1. Insertion Sort: Worst Case

(a) In the list below, 4 passes of the insertion sort algorithm have been completed, and the double bar separates the sorted part of the list from the unsorted part. The item at index \(i\) is missing. Fill in the missing item with a value that will cause \(\text{insert}(L, i)\) to perform the most number of steps. (As a reminder, this is called the worst case.)

\[
\begin{array}{ccccccc}
\text{sorted} & \text{} & \text{unsorted} & \text{anything < 3} \\
\hline
L & 3 & 4 & 6 & 6 & 2 & 3 & 1 & 5
\end{array}
\]

(b) When \(\text{insert}(L, i)\) is executed on the example list, how many times does the while loop iterate?

\[
4
\]

(c) When \(\text{insert}(L, i)\) is called on the example list, how many assignment statements are executed?

\[
2 \times 4 + 2 \text{ outside loop} = 10
\]

(d) In general, in the worst case, on pass \(i\) of insertion sort, how many times does the while loop iterate? (Your answer should be a formula that involves \(i\).)

\[
i
\]

(e) In general, in the worst case, on pass \(i\) of insertion sort, how many assignment statements are executed? (Again, your answer should be a formula that involves \(i\).)

\[
2i + 2
\]

(f) In terms of \(i\), in the worst case, does function \(\text{insert}\) have constant running time, linear running time, quadratic running time, or some other running time?

(a) constant (b) linear (c) quadratic (d) something else

(g) In function \(\text{insertion_sort}\), the first time that function \(\text{insert}\) is called, \(i\) is 0; the second time, \(i\) is 1; and so on. What value does \(i\) have the last time that function \(\text{insert}\) is called?

\[
\begin{array}{ccccccc}
\text{Comp.} & 1 & 3 & 2+3 & 2+5 & 2(n-2)+1 & 2(n-1)+1 \\
\hline
\hline
\text{i} & 0 & 1 & 2 & 3 & \ldots & n-2 & n-1
\end{array}
\]

(h) For the call \(\text{insertion_sort}(L)\), in the worst case, write a formula expressing how many comparisons are made during all the calls to \(\text{insert}\).

\[
(1 + 2n - 2 + 1) + (3 + 2n - 4 + 1) + \ldots
\]

\[
2n + 2n + \ldots + 2n = 2n \cdot \frac{n}{2} = n^2
\]

(i) In the worst case, does \(\text{insertion_sort}\) have constant running time, linear running time, quadratic running time, or some other running time?

(a) constant (b) linear (c) quadratic (d) something else
2. Insertion Sort: Best Case

(a) In the list below, 4 passes of the insertion sort algorithm have been completed, and the double bar separates the sorted part of the list from the unsorted part. The item at index \( i \) is missing. Fill in the missing item with a value that will cause \( \text{insert}(L, i) \) to perform the fewest number of steps. (That’s called the best case).

\[
\begin{array}{cccccccc}
& & & & & & & \\
& L & 1 & 3 & 3 & 4 & 9 & 8 & 6 & 5 \\
\end{array}
\]

(b) When \( \text{insert}(L, i) \) is executed on the example list, how many times does the while loop iterate?

\[
\Box
\]

(c) When \( \text{insert}(L, i) \) is called on the example list, how many assignment statements are executed?

\[
2
\]

(d) In general, in the best case, on pass \( i \) of insertion sort, how many times does the while loop iterate?

\[
\Box
\]

(e) In general, in the best case, on pass \( i \) of insertion sort, how many assignment statements are executed?

\[
2
\]

(f) In the best case, does insert have constant running time, linear running time, quadratic running time, or some other running time?

(a) constant  (b) linear  (c) quadratic  (d) something else

(g) For the best case, write a formula expressing how many comparisons are made during all the calls to insert.

\[
\begin{array}{cccccccc}
& \text{comp} & 1 & 2 & 2 & \cdots & \text{n-1} \\
& i & 0 & 1 & 2 & \cdots & \text{n-1} \\
2 & \leftarrow & \text{addup} & (n-1)+2 & +1 \\
\Rightarrow & 2n-1 \\
\end{array}
\]

(h) In the best case, does insertion_sort have constant running time, linear running time, quadratic running time, or some other running time?

(a) constant  (b) linear  (c) quadratic  (d) something else