Homework #8 - Sorting

Implement 3+ sorting algorithms of your choice!

In Swift, they should be implemented as an extension to Array (I'll post a video explaining this).

Calculate the average time taken to sort an Array of 1000 randomly sorted Integers, as well as the time taken to sort an Array of 1000 numbers already in order, and 1000 numbers in reverse order.

Fill out the following chart:

Algorithm	Presorted	Random (avg of 100 runs)	Reverse order
Heap Sort	0.0614	0.2 secs (comp super slow)	0.0565
Insertion Sort	0.0041	2.293 secs	3.435
Selection Sort	1.9249 ??	2.1065	1.991
Bubble Sort	0.0112	7.785???	11.304 ????

If you're feeling ambitious, count how many times in each algorithm makes a comparison between numbers in the array, and how many times a number is copied somewhere (swapped/into a temp variable).

Algorithm	Presorted		Random (avg of 100 runs)		Reverse order	
	# of comparisons	# of moves	# of comparisons	# of moves	# of comparisons	# of moves

Written

1) Which of the following tasks would be faster on sorted data

a. Finding the smallest element
b. Calculating the average of a set of values
c. Finding the median value
d. Checking for the existence of a particular element
e. Finding the most common element

Faster on sorted data compared to unsorted: a, c, d, e

2) What sorting algorithm may be bet for the following situations

a. You have 100 computers to split up the sorting on — Merge Sort.
b. You have a set of small integers, unique 1-1000 — Radix Sort
c. You have a set of floats/doubles from 1-1000 — Heap Sort!
d. You have a nearly sorted list — Bubble Sort

3) Which of the algorithms would be easily adaptable to a doubly linked list? Bubble Sort
Insertion Sort is also ok

4) Come up with an example of a 10 element array that is the best case scenario for a bubble sort, and one that's a worst case scenario for a bubble sort.

Best case: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Worst case: reversal of best case