Chanelle Mosuqera
1 Destructive Testing
CSC-349-05

Destructive Testing

1. A derivation of f(n) where f(n) is the worst case number of drops needed by your program to find the highest safe rung on a ladder of n rungs. Namely:
    a. The definition of the worst case or cases for your algorithm. This may vary depending on the inputs ladder size and highest safe rung.

        For my algorithm, the worst case is $\Theta(\frac{n}{\sqrt{n}})$

    b. Clearly explain why this is the worst case

        The algorithm starts out by increasing in rung by $\sqrt{n}$ rounded to the nearest integer. Once a device breaks, we increment by 1 rung starting at the safest known rung. So, we know that the ladder consists of $\sqrt{n}$ rungs per gap and $\frac{n}{\sqrt{n}}$ gaps. At worst case, the safest rung will be within the last gap, which would take $\frac{n}{\sqrt{n}}$ drops, and be the last rung in that gap, which would take $\sqrt{n}$ drops.

    c. Show how you determined f(n) for these cases. That is, you must justify that f(n) represents the number of drops that will be simulated by your program.

        Following the calculations and explanation above (1.b), recall that I divide my ladder by $\sqrt{n}$ gaps, each gap containing $\sqrt{n}$ rungs. With my first device, I increment by gaps, so it'll take $\sqrt{n}$ drops until I reach the highest gap where my device breaks. So I know that the safest rung must be in the gap between the last known safest rung, and the rung where the device broke. Since there are $\sqrt{n}$ rungs within a gap, I increment 1 rung at a time until my device breaks again. Once I reach $\sqrt{n}$ drops, my second device breaks, which determines that the safest rung is the (n-1)th rung. In total, I've done 2*$\sqrt{n}$ drops.

        Therefore, in the worst case, the number of drops is 2*$\sqrt{n}$, and the safest rung is n-1.

2. A table that shows empirical results of the number of drops required for different ladder sizes and different highest safe rungs. The highest safe rung is specified using either as a function of the ladder size or some fixed number -see below. The table rows are labeled by the ladder sizes 100, 10,000, 1,000,000, 100,000,000. The table columns are labeled by how the input highest safe rung is calculated. Namely

a. Column 1 label: ladder size - 3. (ladder size=1,000highest safe rung = 997)
b. Column 2 label: ladder size/2 - 2. (ladder size=1,000highest safe rung = 498)
c. Column 3 label: highestSafeRung = 2. (ladder size=1,000highest safe rung = 2)

	=Size - 3	=Size/2 - 2	= 2
100	Safe rung = 97 # of drops = 18	Safe rung = 48 # of drops = 14	# of drops = 4
10,000	Safe rung = 9997 # of drops = 198	Safe rung = 4998 # of drops = 149	# of drops = 4
1,000,000	Safe rung = 999997 # of drops = 1998	Safe rung = 499998 # of drops = 1499	# of drops = 4
100,000,000	Safe rung = 9997 # of drops = 19998	Safe rung = 49999998 # of drops = 14999	# of drops = 4

3. Finally, interpret the results in the table.
   a. Do they agree with your theoretical analysis? Do this by comparing the number of drops calculated by your analysis in 1) with the actual number of drops performed by your simulation.
   b. If they do not agree, give your explanation.

   The results of my simulation agree with my theoretical analysis. As I've mentioned, the worst case # of drops is 2*$\sqrt{n}$. For example, when n = 100 and the safest rung = n-1, it would take 2*$\sqrt{100}$ = 20 drops. If the safest rung were n-1-2 (or, 100 – 1 - 2 = 97, such as the example in the table above), then the # of drops would be 20 – 2 = 18, which is what I got when running the simulation.