Sorting I Worksheet: Comparison-Based Sorts

Selection Sort

Strategy: Select items from the unsorted portion, and add to the sorted portion
Main Idea:

1. Find the smallest number from the unsorted portion of the list
2. Swap that number to the end of the sorted portion of the list
3. Continue until the list is fully sorted.

Insertion Sort

Strategy: Bubble in items from the unsorted portion into the sorted portion
Main Idea:

1. Consider the item in the front of the unsorted portion of the list
2. Bubble that item into the correct position in the sorted portion of the list.
3. Continue until the list is fully sorted.

Merge Sort

Strategy: First recurse on the left/right halves, then do the 'sorting work'
Main Idea:

1. Split the list into left and right halves, and recursively sort the halves
2. **Merge** the sorted left half with the sorted right half

Quick Sort

Strategy: First do the 'sorting work', then recurse on the left/right halves
Main Idea:

1. Choose an element to be the pivot (let's say this is the first element of the list)
2. For all other elements in the list: if the item is less than the pivot, move it to the left of the pivot. Otherwise, move it to the right of the pivot
3. Recursively sort the list to the left of the pivot, and the list to the right of the pivot.
4. Combine the sorted left list, the pivot, and the sorted right list.

Best/Worst-Case Runtimes

	Selection	Insertion	Merge	Quick
Best-Case				
Worst-Case				

Practice Problems

Distinguishing Sorts

Below you will find intermediate steps in performing various sorting algorithms on the same input list. The steps do not necessarily represent consecutive steps in the algorithm (that is, many steps are missing), but they are in the correct sequence. For each of them, select the algorithm it illustrates from among the following choices: insertion sort, selection sort, mergesort, and quicksort (where the first element of sequence is the pivot).

Input list: 1429, 3291, 7683, 1337, 192, 594, 4242, 9001, 4392, 129, 1000

(a) _____

1429, 3291, 7683, 192, 1337, 594, 4242, 9001, 4392, 129, 1000

1429, 3291, 192, 1337, 7683, 594, 4242, 9001, 129, 1000, 4392

192, 1337, 1429, 3291, 7683, 129, 594, 1000, 4242, 4392, 9001

(b) _____

1337, 192, 594, 129, 1000, 1429, 3291, 7683, 4242, 9001, 4392

192, 594, 129, 1000, 1337, 1429, 3291, 7683, 4242, 9001, 4392

129, 192, 594, 1000, 1337, 1429, 3291, 4242, 9001, 4392, 7683

(c) _____

1337, 1429, 3291, 7683, 192, 594, 4242, 9001, 4392, 129, 1000

192, 1337, 1429, 3291, 7683, 594, 4242, 9001, 4392, 129, 1000

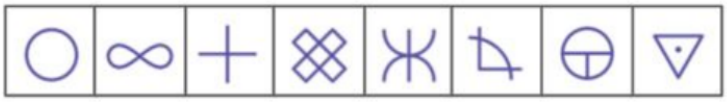
192, 594, 1337, 1429, 3291, 7683, 4242, 9001, 4392, 129, 1000

Reverse Engineering

Consider the following unsorted array, and the array after an unknown number of iterations of selection sort as discussed in class (where we sort by identifying the minimum item and moving it to the front by swapping). Assume no two elements are equal.

Unsorted:

After ? Iterations of Selection Sort:



For each relation below, write $<$, $>$, or ? for insufficient information regarding the relation between the two objects.

Solutions available on http://datastructur.es/sp17/materials/discussion/discussion13epsol.pdf