

CS 300 Data Structures
Problem Set #11: Elementary Sorting Algorithms

1. Show the steps required to do a *selection sort* on the following array.

	# of COMPARES	# of SWAPS			
n=5	5	1	12	10	16 11 9 7
					INDEX-OF-LARGEST = 2 LAST = 5
n=4	4	1	12	10	7 11 9 16
					INDEX-OF-LARGEST = 0 LAST = 4
n=3	3	1	9	10	7 11 12 16
					INDEX-OF-LARGEST = 3 LAST = 3
	2	1	9	10	7 11 12 16
					INDEX-OF-LARGEST = 1 LAST = 2
n=2	1	1	9	7	10 11 12 16
					INDEX-OF-LARGEST = 0 LAST = 1
+	15	5	7 9 10 11 12 16		

$$\approx \frac{N \cdot (N+1)}{2} + N$$

$$O(N^2)$$

```

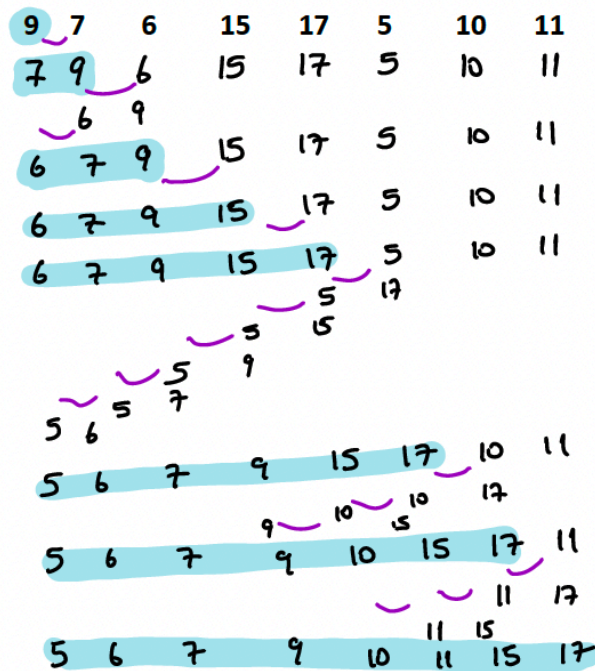
for (int last = n-1; last > 0; last--) {
    int indexOfLargest = findLargest(array, last+1);
    swap(array[last], array[indexOfLargest]);
}
        
```

2. How many compares does selection sort make when the input array is already sorted?

- constant
- logarithmic
- linear
- quadratic**
- exponential

$O(N^2)$

3. Show the steps required to do an *insertion sort* on the following array.



#COMPARES	#SWAPS
1	1
2	2
1	0
1	0
5	4
3	2
3	2

4. Consider the following lists of partially sorted numbers. The double bars represent the sort marker. How many comparisons and swaps are needed to sort the next number [Insertion Sort].

[1 3 4 8 9 || 5 2]
1 3 4 8 5 9 2
1 3 4 5 8 9 2

3 comparisons
2 swaps

5. Consider the following lists of partially sorted numbers. The double bars represent the sort marker. How many comparisons and swaps are needed to sort the next number [InsertionSort].

[1 3 4 5 8 9 || 2]

6 COMPARES
5 SWAPS

6. How many compares does insertion sort make on an input array that is already sorted
- a. constant
 - b. logarithmic
 - ☒ c. linear $O(N)$
 - d. quadratic
 - e. exponential