EE360C: Algorithms
The University of Texas at Austin — Quiz #3
Dr. Christine Julien, Dr. Vallath Nandakumar — February 8, 2018 (*in class*)

Name: **EID:**

Quiz #3

Complete this problem individually. Use only this paper for your answer. **Clearly note somewhere on the front if you use the back of this sheet for your answer.**

Problem 1: Time Complexity

There exists a very popular sorting algorithm called Timsort, the default sorting algorithm in both Python and Java. This sort is a combination of two different sorting algorithms: Merge sort, and Insertion sort. Insertion sort is used for the smaller arrays, and Merge sort is used for the larger arrays. Recall that Merge sort is $O(nlogn)$ and Insertion sort is $O(n^2)$. What advantage would Timsort have to combine the two algorithms if Merge sort has a better Big-O metric?

Solution

The best case time complexity for Insertion sort is $O(n)$, where as the best case for Merge sort is $O(nlogn)$. You can also argue (since we don't expect you to know the best case off the top of your head) that the coefficients of both Big-Os cause smaller arrays to be a better choice in Insertion sort.

Problem 2: Time Complexity Continued

Consider two algorithms: $f(n)$ and $g(n)$. You run both algorithms with an input n=10,000. You find that $f(n)$ takes 10 ms while $g(n)$ takes 1 min to run. Which of these has a better (i.e. smaller) Big-O metric?

Solution

You can't definitively tell without more data points or without seeing the code itself.