Quickcheck 07: Sorting mystery

Name:

Consider the following sorting algorithm in pseudocode:

```
1: function sort(A)
2:     for i = 1 to A.length − 1 do
3:         for j = 0 to i − 1 do
4:             if A[j] ≥ A[i] then
5:                 x = A[i]
6:                 Shift every element from j to i − 1 right by one
7:                 A[j] = x
8:                 break
```

(a) Which type of sorting does the above algorithm do (e.g., selection sort, insertion sort, merge sort, quick sort)? What is the worst-case input and worst-case big-$\mathcal{O}$ bound for this algorithm?

(b) Is this sorting algorithm stable? If yes, explain why. If not, what would you change to make it stable?

(c) Explain why this algorithm runs in $\mathcal{O}(n^2)$ time when given an already-sorted array.

(d) How would you change the algorithm so it runs in $\mathcal{O}(n)$ time instead when given an already-sorted array?

Another question

Do you have any questions about this course? It could be about policy, content, instructors, TAs, etc.