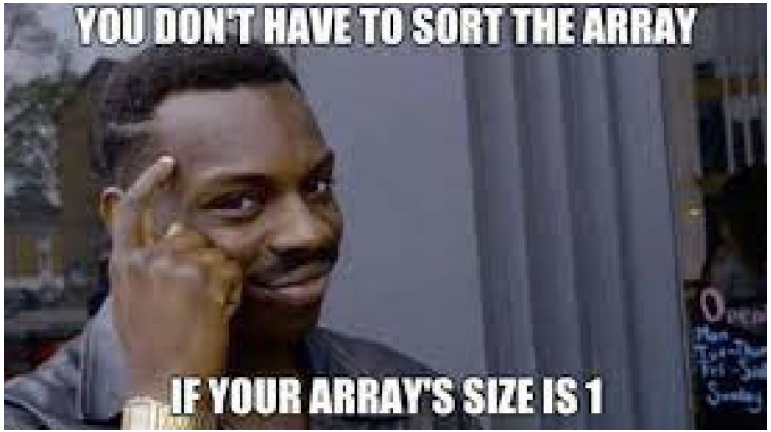
CS 1332R
WEEK 10



Selection Sort

Insertion Sort

Bubble Sort

Cocktail Shaker Sort

ANNOUNCEMENTS

-

Introduction to Sorting

ITERATIVE SORTS

- ➔ Selection
- ➔ Insertion
- ➔ Bubble
- ➔ Cocktail Shaker

NON-COMPARISON BASED SORTS

- ➔ LSD Radix Sort

RECURSIVE SORTS

- ➔ Merge Sort
- ➔ QuickSort

Sorting Algorithms: Properties

IN-PLACE / OUT-OF-PLACE

- **In-Place:** does not use any external data structures to implement the sort, all sorting is done *within the original array*
- **Out-of-Place**: uses external data structures to implement the sort, ***not O(1) space complexity***

STABILITY

- **Stable:** duplicate data maintains its relative order in the array after the sort
- **Not Stable**: duplicate date could change relative order throughout the sort

Why would we want stability?

So we can change the order Objects by one of their attributes without sacrificing the order of another attribute.

ADAPTABILITY

- **Adaptive:** takes advantage of the parts of the array that are already sorted, reducing the number of comparisons needed
- **Not Adaptive**: behaves the same way whether the data is completely sorted or unsorted

What could time complexity tell us about adaptivity?

If a sort is not adaptive, it will have the same best & worst case time complexity.

Selection Sort

INTUITION:

Find the maximum element in the list, swap it to the end.

Which piece of data is always in its final location after the ith iteration?

The ith largest element

PSEUDOCODE:

```
procedure SelectionSort(array):
  length is array's length
  for (i from end to start):
    initialize max value
    for (j from i to end):
      if (array[j] > max value):
        max value is array[j]
      end if
    end for
    swap max value with array[i]
  end for
end procedure
```

Selection Sort: Properties & Efficiency

PSEUDOCODE
(available on csvistool)

```
procedure SelectionSort(array):
  length is array's length
  for (i from end to start):
    initialize max value
    for (j from i to end):
      if (array[j] > max value):
        max value is array[j]
      end if
    end for
    swap max value with array[i]
  end for
end procedure
```

