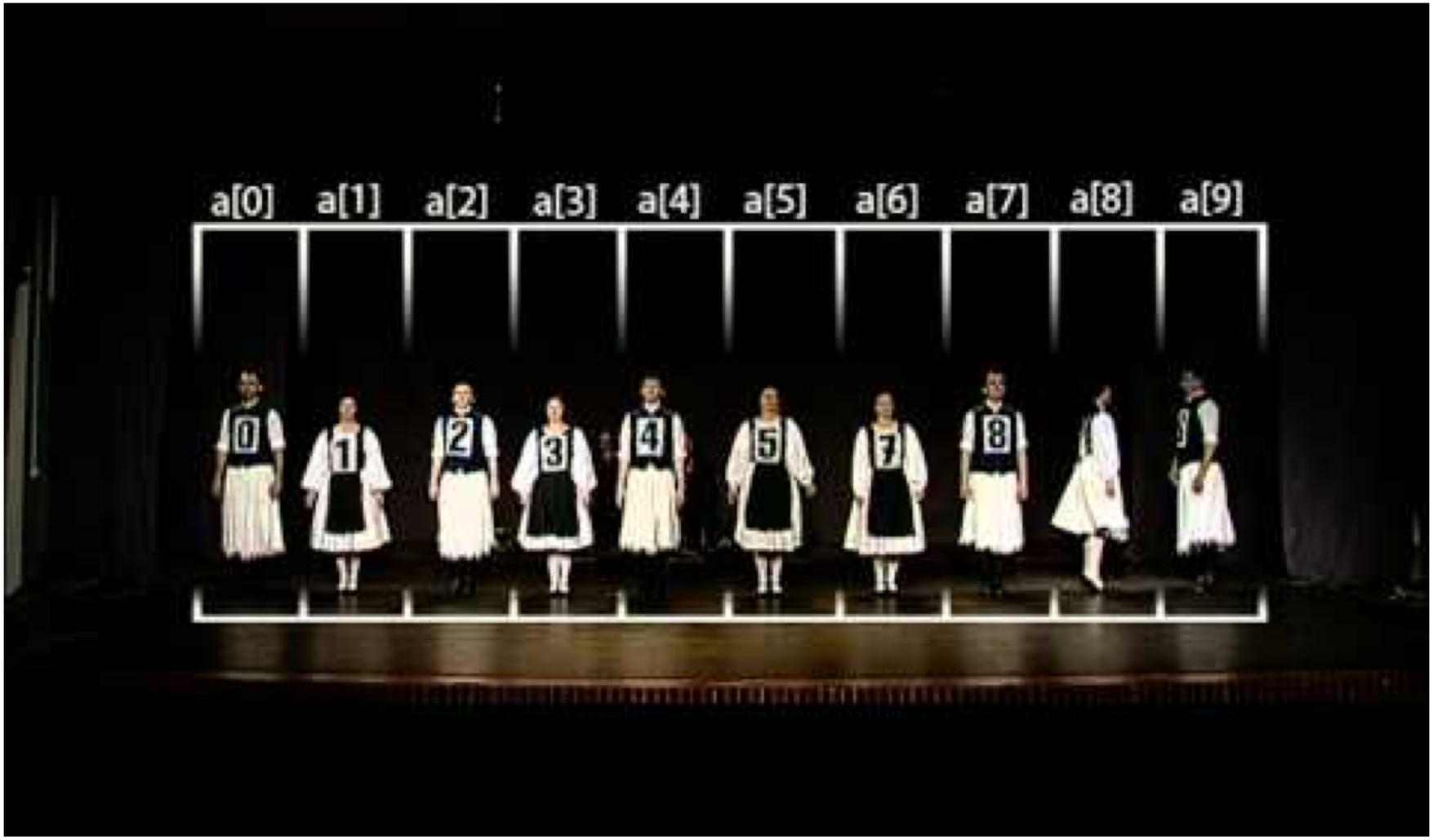
[3, 0, 1, 8, 7, 2, 5, 4, 9, 6]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

InsertionSort: visually



SelectionSort: visually

BubbleSort:visually

2. Now consider a different example

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

3. What about in **general**?

[???]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

Algorithmic analysis

- **What we want**: a way to get a rough bound (limit) on how many steps an algorithm could take, *given input of a particular size*
  - If the list has 5 items?
  - What about 10?
  - What about `n`?
- **Often care about**: "what's the **worst** that could happen?"
- **Mathematics** to the rescue!

Asymptotic ("big-O") notation

- We can **avoid details** when they don't matter, and they don't matter when input size (**n**) is big enough
- For **polynomials**: only the leading term matters
- Ignore **coefficients**, talk about **degree**: linear, quadratic

$y = 3x$ $\quad$ $y = 6x-2$ $\quad$ $y = 15x + 44$

$y = x^2$ $\quad$ $y = x^2-6x+9$ $\quad$ $y = 3x^2+4x$

3. What about in **general**?

[???]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

SelectionSort

Given some list of comparable items:

Find the smallest item.

Swap it with the item in position 0.

Repeat finding the next-smallest item, and swapping it into the correct position until the list is sorted.

