335 — Algorithms — Sort Race

Project #3 – Sort Race

Group: Bungalow

Members: Thomas Smith (thqmas@csu.fullerton.edu), Vincent DeAugustine (vinnied44@csu.fullerton.edu), Matthew Quinlan (mdquinlan@csu.fullerton.edu)

Merge Sort:

The algorithm for merge sort is recursive. The algorithm works by splitting the array we want to sort into two sub arrays. Those sub arrays are divided into their own sub arrays, and so on until the sub arrays can all no longer be divided. The sub arrays are then merged back together in sorted order. This process can be represented by the recurrence relation T(N) = 2T(N/2) + O(N). The 2T(N/2) complexity comes from recursively dividing each array in half. The O(N) is the complexity of merging each of the sub arrays. Using the Master method, we get a time complexity of O(NLogN) for merge sort.

Quick Sort:

The algorithm for quick sort is recursive. It works by choosing a “pivot” and putting all elements smaller than that pivot to the left and all larger than it to the right. O(N). Once this “pivot” is in place, the arrays next to the “pivot” repeat this, choosing a new “pivot” and sorting the sub arraysO(log(n)). This repeats until the the entire array is sorted. This means the average case of the algorithm O(N(log(N)).

Gold’s Poresort:

The algorithm for Gold’s Poresort is non-recursive. It works by checking all even numbered locations in the array and checking with each of their neighbors to the right and swapping if it is less than its neighbor. (O(N)) It then checks all odd numbered locations and swaps with its neighbor to the right if its less than it. (O(N)). These two operations together equal O(N+N) or O(2N) or simply O(N) since constants go to 1. You then repeat these 2 operations until the array is sorted O(N). Since you are doing an N number of operations and N number of times, the big O notation for Gold’s poresort (average case) is O(N^2).