Place	Stable	Adaptive	Best	Average	Worst
In	*No*	*No*	$O(n^2)$	$O(n^2)$	$O(n^2)$

Selection Sort: Practice

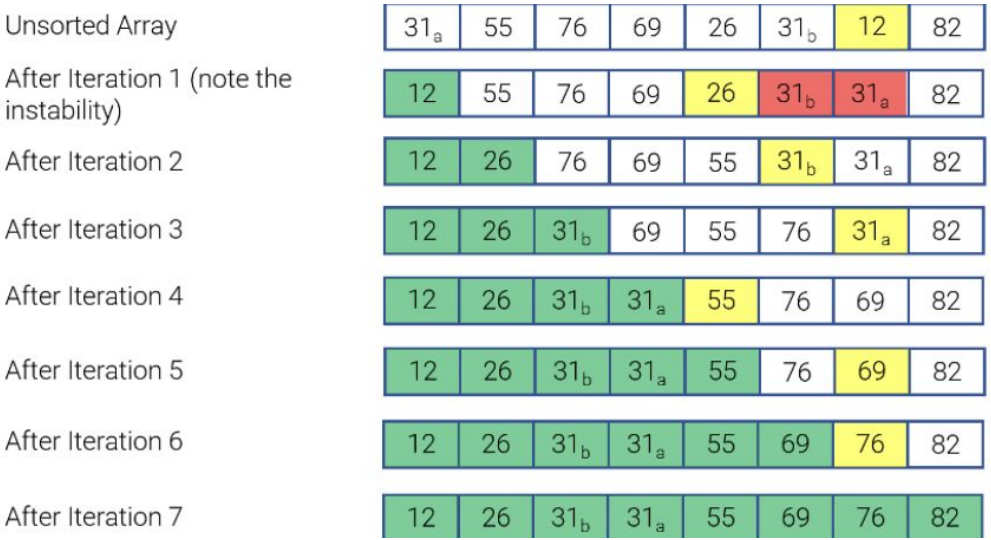
Diagram this array after every iteration of selection sort using the minimum element, noting which portion of the array is sorted.

Unsorted Array

After Iteration 1 (note the instability)

Selection Sort: Practice

Diagram this array after every iteration of selection sort using the minimum element, noting which portion of the array is sorted.



Insertion Sort

INTUITION:

- *Divide the array into an unsorted and sorted portion.*
- *“Insert” the next element in the array into the sorted portion.*

What is the state of the array after i iterations?

The first i + 1 elements are relatively sorted

PSEUDOCODE:

```
procedure InsertionSort(array):
  length is array's length
  for (i from start to end):
    j points to i
    while (j is positive and array[j - 1] > array[j]):
      bubble array[j] by swapping down
      decrement j
    end while
  end for
end procedure
```

Insertion Sort: Properties & Efficiency

PSEUDOCODE
(available on csvistool)

```
procedure InsertionSort(array):
  length is array's length
  for (i from start to end):
    j points to i
    while (j is positive and array[j - 1] > array[j]):
      bubble array[j] by swapping down
      decrement j
    end while
  end for
end procedure
```

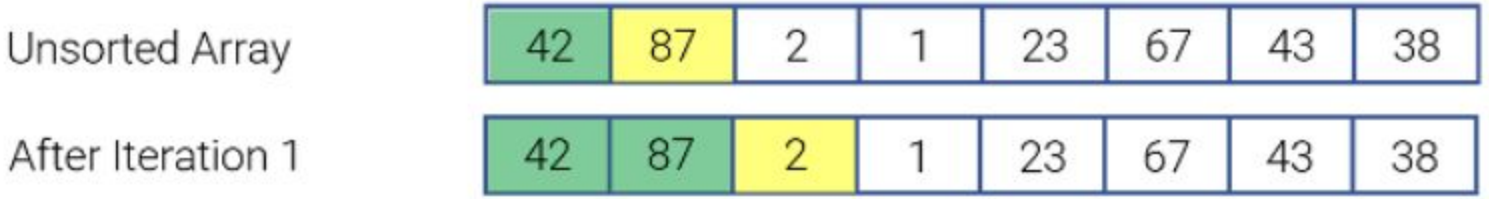
Place	Stable	Adaptive	Best	Average	Worst
In	*Yes*	*Yes*	*O(n)*	*$O(n^2)$*	*$O(n^2)$*

