Northeastern
University

Lecture 25: An Introduction to Sorting - 1

Prof. Chen-Hsiang (Jones) Yu, Ph.D.

College of Engineering

Materials are edited by Prof. Jones Yu from

Data Structures and Abstractions with Java, 5th edition. By Frank M. Carrano and Timothy M. Henry.
ISBN-13 978-0-13-483169-5 © 2019 Pearson Education, Inc.

Sorting

- Arranging things into either ascending or descending order is called sorting.
- We seek algorithms to arrange items, a_i, such that

  $a_1 \leq a_2 \leq . . . \leq a_n$
- Sorting an array is usually easier than sorting a chain of linked nodes.
- Efficiency of a sorting algorithm is significant.

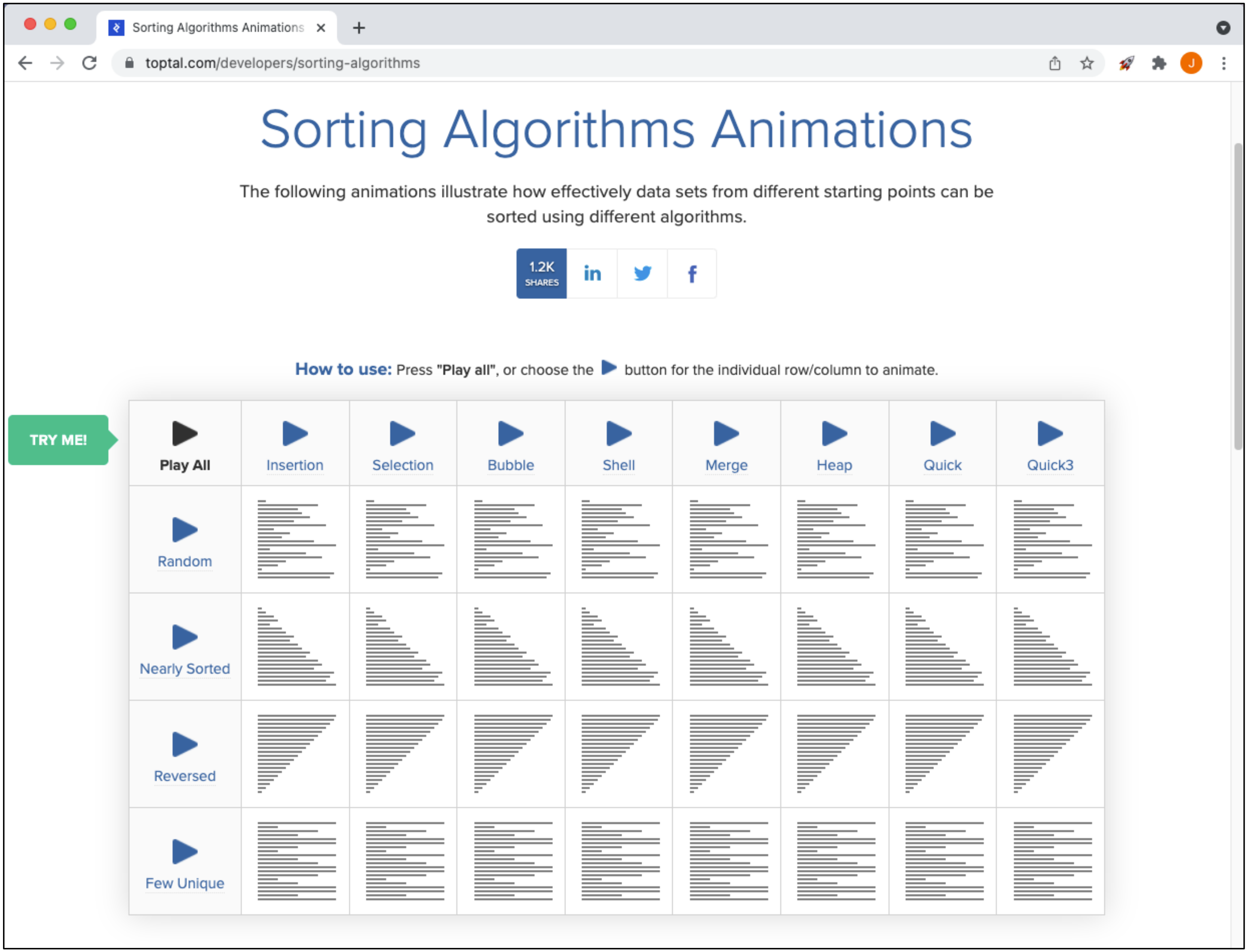
Sorting Algorithms

- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Quick Sort
- Radix Sort
- Bubble sort
- Heapsort
- Introsort
- Timsort
- Cubesort
- …