BubbleSort

While the list is still unsorted:

  For p in [0, ..., n-2]:

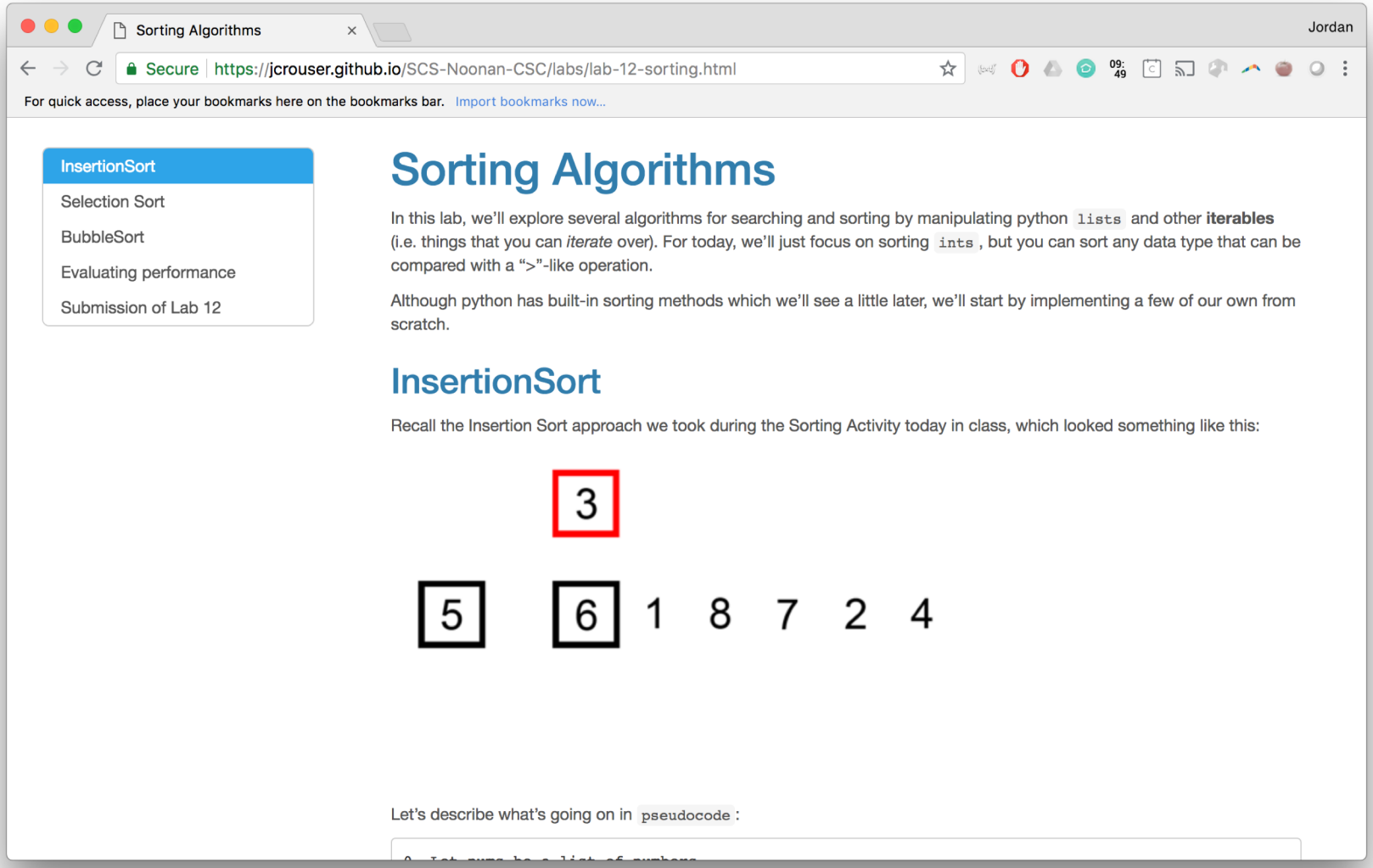
    If item in position p is greater than item in position p+1:

      Swap them

Takeaways

- Big-O notation gives us a language to talk about and compare algorithms **before** we implement them (smart!)
- InsertionSort, SelectionSort, and BubbleSort:
  - all the same in the **worst** case
  - different in the **best** case
  - what about the **average** case?
  - is there any algorithm that could do **better** in the worst case? (yes!)
- As your programs get more complex, it's helpful to think about the **algorithm** before you start coding

Lab: Sorting


Sorting Algorithms
Jordan
Secure https://jcrouser.github.io/SCS-Noonan-CSC/labs/lab-12-sorting.html
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
InsertionSort
Selection Sort
BubbleSort
Evaluating performance
Submission of Lab 12
Sorting Algorithms
In this lab, we'll explore several algorithms for searching and sorting by manipulating python lists and other iterables (i.e. things that you can iterate over). For today, we'll just focus on sorting ints, but you can sort any data type that can be compared with a ">"-like operation.
Although python has built-in sorting methods which we'll see a little later, we'll start by implementing a few of our own from scratch.
InsertionSort
Recall the Insertion Sort approach we took during the Sorting Activity today in class, which looked something like this:
3
5
6
1
8
7
2
4
Let's describe what's going on in pseudocode:

Coming up

- ✓Announcements
- ✓Final Project Proposal Recap
- ✓Working with files
  - ✓what is a file?
  - ✓reading data in
  - ✓writing data out
- ✓Intro to Algorithms
- Lab: Sorting
- Handling `Exceptions`