sorted order

reverse sorted order

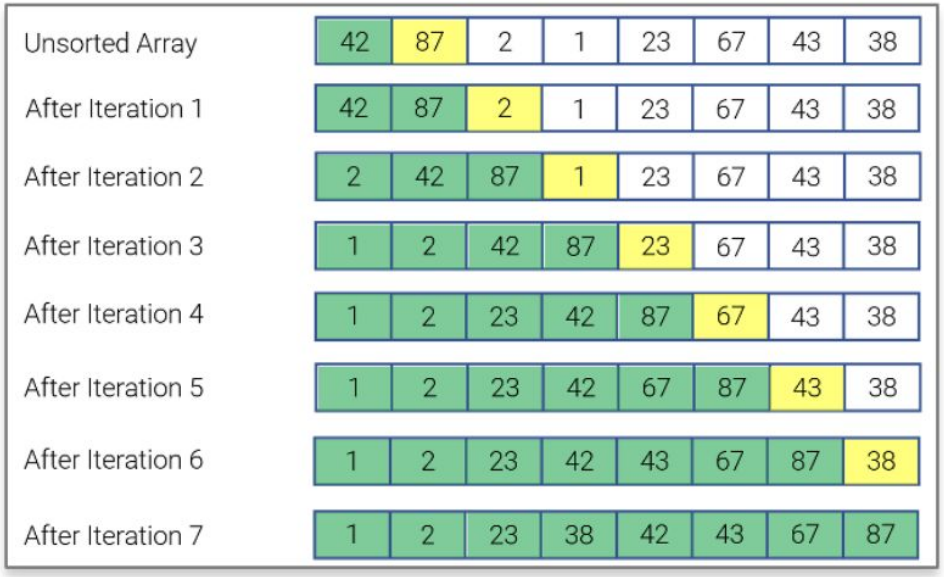
Insertion Sort: Practice

Diagram this array after every iteration of insertion sort, noting which portion of the array is sorted.



Insertion Sort: Practice

Diagram this array after every iteration of insertion sort, noting which portion of the array is sorted.



Bubble Sort

INTUITION:

Swap adjacent elements that are out-of-order, "bubbling" the largest element to the end.

Which piece of data is always in its final location after the ith iteration?

The ith largest element

PSEUDOCODE:

```
procedure BubbleSort(array):
  end points to last element
  start points to first element
  while (start < end):
    lastSwapped points to start
    for (j from start to end):
      if (array[j] > array[j + 1]):
        bubble array[j] by swapping up
        lastSwapped points to j
      end if
    end for
    end points to lastSwapped
  end while
end procedure
```

Bubble Sort

OPTIMIZATIONS:

#1: swapsMade:

→ If no swaps were made in one iteration, the array is completely sorted.

***#2: lastSwapped*:**

→ End an iteration at the lastSwapped index of the previous iteration.

MUST USE THIS IMPLEMENTATION

PSEUDOCODE:

```
procedure BubbleSort(array):
  end points to last element
  start points to first element
  while (start < end):
    lastSwapped points to start
    for (j from start to end):
      if (array[j] > array[j + 1]):
        bubble array[j] by swapping up
        lastSwapped points to j
      end if
    end for
    end points to lastSwapped
  end while
end procedure
```

Bubble Sort: Properties & Efficiency

PSEUDOCODE
(available on csvistool)

```
procedure BubbleSort(array):
  end points to last element
  start points to first element
  while (start < end):
    lastSwapped points to start
    for (j from start to end):
      if (array[j] > array[j + 1]):
        bubble array[j] by swapping up
        lastSwapped points to j
      end if
    end for
    end points to lastSwapped
  end while
end procedure
```

