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**.)

\[ \overline{[6, 5, 4, 3, 2, i]} \]

(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?

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

(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 $i$.)

Since we only leave loop b/c 1st condition becomes false, same as: how many times do we need to do $i := i-1$ to get $i := 0$?

\[ i \]

(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 $i$.)

\[ 2i + 2 \left( \text{outside loop} \right) \]

(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?

\[ \text{len}(L) - i \]

(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 $\text{insert}$.

\[ a_n + 2a_{n-1} + \ldots + 2a_0 \]

\[ a_{n/2} \]

\[ 2n + 2n + \ldots + 2n \]

\[ \Rightarrow 2n \cdot \frac{n}{2} \Rightarrow n^2 \]

(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).

L 1 3 3 4 9 8 6 5

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

0

(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?

0

(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 $\text{insert}$ 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) For the best case, write a formula expressing how many comparisons are made during all the calls to $\text{insert}$.

$n = \ln(n)$

$0, \ldots, n-1$

Each call (except $i = 0$) makes 2 comparisons

$2 \cdot (n-1) + 1 = 2n-1$

(h) In the best 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