DestTest Analysis

1.) Worst Case Derivation:

a. My algorithm splits the ladder into $\sqrt{n}$ size gaps to traverse. The worst case for my algorithm would be if the safe rung was the n-1 rung, for an n-rung ladder.

b. My algorithm uses two loops, one to traverse the gaps until a break, and one to go linearly from the previous gap the break upwards to search for the specific rung.

   i. In the worst case, since the first loop would have to iterate the maximum number of gaps, and the second loop would have to iterate the maximum gap length. (+ some constant for nonperfect square ladder lengths.)

c. Worst Case Derivation: To find the worst case, we must add up and determine the rung where the maximum number of drops (iterations) that will be performed for each loop, then sum them. The total will follow the form of: (Num of Drops Loop 1) + (Num of Drops Loop 2) + C, where C is constant accounting for non-perfect square ladder lengths.

   Note: $\sqrt{n}$ has been floored to avoid fractional rungs, which are impossible.

   i. First Loop Maximum Drops:

      1. At worst case, the first loop will perform approximately $\sqrt{n} + \mathrm{c}$ drops, since the gap size $\sqrt{n}$, and to reach the n-1 rung, we must reach the nth to break the first device. Constant **c** is to account for non-perfect squares, where at reaching $\sqrt{n}$ iterations, there may be additional rungs remaining.

Drop Iteration	Height
1	$\sqrt{n}$
2	$2\sqrt{n}$
…	…
$\sqrt{n} + \mathrm{c}$	$\sqrt{n} * \sqrt{n} = n$

   ii. Second Loop Maximum Drops:

      1. Once the gap where the safe rung exists has been determined, (for worst case the $(\sqrt{n} - 1)\sqrt{n}$ rung), we begin linearly searching that gap until the second device breaks (nth rung worst case). The search will require $\sqrt{n} + \mathrm{d}$ drops (iterations). Constant **d** accounts for non-perfect square ladders lengths, where, upon reaching the $\sqrt{n}$ gap, there are **d** remaining rungs to reach the nth rung.

Drop Iteration	Height
1	$(\sqrt{n} - 1)\sqrt{n} + 1$
2	$(\sqrt{n} - 1)\sqrt{n} + 2$
…	…
$\sqrt{n} + d$	$(\sqrt{n} - 1)\sqrt{n} + \sqrt{n} + \mathrm{d} = n$

   iii. Summation: Thus, total number of drops for the worst case is

$$(\sqrt{n} + \mathrm{c}) + (\sqrt{n} + \mathrm{d}) = 2\sqrt{n} + \mathrm{C}$$

   where C = c + d. Thus, the time complexity of the worst case is $\theta(\sqrt{n})$.

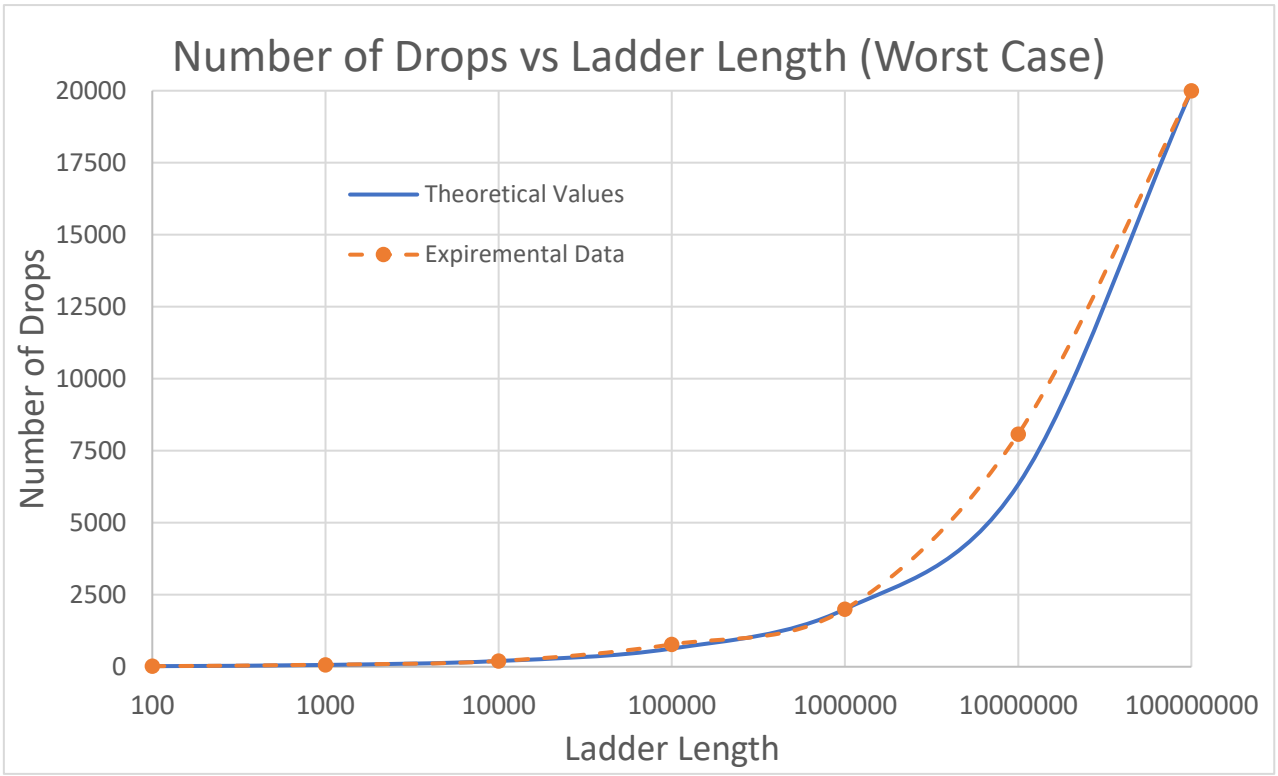
2.) Data Tables

Provided Case Table	Safe Rung Position		
Ladder Size (n)	$n-3$	$\frac{n}{2}-2$	2
100	97	48	2
1,000	997	498	2
10,000	9,997	4,998	2
100,000	99,997	49,998	2
100,000,000	99,999,997	49,999,998	2
	Drops Required		
100	18	14	4
1,000	69	20	4
10,000	198	149	4
100,000	774	230	4
100,000,000	19,998	14,999	4

Worst Case Table	Safe Rung Position	
Ladder Size (n)	$n-1$	Drops Required
100	99	20
1,000	999	71
10,000	9,999	200
100,000	99,999	776
1,000,000	999,999	2000
10,000,000	9,999,999	
100,000,000	99,999,999	20,000

3.) Table interpretations

a.



b. To flush out the graphs, I added a few more data points for worst case. As you can clearly see they agree with one another, with the distance between then credited to the constants mentioned earlier.