Place	Stable	Adaptive	Best	Average	Worst
In	*Yes*	*Yes*	$O(n)$	$O(n^2)$	$O(n^2)$

sorted order

reverse sorted order

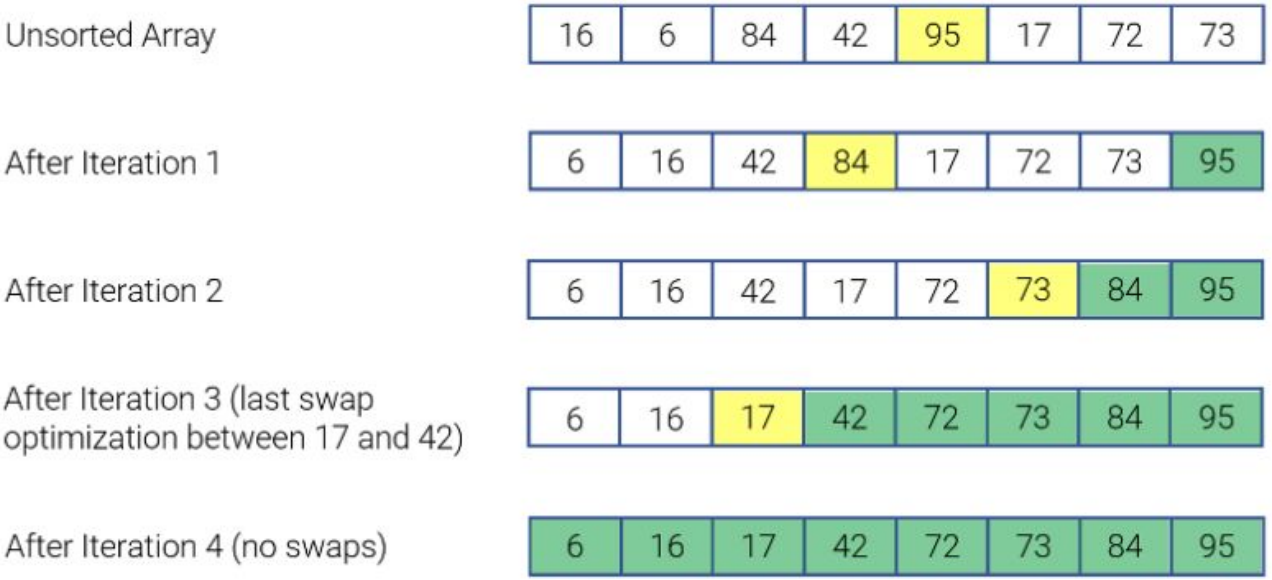
Bubble Sort: Practice

Diagram this array after every iteration of bubble sort with the last swapped optimization, noting which portion of the array is sorted.

Unsorted Array

Bubble Sort: Practice

Diagram this array after every iteration of insertion sort using the minimum element, noting which portion of the array is sorted.



Cocktail Shaker Sort

INTUITION:

One iteration of cocktail shaker sort is a forward & backwards iteration of bubble sort.

Which piece of data is always in its final location after the ith iteration?

The ith largest element & the ith smallest element

PSEUDOCODE:

```
procedure CocktailSort(array):
  end points to last element
  start point to first element
  while (start < end):
    lastSwapped points to start
    for (i from start to end):
      if (array[i] > array[i + 1]):
        bubble array[i] by swapping up
        lastSwapped points to i
      end if
    end for
    end points to lastSwapped
    for (i from end to start):
      if (array[i] < array[i - 1]):
        bubble array[i] by swapping down
        lastSwapped points to i
      end if
    end for
    start points to lastSwapped
  end while
end procedure
```

Cocktail Shaker Sort

OPTIMIZATIONS:

#1: Swaps Made

→ If no swaps were made in one iteration, the array is completely sorted. ***We stop all future comparisons.***

#2: Last Swapped →

MUST USE THIS IMPLEMENTATION

PSEUDOCODE:

```
procedure CocktailSort(array):
  end points to last element
  start point to first element
  while (start < end):
    lastSwapped points to start
    for (i from start to end):
      if (array[i] > array[i + 1]):
        bubble array[i] by swapping up
        lastSwapped points to i
      end if
    end for
    end points to lastSwapped
    for (i from end to start):
      if (array[i] < array[i - 1]):
        bubble array[i] by swapping down
        lastSwapped points to i
      end if
    end for
    start points to lastSwapped
  end while
end procedure
```

Cocktail Shaker Sort: Properties & Efficiency

PSEUDOCODE
(available on csvistool)

```
procedure CocktailSort(array):
  end points to last element
  start point to first element
  while (start < end):
    lastSwapped points to start
    for (i from start to end):
      if (array[i] > array[i + 1]):
        bubble array[i] by swapping up
        lastSwapped points to i
      end if
    end for
    end points to lastSwapped
    for (i from end to start):
      if (array[i] < array[i - 1]):
        bubble array[i] by swapping down
        lastSwapped points to i
      end if
    end for
    start points to lastSwapped
  end while
end procedure
```

Place	Stable	Adaptive	Best	Average	Worst
In	*Yes*	*Yes*	$O(n)$	$O(n^2)$	$O(n^2)$

sorted order

reverse sorted order

Sorting Algorithms: Efficiencies

Sorts	Best	Average	Worst	Place	Stable	Adaptive
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	In	No	No
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$	In	Yes	Yes
Bubble	$O(n)$	$O(n^2)$	$O(n^2)$	In	Yes	Yes
Cocktail	$O(n)$	$O(n^2)$	$O(n^2)$	In	Yes	Yes

Use the Comparator provided in your HW functions to count the # of comparisons you make.

LEETCODE PROBLEMS

147. Insertion Sort List

75. Sort Colors

148. Sort Linked List

Any questions?

Name
Office Hours
Contact

Name
Office Hours
Contact

Let us know if there is anything specific you want out of recitation!