Sorting

# • *Sorting* is the process of arranging a list of items in a particular order

# • The sorting process is based on specific value(s)

* sorting a list of test scores in ascending numeric order
* sorting a list of people alphabetically by last name

• There are many algorithms, which vary in efficiency, for sorting a list of items

• Below are 3 Sorting Algorithms –   
Selection, Insertion, and Bubble Sort

# Bubble Sort

• The approach of Bubble Sort:

* Loop j based on the number of items:
  + Loop k based on the number of items - 1:
    - Swap each element with its neighbor, if it is larger (or smaller for descending order).
  + Repeat until the largest item in this set is in the right spot.
* repeat until all items have been sorted

# Selection Sort

* The approach of Selection Sort:
  + select a value and put it in its final place in the list
  + repeat for all other values

* **In more detail:** 
  + Starting at the beginning of the list, find the smallest value in this sub-list and switch it with the value in the 1st position
  + find the next smallest value in the list and switch it with the value in the 2nd position
  + repeat until all values are in their proper places
  + Each time, the next smallest remaining value is found and exchanged with the element in the "next" position to be filled

# Insertion Sort

• The approach of Insertion Sort:

* pick any item and insert it into its proper place in a sorted sublist
* repeat until all items have been inserted

• **In more detail:**

* consider the first item to be a sorted sublist (of one item)
* insert the second item into the sorted sublist, shifting the first item as needed to make room to insert the new addition
* insert the third item into the sorted sublist (of two items), shifting items as necessary
* repeat until all values are inserted into their proper positions

**Visual Demonstration of Sorting Algorithms:** <http://math.hws.edu/eck/jsdemo/sortlab.html><https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html>

**public** **static** **void** main(String[] args) {

String[] data1 = { "30", "12", "19", "5", "26", "44", "23", "6"};

String[] data2 = { "30", "12", "19", "5", "26", "44", "23", "6"};

String[] data3 = { "30", "12", "19", "5", "26", "44", "23", "6"};

***selectionSort***(data1, data1.length);

***insertionSort***(data2, data2.length);

***bubbleSort***(data3, data3.length);

*display*(data1, data1.length);

*display*(data2, data2.length);

*display*(data3, data3.length);

}

**public** **static** **void** display(**String**[] list, **int** n) {

**for** (**int** i = 0; i < n; i++)

System.***out***.print(list[i] + " ");

System.***out***.println();

}

**public** **static** **void** bubbleSort(**String**[] data, **int** n) {

// check n to make sure that it doesn't

// exceed the array bounds

**if** (n <= 0 || n > data.length) n = data.length;

**for** (**int** i = 0; i < (n - 1); i++) {

**for** (**int** j = 0; j < (n - i - 1); j++) {

**if** (data[j].compareTo(data[j + 1]) > 0) {

// swap adjacent items

**String** temp = data[j];

data[j] = data[j + 1];

data[j + 1] = temp;

} // end if

}// end for j

} // end for i

}

**public** **static** **void** selectionSort(**String**[] data, **int** n) {

// check n to make sure that it doesn't exceed the array bounds

**if** (n <= 0 || n > data.length) n = data.length;

**for** (**int** i = 0; i < n - 1; i++) {

// find the smallest of the unsorted values

**int** smallest = i;

**for** (**int** j = i + 1; j < n; j++) {

**if** (data[j].compareTo(data[smallest]) < 0)

smallest = j;

} // end for j

// Swap (exchange) the value at [i] with the value at [smallest]

**String** temp = data[smallest];

data[smallest] = data[i];

data[i] = temp;

} // end for i

}

**public** **static** **void** insertionSort(**String**[] data, **int** n) {

**int** position = -1;

// check n to make sure that it doesn't exceed the array bounds

**if** (n <= 0 || n > data.length) n = data.length;

// for each value starting from the second to the end of the array

**for** (**int** i = 1; i < n; i++) {

// store value and its position

**String** temp = data[i];

position = i;

// shift values on the left side of the current position

// that are larger than value at current position, to the right

**while** (position > 0 && data[position - 1].compareTo(temp) > 0) {

data[position] = data[position - 1];

position--;

} // end while

// move the stored value (temp) into position

data[position] = temp;

} // end for i

}