Sorting Olympics Running Time

Jared Dyreson
California State University, Fullerton

Mason Godfrey, California State University, Fullerton
Brian Lucero, California State University, Fullerton

2020
November

Contents

1 Big-O Analysis

1.1 Initial Setup

This program starts off by randomly selecting an input from a list that was provided in the project description. The input is then given to four separate instances of a specific sorting algorithm, which will process it. There is then a main loop in which all the algorithms are allowed to do a pass and will signal if completed.

1.1.1 Pore's Gold Sort

This algorithm is similar to Bubble sort as is follows the same behavior where the biggest element "bubbles" to the end. Pore's key distinction however, is that even and odd pairs are swapped if they satisfy the swapping condition. Thus the Big-O running time is $O(n^2)$.

1.1.2 Insertion Sort

The basis of insertion is that once it encounters a swap candidate, it has to back track from right to left until it finds the final resting place of that element. That being said, the most amount of movements until the swap would be the length of the container. Here we can say that the Big-O running time is $O(n^2)$.

1.1.3 Merge Sort

Merge works by breaking up the container into multiple sub-containers and then sorting them recursively. At the end, each of the sub-containers are stitched together, making one complete container. This style of approach is called divide and conquer, and generally runs in $O(n * log(n))$.

1.1.4 Quick Sort

Lastly, quick sort is also in the same family of algorithms as merge sort. A defining characteristic of this algorithm is it's use of pivot values and moving values less than itself to the left. This creates inherit order and structure to each pass it does. It too creates sub-containers but only through the use of ranges and does not allocate any extra memory for this. Thus the running time for quick sort is $O(n * log(n))$.