1. If you were implementing a sort algorithm for a new language, which sort would you use?
This fundamentally depends on my purpose for using a sorting algorithm. How much do I need speed? Insertion sort is rather simple but it is not that efficient. However if my purpose involves a smaller set, who cares what degree of instantaneous it is? There is the question of time coding vs time saved in execution. If for my use case the time saved by deploying a O(nlogn) algorithm vs a O(n^2) is measured in nano seconds than Insertion sort all the way. However merge sort would probably be my go to for a fast algorithm. It is a bit chunkier on storage space then some of the similar speed algorithms but it is better than quick sort in the worst case. Heapsort is also a good option for a faster sort. Now if I want my users to repent I might deploy bogoSort.

2. Was there a difference in the time it took for bubble and insertion sort to run? Does this make sense given the time complexities for these sorting algorithms?
So for perceived time there was not a difference. However the time clocked I made a table given below. So numerically there is a significant difference in time. Insertion is about 100 times slower than bubble sort. This does not from a numerical stance align with their big O values. Since they should both be O(N^2) on average.

Iteration	Insertion(nanoseconds)	Bubble(nanoseconds)
1	6,085,900	46,000
2	6,011,600	43,100
3	6,414,900	46,400
4	5,320,100	45,200
5	6,042,700	46,800
6	5,977,600	46,300
7	5,667,600	44,000
8	6,002,500	42,800
9	5,938,000	45,600
10	6,078,900	39,400

3. Which sort algorithm has an easier implemenation (in terms of understanding) to you?
For me insertion sort is easiest to understand.