Sorting Algorithms

- Selection Sort
- Insertion Sort
- Shell Sort
- Merge Sort
- Quick Sort
- Radix Sort

- Bubble sort
- Heapsort
- Introsort
- Timsort
- Cubesort
- …


Sorting Algorithms Animations
toptal.com/developers/sorting-algorithms
1.2K SHARES
TRY ME!

Sorting Algorithms Animations

The following animations illustrate how effectively data sets from different starting points can be sorted using different algorithms.

How to use: Press **"Play all"**, or choose the ▶ button for the individual row/column to animate.

▶ Play All	▶ Insertion	▶ Selection	▶ Bubble	▶ Shell	▶ Merge	▶ Heap	▶ Quick	▶ Quick3
▶ Random								
▶ Nearly Sorted								
▶ Reversed								
▶ Few Unique								

Sorting Algorithm Animation: http://www.sorting-algorithms.com/

Selection Sort

Selection Sort

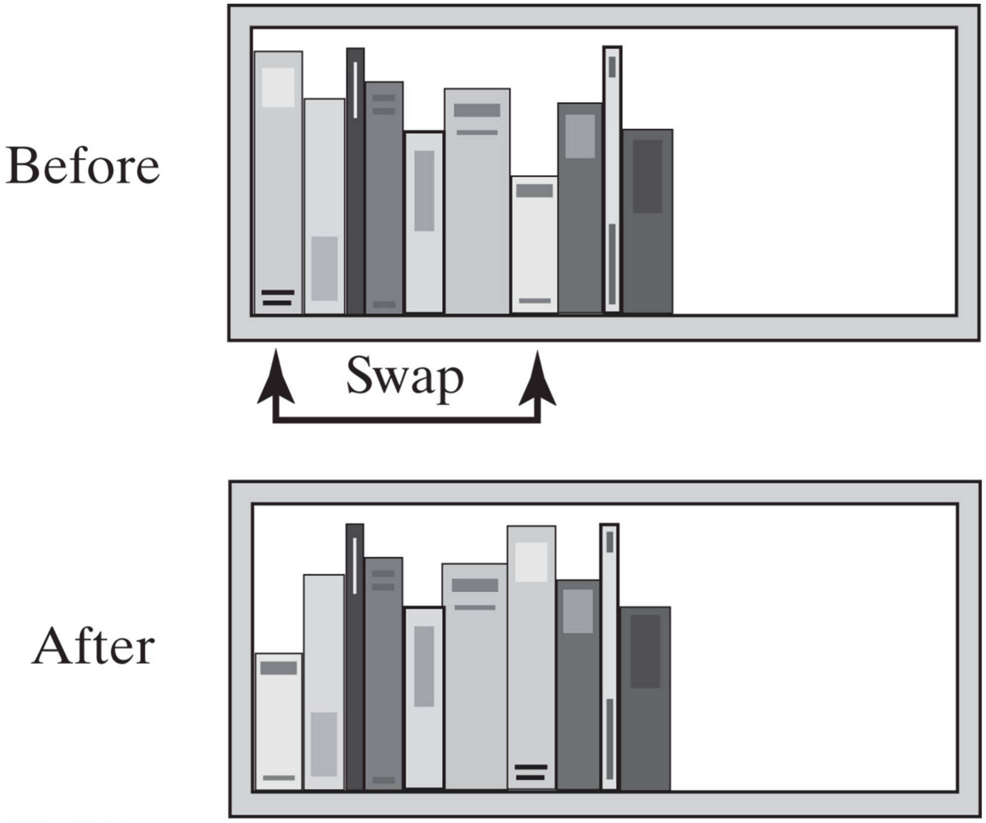


Figure 15-1: Before and after exchanging the shortest book and the first book

Selection Sort

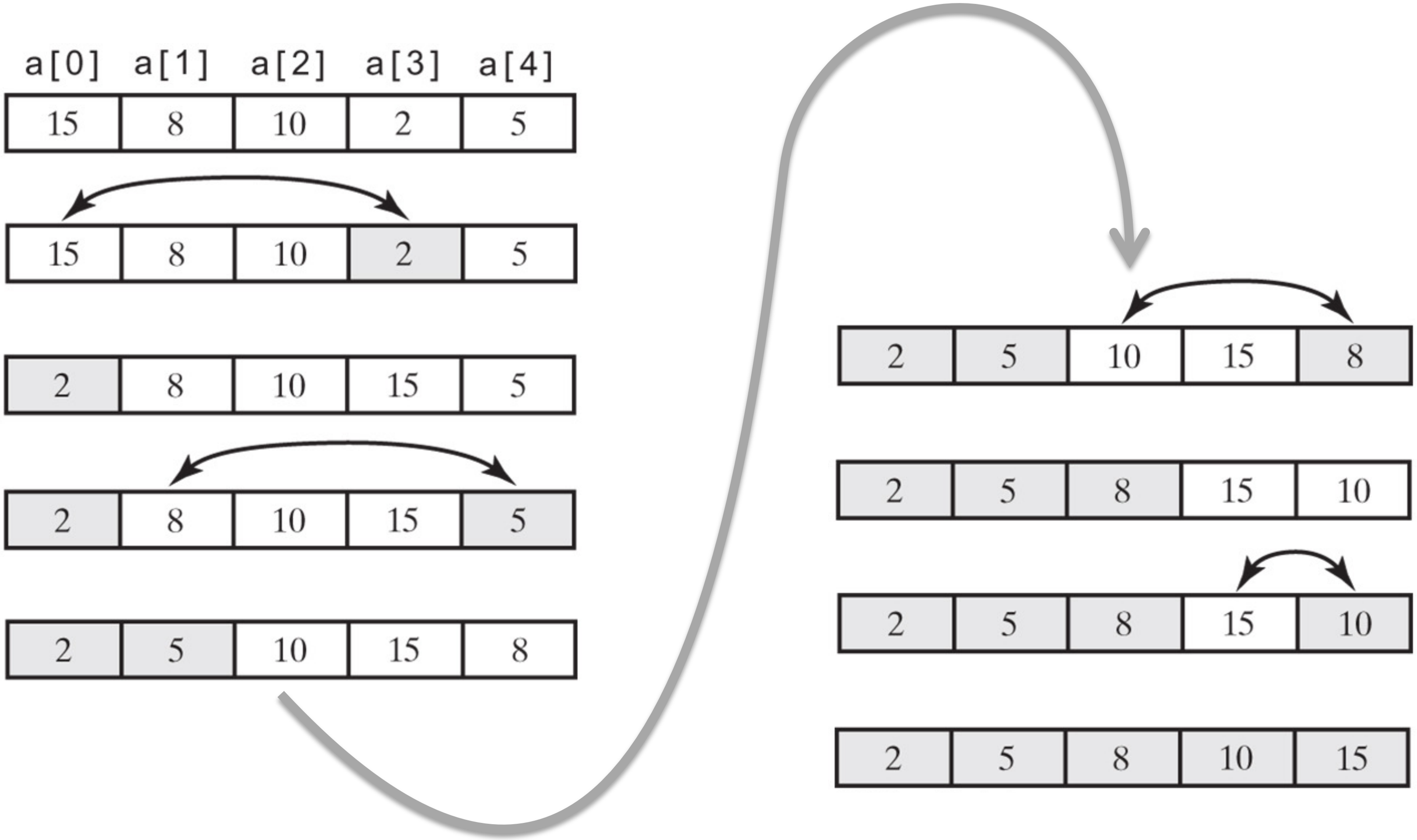


Figure 15-2: A selection sort of an array of integers into ascending order

Iterative Selection Sort

```
Algorithm selectionSort(a, n)

// Sorts the first n entries of an array a.
for (index = 0; index < n – 1; index++)
{

  indexOfNextSmallest = the index of the smallest value among
                                  a[index], a[index + 1], . . . , a[n – 1]
  Interchange the values of a[index] and a[indexOfNextSmallest]

  // Assertion: a[0] ≤ a[1] ≤ . . . ≤ a[index], and these are the smallest
  // of the original array entries. The remaining array entries begin at a[index + 1].

}
```

