**PROJECT 1**

**SORTING ALGORITHMS**

**NAME: LUIS LEDEZMA**

|  |  |
| --- | --- |
| **Sorting Algorithm** | **Real Running Time (Seconds)** |
| \*Counting Sort | 1.46 |
| \*Insertion Sort | 41,042.35 |
| \*Selection Sort | 43,381.73 |
| \*Quick Sort | 32.44 |
| \*Merge Sort | 42.91 |
| \*Heap Sort | 69.19 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sorting Algorithm** | **Worst** | **Average** | **Best** |
| \*Counting Sort | n+k | n+k | n+k |
| \*Insertion Sort | n2 | n2 | n |
| \*Selection Sort | n2 | n2 | n2 |
| \*Quick Sort | n2 | nlog(n) | nlog(n) |
| \*Merge Sort | nlog(n) | nlog(n) | nlog(n) |
| \*Heap Sort | nlog(n) | nlog(n) | nlog(n) |

\***Computer specs**:

\***Model:** MacBook Pro 2013

\***Processor**: 2.4 GHz 6-Core Intel Core i5

\***Memory**: 4 GB 1600 MHz DDR3

**\*COUNTING SORT:**

Data structure used for this algorithm was an array. For the counting sort algorithm method is a short one with about ten lines of code. It takes two inputs; an array and the largest value in the array. Creates an array of side based on the largest digit. First loop has the number of times each number is in the array. The second loop inserts the values into a sorted array.

**\*INSERTION SORT:**

Data structure used for this algorithm was an array. The insertion sort algorithm method is a short one with about seven lines of code. It takes one input, the array to be sorted. A loop that traverses through 1 to the length of the array. The move elements of the array, that are greater than the key, to one position ahead of their current position.

**\*SELECTION SORT:**

Data structure used for this algorithm was an array. The counting sort algorithm method is a short one with about six lines of code. It takes the array to be sorted as input. Two nested loops, a linear search that compares each value with all the values. After, it finds the smallest value it swaps it with the first index value. After, the same cycle repeats with the second index value in the array being swapped with the next smallest value and so on.

**\*QUICK SORT:**

Data structure used for this algorithm was an array. The quick sort is a recurrence algorithm with two methods called: quicksort and partition that take three parameters; array to be sorted and first index and last index in the array. Quicksort methods divides the array and partition sorts the numbers in the partitions and places them in the right place in the original algorithm.

**\*MERGE SORT:**

Data structure used for this algorithm was an array. The merge sort algorithm is another recurrence algorithm with one method: mergeSort. The mergeSort method takes one parameter; the array to be sorted. It breaks the array into subarrays of size 1 and then it merges the subarrays while also sorting them until we have the final sorted array.

**\*HEAP SORT:**

Data structure used for this algorithm was an array. The heap algorithm has two parameters called: heapsort and heapify. The algorithm is simple to understand if you know the basics of a heap. The array to be sorted is taken as input. After, call on heapify to create a heap out of the array. Then, after the heap is formed, we start forming a sorted array. The root of the heap will contain the largest value, so it swaps the last index in the heap with the value in the root and call on heapify to maintain the heap structure. The cycle is repeated again and again until we have sorted array.