Data Structures and Algorithms - Homework 2

HK Rho

February 7, 2020

Problem 2a

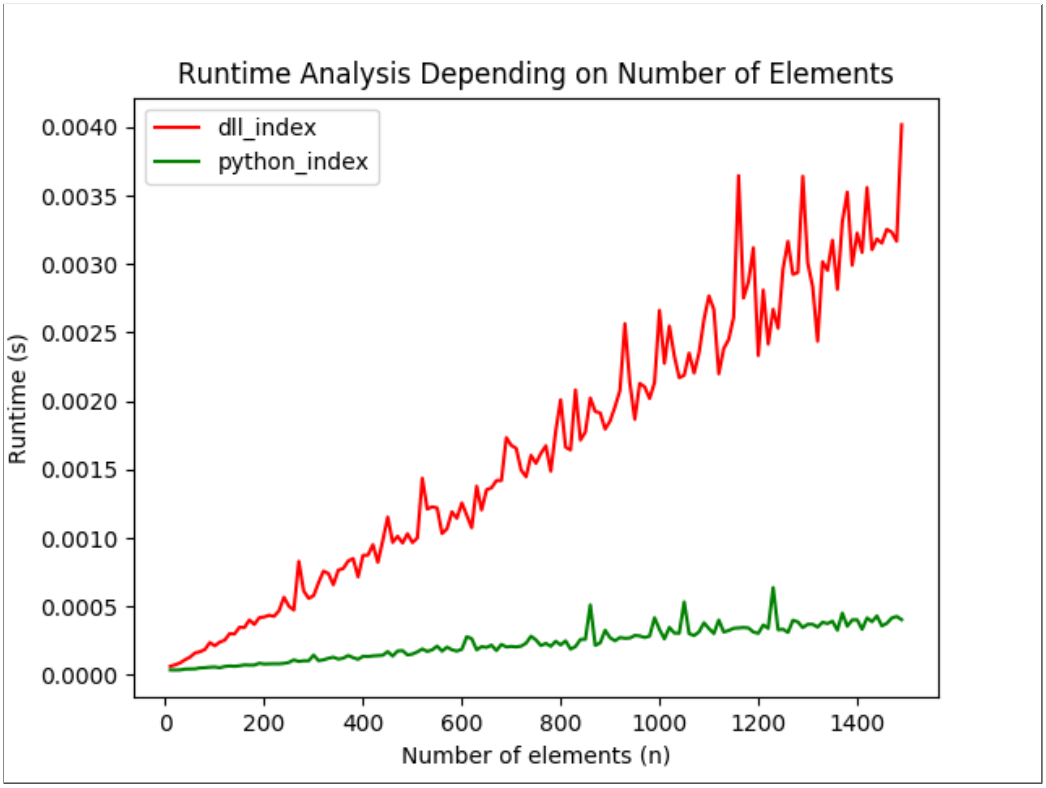

Figure 1: Run-time analysis with the starting n value = 10, ending n value = 1500, and step size = 10. This was not observed from starting n value = 10 to ending n value = 10,000 for the execution of the program taking too long.

From what I observe in Figure 1, it seems like the big O analysis for indexing a doubly linked list is $O(n)$ whereas the big O analysis for indexing a python list seems to be $O(1)$. Looking at the general pattern, the run-time for indexing a doubly linked list steadily increases in a linear fashion when given n elements. On the contrary, the run-time for indexing a python list seems to maintain a constant fashion as it is more horizontal when compared to the run-time graph for indexing doubly linked lists.

Problem 2b

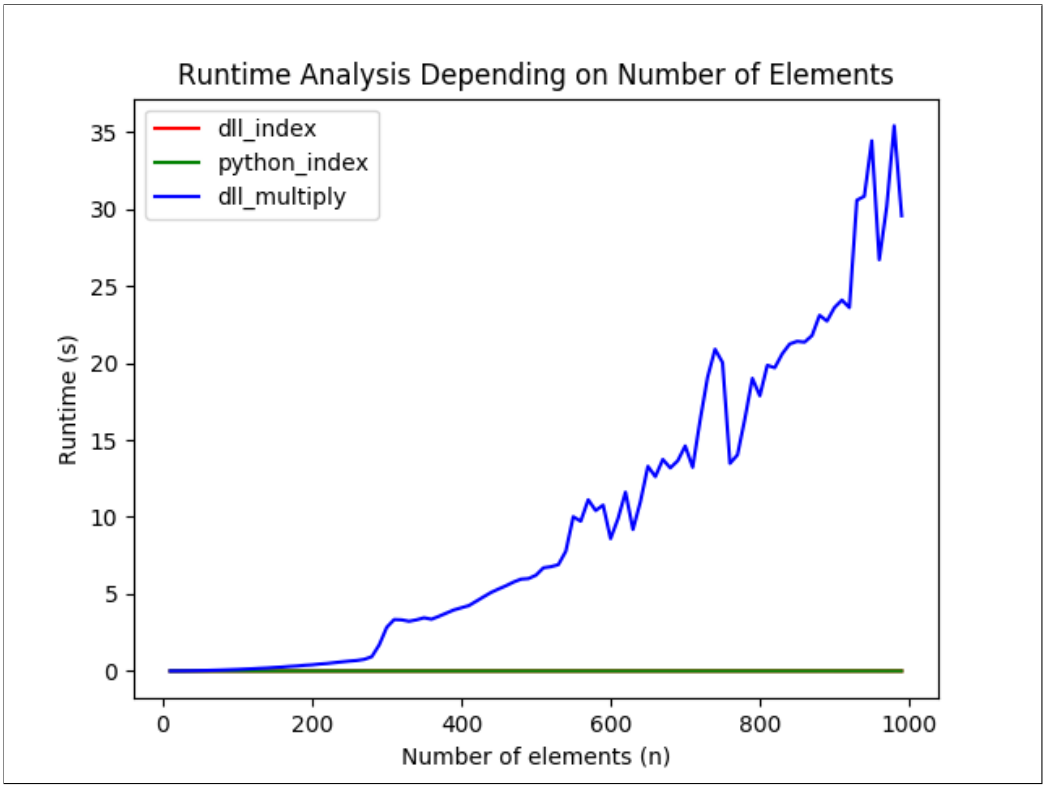

Figure 2: Run-time analysis with the starting n value = 10, ending n value = 1000, and step size = 10. This was not observed from starting n value = 10 to ending n value = 10,000 for the execution of the program taking too long.

From what I observe in Figure 2, it seems like the big O analysis for multiplying all pairs of a doubly linked list is $O(n^2)$. The graph of its run-time displays a quadratic behavior as the number of elements are increased. Because the graph of *'dll multiply'* is dominant in comparison to the other two graphs, it has the effect of making the other two graphs to look as they have a constant run-time overall.