Intro to Sorting Algorithms

CSSE 221

Fundamentals of Software Engineering Honors
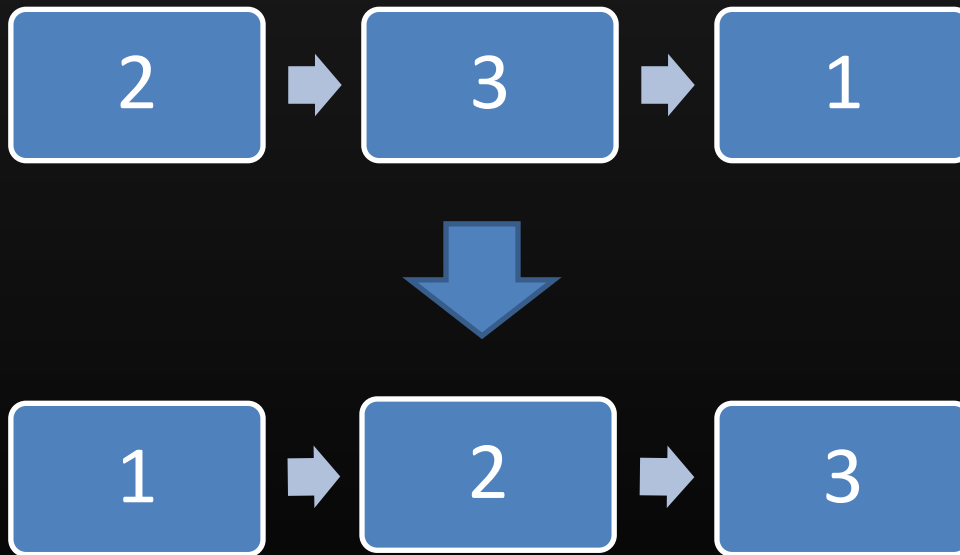
Rose-Hulman Institute of Technology

Understanding the concepts of sorting a collection

Sorting

Sorting

- Ways to arrange data into sorted order

Sorted Order

- What is sorted order?
  - Numeric (int)?
  - Alphabetical (String)?

- Depends on CompareTo() method
  - from **Comparable** interface

Algorithms

- Examples:
  - Bubble Sort
  - Selection Sort
  - Insertion Sort
  - Merge Sort
  - Quick Sort

Slow

Fast

Arrays.sort()

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

How to use and implement a selection sort algorithm

Selection Sort

Selection Overview

- Finds the element of lowest value by searching the entire list

- Then swaps the lowest value with the current first element

- Same efficiency regardless of the collection's initial state

Selection Process

1. Start from the beginning
2. Search the collection for the element which should be placed at the start
3. Swap the found element with the current first element
4. Repeat the process starting from element at second index
5. Continue until starting index is the last element

Example

Given
- [4, 2, 6, 1, 7, 2]

Swap 1
- [1, 2, 6, 4, 7, 2]

Swap 2
- [1, 2, 6, 4, 7, 2]

Swap 3
- [1, 2, 2, 4, 7, 6]

Swap 4
- [1, 2, 2, 4, 7, 6]

Swap 5
- [1, 2, 2, 4, 6, 7]

Min value

Swap

How to use and implement an insertion sort

Insertion Sort

Insertion Overview
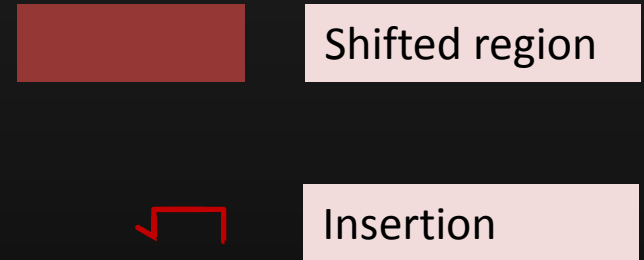
- Looks at an element and those left of it
- Then shifts all elements of higher value to the right and inserts the element
- Continues until the collection is full

- Efficiency changes based on initial state of the collection

Insertion Process

1. Start with the second element
2. Look to the left
3. Elements of higher value than selected element shift right
4. Insert element before those shifted
5. Repeat for each index in the collection until the end is reached

Example

- [4, 2, 6, 1, 7, 2]

- [2, 4, 6, 1, 7, 2]

- [2, 4, 6, 1, 7, 2]

- [1, 2, 4, 6, 7, 2]

- [1, 2, 4, 6, 7, 2]

- [1, 2, 2, 4, 6, 7]

Shifted region

Insertion

Efficiency (Big-Oh Analysis)

- Selection & Insertion have the same efficiency when a collection is unsorted
- But Insertion works faster on a partially sorted array

	Unsorted	Sorted
Selection Sort	• $O(n^2)$	• $O(n^2)$
Insertion Sort	• $O(n^2)$	• $O(n)$

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Examining the properties of Insertion and Selection sorts

Demo

Implementing sorting algorithms

Activity