**Algorithm Coursework 2**

**PART 1**

**(i)** The data used for the sorting were randomly generated integers. Decimal numbers were not used as because for a larger size, comparing the values of decimals can cause a significant time cost. This is for the fact that decimal numbers with memory allocations of type double takes 64 bits of storage where as integers with memory allocations of type int takes 32 bits of storage.

For a size of 100K, the maximum value of the list of values is 100K and the least value is 0. Similarly, for a larger size of 10 Million, the maximum number to be sorted would be 10M and the least number be 0. Each of these sizes are stored in an array and arrays of sizes 100K, 250K, 500K,750K, 1M, 2.5M, 5M, 7.5M and 10M were made. All the nine arrays were initially made and before each run all the arrays were shuffled. So for all the 10 runs they were shuffled 10 times and thus its puts into perspective the fact that all the numbers being sorted are the same for each array, that is, the numbers did not change which acts as our control for this experiment, only the order of the numbers changed which is what being experimented upon. So, all the tests would indicate to the performance depending on the different order of numbers being sorted each time. The ordering is extremely important as because if the order was already sorted or an order where the majority was already sorted, implementing quicksort on that data would have taken O(n2) time complexity (which is quicksort’s worst case) as it would have to swap the places of each number n times of an array of length n. However, this would not impact heapsort as its time complexity is always O(nlogn).

**(ii) For the ten runs:** The average of comparisons + exchanges along with their standard deviations.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Heapsort |  |  |  | Quicksort |  |  |  |
| RUN | Size | Compares and Exchanges | Standard  Deviation | RUN | Size | Compares and Exchanges | Standard  Deviation |
|  | **100000** | 3737393 | 546.3580531 |  | **100000** | 2172202 | 4989110.392 |
|  | **250000** | 10169645 | 1079.488644 |  | **250000** | 5858835.4 | 12242758.81 |
|  | **500000** | 21583782 | 861.5456717 |  | **500000** | 12438704.2 | 27649176.58 |
|  | **750000** | 33481169 | 2414.081597 |  | **750000** | 19338520 | 41182970.22 |
|  | **1000000** | 45666447 | 2234.16866 |  | **1000000** | 26913105 | 54125738.52 |
|  | **2500000** | 58086032 | 2020.020047 |  | **2500000** | 33560739.4 | 67921903.86 |
|  | **5000000** | 257347223 | 4074.01828 |  | **5000000** | 149641449.4 | 296475340.3 |
|  | **7500000** | 397025982 | 5229.585887 |  | **7500000** | 234780951.4 | 451494415.5 |
|  | **10000000** | 539694787 | 4173.921705 |  | **10000000** | 315276799.8 | 611823191.3 |

From the table it can be observed that for Heapsort the standard deviation for Comparisons + Exchanges is quite small compared to the standard deviation of the Comparison + Exchanges for Quicksort.

The average of times taken for sorting along with their standard deviations.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Heapsort |  |  |  | Quicksort |  |  |  |
| RUN | Size | Time | Stand. Dev | RUN | Size | Time | Stand. Dev |
|  | **100000** | 13475961 | 2658111.772 |  | **100000** | 10596038 | 3880168.743 |
|  | **250000** | 44557467 | 2064838.884 |  | **250000** | 26173927 | 1551013.387 |
|  | **500000** | 82546634 | 10804570.42 |  | **500000** | 58835325.6 | 3180819.057 |
|  | **750000** | 169539690 | 9863632.672 |  | **750000** | 87897432.2 | 6039256.822 |
|  | **1000000** | 197860107 | 7222167.139 |  | **1000000** | 115993059.4 | 6000376.608 |
|  | **2500000** | 274368217 | 39732275.83 |  | **2500000** | 145916243.6 | 6195561.434 |
|  | **5000000** | 1451018233 | 96736405.99 |  | **5000000** | 635865667.9 | 23756545.57 |
|  | **7500000** | 2397486536 | 21659934.82 |  | **7500000** | 970369240.8 | 16374684.03 |
|  | **10000000** | 3383363463 | 38964042.6 |  | **10000000** | 1314040782 | 22251734.8 |

**(iii)** The experiment to compare Quicksort and Heapsort has produced these two graphs which is the **average of 10 runs** for a size list of arrays being sorted in the range of 100,000 to 10,000,000 (**100K to 10M**). The other 20 graphs that was required for these two graphs are given in **Appendix A**.

**Appendix A**

1)Compares and Exchanges:

2)Time VS Size:

**ALL TIMES ARE IN NANOSECONDS**