Project 1

Nicolas Nguyen

Algorithm Descriptions

- Counting Sort: Count the occurrences of each number in the array, build a new array sorted based on this count

- Insertion Sort: For each element in the array, while the element is less than its preceding element, swap. After n iterations, the first n elements are sorted.

- Selection Sort: Each iteration, search the unsorted indices for the minimum element. Move the minimum element to the end of the sorted section.

- Quick Sort: Recursively partition the array into subarrays consisting of elements less than a chosen element (“pivot”) and elements greater than the pivot, respectively. Once this is done, put the pivot in its correct position. If this is done recursively on each partitioned array, the final list will be sorted.

- Merge Sort: Recursively split the array in half, and sort the halves as you merge them back into one. Base case = array length 1.

- Heap Sort: Build a max heap from the original array. For each element in the heap, remove the max element, place it at the end of the array, decrement the heap size, and reform the heap.

Experiment Info

- The experiment was run on a 2019 Macbook Pro with a 1.4GHz Quad Core i5
- The code does not produce console output, so there is no output to share beyond the sorted number files
- All sorting algorithms were implemented using python lists including heap sort, which used a list based max heap.
- Unfortunately, the $O(n^2)$ algorithms ran for 48+ hours and did not terminate within the deadline for this project. The programs contained intermittent logs to the console to show that it was indeed running (not hanging in an infinite loop). The source code is available, so feel free to inspect the algorithms for issues that would cause them to run for so long. I will be reporting these times as >48hrs (48 hrs * 60 min * 60 sec = 172800 seconds).

Time Complexities

Algorithm	Asymptotic Bound
Counting Sort	O(n)
Insertion Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Quick Sort	O(nlogn)
Merge Sort	O(nlogn)
Heap Sort	O(nlogn)

Experiment Results

Algorithm	Elapsed Sorting Time (s)
Counting Sort	4.485639810562134
Insertion Sort	> 172800
Selection Sort	> 172800
Quick Sort	18.43890404701233
Merge Sort	27.121155261993408
Heap Sort	51.22308802604675