Matt Wisner

Professor Siska

Algorithm Engineering

November 16, 2019

Asgwilanga Caverns Big-O Analysis

Big-O runtime analysis of Algorithms is used to measure the time complexity of algorithms to give a sense of the worst-case scenario for an algorithm. Big-O notation provides us with the efficiency analysis framework to find the order of growth for an algorithm’s basic operation count. There are different time complexities that an algorithm can have. O(logn), O(n), O(nlogn), O(n!) and many more. Comparison of algorithms is made by comparison of Big-O values and n (which is the input size). For our project, Asgwilanga Caverns, we determined that the time complexity is O(n^2). We came to this conclusion by tracing each step of the algorithm. First, the algorithm finds all reachable caves using the sum rule, making sure that the three id parts add up to 16, and checking if any of the three values are min or max(the maximum values being 16,8,7 . This is how we get our 30 caves. This portion of the algorithm runs in n time, n being the 16\*8\*7 = 896 loops in total. The algorithm then iterates through each of the caves from the cave that it is in as it moves. This means that the function is n^2 for the movement portion of the algorithm. Therefore the function that we found was n^2 + n. By simplifying using the rules of Big-O notation, we know that the n runtime portion of the function can be ignored because n^2 is much larger. The result is a Big-O runtime on n^2.