Q1: Is insertion sort similar to bubble sort? Is the number of iterations in bubble sort fixed?

Answer: The idea is different.

For insertion sort, we look at *a single element* at a time and find its correct position in the sorted sublist.

For bubble sort, at each pass, we look at *every consecutive pair of elements* and swap them if they are not in order. In this way, smaller elements move up and larger elements move down. In bubble sort, we don't maintain a sorted sublist. Bubble sort performs multiple passes (iterations) to examine consecutive pairs of elements. It terminates whenever there isn't any swap in a pass. So it is possible that it just runs 1 pass (best case when it is already sorted). In the worst case, it runs n-1 passes.

Q2 (Page 63): Why it is big theta not big O here for complexities?

Answer: Big theta gives you a tight bound, while big O is an upper bound. It is also correct to put big O here. Since it is an upper bound, by looking at $O(n^2)$, you don't know if the actual complexity is of complexity n^2, n, log n, or some others. Putting $\Theta(n^2)$, you precisely know it is of quadratic complexity.

Q3: Why the number of comparison is different at different position?

Answer: The length of your sorted sublist grows by 1 at each iteration. Finding the correct position for an item depends on the length of the sorted sublist. If the item is facing with a short sorted sublist, the cost of finding its correct position is cheap. On the contrary, it will be expensive.

Q4 (Page 56): Why is best case $\Theta(n)$ and not $\Theta(1)$?

Answer: The best case takes (n-1) number of comparisons. So it is $\Theta(n)$. For $\Theta(1)$, it means constant time complexity, i.e., the time doesn't depend on the input size n. In this case, it depends on n and thus is not $\Theta(1)$.

Q5: We can start from 0 if we compare slot[j] with slot[j+1] right? then we run from 0 to size -2.

Answer: Yes, you could have different implementation details, as long as it implements the same design idea.

Q6 (common mistake): can we assume a specific input size (e.g., n=1) or input content (e.g., identical elements) when talking about best case/worst case?

Answer: We cannot. When talking about best case/worst case, we always assume the input size is in general n and the input content (i.e., the key values) is arbitrary. Then we think about specific arrangements of the elements in the input that would give you best case or worst case.