Name: Anisha Bhattacharya

Godbless Chille

NETID: abhatta9

gchille

**LAB8 REPORT.**

Assumption made:

To find the estimated run-time, we used the largest data values to find the value of the constant k in the big o equation. Thereafter, we used this k multiplied by the size to find the theoretical run time for the respective sorting algorithm.

For example. For insertion sort, our largest array had size = 1000. We divided the actual run-time, which is 5ms by 1000 to get a constant k. Then we multiplied this value by the smaller data sizes, which are 800, 400, and 100.

Here are the graphs for the run times of the sorting algorithms.

Key. (for all the following graphs)

x- input size

y- the **average** run time given in milliseconds(ms)

In the big-O notation;

n- input size

**Bubble sort -O(n^2)**

A close up of a window

Description automatically generated

The theoretical run times were generally lower than the actual run-times. For example, for the array of size 400, the actual run-time was 8.0ms but the theoretical run-time was 5.48ms.

**Insertion Sort -O(n^2)**

A close up of a window

Description automatically generated

The theoretical run-time was generally lower than the actual run-time. For example, for the array of size 800, the actual run-time was 4.67ms whilst the theoretical run-time was 3.2ms.

**Shell Sort -O(n^2)**

This is because this algorithm greatly uses insertion sort.

A close up of a window

Description automatically generated

The theoretical run-time was generally lower than the actual run-time. For example, for the array of size 400, the actual run-time was 3.0ms whilst the theoretical run-time was 2.0ms.

**QuickSort -O(nlogn)**

*Assumption*: based on the fact that the median element can be found in linear time. So we find the median first, then partition the array around the median element. Through this, the worst-case complexity of nlogn can be achieved.

A picture containing indoor

Description automatically generated

The theoretical run-time was generally lower or equal to the actual run-time. For example, for the array of size 800, the actual run-time was 1.0ms whilst the theoretical run-time was 0.0ms.

However, for data sizes of 400 and 100, the actual run-time was equal to the theoretical run-time, which was 0.0ms.

**LibrarySort – O(nlogn)**

Using the assumption that this algorithm can be operated exactly like quicksort.

A close up of a window

Description automatically generated

Conjecture.

In comparison to the other sorting algorithms, this algorithm’s run-time was very similar to that of the quicksort algorithm. For data sizes (100, 400, 800, 1000), the run-time was always equal or less than that of the quicksort algorithm. Therefore, the library sort algorithm has a *quicksort* nature, complexity of O(nlogn)