This pseudocode describes an iterative algorithm for the selection sort

Iterative Selection Sort - Part 1

```
/**  A class of static, iterative methods for sorting an array of
   Comparable objects from smallest to largest. */
public class SortArray
{
   /** Sorts the first n objects in an array into ascending order.
       @param a  An array of Comparable objects.
       @param n  An integer > 0. */
   public static <T extends Comparable<? super T>>
          void selectionSort(T[] a, int n)
   {
      for (int index = 0; index < n – 1; index++)
      {
         int indexOfNextSmallest = getIndexOfSmallest(a, index, n – 1);
         swap(a, index, indexOfNextSmallest);
      // Assertion: a[0] <= a[1] <= . . . <= a[index] <= all other a[i]
      } // end for
   } // end selectionSort
```

Listing 15-1: A class for sorting an array using selection sort

Iterative Selection Sort - Part 2

```java
// Finds the index of the smallest value in a portion of an array a.
// Precondition: a.length > last >= first >= 0.
// Returns the index of the smallest value among
// a[first], a[first + 1], . . . , a[last].
private static <T extends Comparable<? super T>>
        int getIndexOfSmallest(T[] a, int first, int last)
{
   T min = a[first];
   int indexOfMin = first;
   for (int index = first + 1; index <= last; index++)
   {
      if (a[index].compareTo(min) < 0)
      {
         min = a[index];
         indexOfMin = index;
      } // end if
      // Assertion: min is the smallest of a[first] through a[index].
   } // end for

   return indexOfMin;
} // end getIndexOfSmallest
```

Listing 15-1: A class for sorting an array using selection sort

Iterative Selection Sort - Part 3

```
  // Swaps the array entries a[i] and a[j].
   private static void swap(Object[] a, int i, int j)
   {
      Object temp = a[i];
      a[i] = a[j];
      a[j] = temp;
   } // end swap
} // end SortArray
```

Listing 15-1: A class for sorting an array using selection sort

Recursive Selection Sort

```
Algorithm selectionSort(a, first, last)

// Sorts the array entries a[first] through a[last] recursively.
if (first < last)
{
  indexOfNextSmallest = the index of the smallest value among
                                  a[first], a[first + 1], . . . , a[last]
  Interchange the values of a[first] and a[indexOfNextSmallest]

  // Assertion: a[0] ≤ a[1] ≤ . . . ≤ a[first] and these are the smallest
  // of the original array entries. The remaining array entries begin at a[first + 1].

  selectionSort(a, first + 1, last)
}
```

Recursive selection sort algorithm

Exercise

- Show the contents of the array of integers 5 7 4 9 8 5 6 3 each time a selection sort changes it while sorting the array into ascending order.

Answer

Initial array:

5 7 4 9 8 5 6 3

Array after each selection and swap:

5 7 4 9 8 5 6 3

3 7 4 9 8 5 6 5

3 4 7 9 8 5 6 5

3 4 5 9 8 7 6 5

3 4 5 5 8 7 6 9

3 4 5 5 6 7 8 9

Two more passes are performed, but they just swap the 7 with itself and the 8 with itself.

Efficiency of Selection Sort

- Selection sort is $O(n^2)$ regardless of the initial order of the entries.
  - Requires $O(n^2)$ comparisons
  - Does only $O(n)$ swaps

Exercise

- Write pseudocode for a selection sort algorithm that selects the largest, instead of the smallest, entry in the array and sorts the array into descending order.

Answer

```
Algorithm selectionSort(a, n)

// Sorts the first n entries of an array a into descending order.

for (index = 0; index < n − 1; index++)
{
   indexOfNextLargest = the index of the largest value among a[index],
                                 a[index + 1], . . . , a[n − 1]

   Interchange the values of a[index] and a[indexOfNextLargest]

   // Assertion: a[0] ≥ a[1] ≥ . . . ≥ a[index], and these are the largest of
   // the original array entries.

   // The remaining array entries begin at a[index + 1].
}
```