Prabhas Kumra

Assignment #3

CS 302 – 1004

Description of Machine: I ran my test script on Dell OptiPlex-7020 3.1 GHz Intel Core i5 Processor and 128 GB SSD. It is running OS Ubuntu 18.04.

* According to the time executions, the fastest sort was Counting Sort. After that Quick Sort, then Insertion Sort and the slowest was Bubble Sort.
* In this assignment, recursion was implemented in a good way much better than it was used in Fibonacci series. Quick Sort recursively sort the arrays into sub arrays, with base case of length of 0 or 1 and then compares the numbers and sort them accordingly.
* Main verifies sorting algorithms by checking if all the numbers are in order or not, after the sort is done. There is a loop in the main file which checks it and if the numbers are out of order then it output an error message. It was just a one **for** loop which just compares two items, so it ran n times.
* Bubble Sort
  + Bubble Sort is a comparison sort, which simply just compares each pair of adjacent items and swaps them if they are in wrong order.
  + Worst case is . Best Case
  + It is stable as it keeps the swapped item in order if the items were same. It is also adaptive which means it exits if the list is already sorted. It only requires additional memory space.
  + Swapped flag checks if the list is already sorted then it exits the loop immediately as it did not make any swap during the pass.
* Insertion Sort
  + Insertion Sort is a comparison sort which takes the current element and put it in its appropriate position in the list.
  + Worst Case is . Best Case
  + It is stable does not change the relative order of the items. It is also adaptive. It only requires additional memory space.
* Counting Sort
  + Counting sort is a non-comparison algorithm. It works by counting the number of objects that have distinct key value and then increase the counter by 1 every time it counts an object.
  + It is not adaptive. It is also not stable.
  + Time Complexity , where is the number of elements. Space Complexity
  + Counting sort has a limitation that it only works for **int** data type and cannot work for **float** data type. It is also very space consuming.
* Quick Sort
  + Quick Sort is a comparison sort; first it picks an element called pivot from the array. Then it does partitioning arranging the numbers less than pivot and greater than pivot. Then it recursively does sorts the arrays into sub-arrays and does partitioning.
  + It is not stable and not adaptive.
  + Worst Case . Best Case . It takes space.
* Quick Sort (modified)
  + Quick Sort ran for 3.722 seconds
  + On a presorted array and medianof3 removed. With the lowest element chosen as pivot there was just one array instead of 2-sub arrays and it was solving recursively. Which made Quick Sort ran in its worst case.
  + Bubble Sort ran for 0.017 seconds after the modification
  + Bubble sort is adaptive thus which made it recognize that the array was already sorted and it exits the process.