LAB 3

DATA STRUCTURES

TANMAY KEJRIWAL

1.When Each Method is Run on Unsorted List of Different Lengths

* + - **Length** of List = **1**
      * **QuickSort** takes **1.92 x 10-5 Seconds**
      * **MergeSort** takes **5.8 x 10-6 Seconds**
      * **HeapSort** takes **1.47 x 10-5 Seconds**
    - **Length** of List = 25 ( Identical Number)
      * **QuickSort** takes **3.56 x 10-5 Seconds**
      * **MergeSort** takes **2.90 x 10-5 Seconds**
      * **HeapSort** takes **2.68 x 10-5 Seconds**
    - **Length** of List = 5000000 ( Identical Number)
      * **QuickSort** takes **2.16 Seconds**
      * **MergeSort** takes **3.58 Seconds**
      * **HeapSort** takes **0.34 Seconds**
    - **Length** of List = **20** 
      * **QuickSort** takes **4.40 x 10-5 Seconds**
      * **MergeSort** takes **3.62 x 10-5 Seconds**
      * **HeapSort** takes **8.44 x 10-5 Seconds**
    - **Length** of List = **100**
      * **QuickSort** takes **3.15 x 10-4 Seconds**
      * **MergeSort** takes **2.56 x 10-4 Seconds**
      * **HeapSort** takes **5.21 x 10-4 Seconds**
    - **Length** of List = **500**
      * **QuickSort** takes **3.1 x 10-3 Seconds**
      * **MergeSort** takes **4.0 x 10-3 Seconds**
      * **HeapSort** takes **4.11 x 10-3 Seconds**
    - **Length** of List = **50000**
      * **QuickSort** takes **8.35 x 10-2 Seconds**
      * **MergeSort** takes **1.2 x 10-1 Seconds**
      * **HeapSort** takes **1.02 x 10-1 Seconds**
    - **Length** of List = **5000000**
      * **QuickSort** takes **4.79 Seconds**
      * **MergeSort** takes **14.57 Seconds**
      * **HeapSort** takes **28.12 Seconds**

**OBSERVATION -**  When the length of the list MergeSort is comparably faster than the other two. When these algorithms are used to sort a small list with identical numbers they almost do it in the same time but when the list (Identical Numbers) size increases Heap Sort is almost 10 times faster than MergeSort and QuickSort . When an Unsorted List of small is used for testing QuickSort is the fastest , it becomes more evident once list of big sizes are used.

2. When Each Method is run on a sorted list of different lengths.

* + - **Length** of Sorted List = **20** 
      * **QuickSort** takes **3.76 x 10-4 Seconds**
      * **MergeSort** takes **4.86 x 10-5 Seconds**
      * **HeapSort** takes **4.23 x 10-5 Seconds**
    - **Length** of Sorted List = **100**
      * **QuickSort** takes **1.2 x 10-3 Seconds**
      * **MergeSort** takes **7.15 x 10-5 Seconds**
      * **HeapSort** takes **1.57 x 10-4 Seconds**
    - **Length** of Sorted List = **5000000**
      * **QuickSort** gives **1.43 Seconds**
      * **MergeSort** takes **5.28 Seconds**
      * **HeapSort** takes **3.79 Seconds**

**OBSERVATION –** It is clear from the data above that QuickSort is able to sort a sorted list faster than other two Sorting algorithms.From the last test case it can be said that quick sort is almost 3 times faster than other two algorithms in Sorted List Case.

*QuickSort Worst Case ( Unsorted ) time complexity is O(n2) and the best Case (Sorted) time complexity is O(nlogn)*

*MergeSort Worst Case and Best Case time complexity is same i.e. O(nlogn)*

*HeapSort Worst Case and Best Case time complexity is same i.e. O(nlogn)*

3. For QuickSort when Pivot Method Selects Middle Term

* + - List Length = 1 it takes **1.6 x 10-5 Seconds**
    - List Length = 25 ( Identical Numbers) it takes **1.01 x 10-4 Seconds**
    - List Length = 5000000 ( Identical Numbers) it takes **2.48 Seconds**
    - List Length = 20 it takes **8.88 x 10-5 Seconds**
    - List Length = 100 it takes **5.99 x 10-4 Seconds**
    - List Length = 500 it takes **3.01 x 10-3 Seconds**
    - List Length = 50000 it takes **6.11 x 10-2 Seconds**
    - List Length = 5000000 it takes **3.19 Seconds**
    - Sorted List Length = 20 it takes **1.24 x 10-4 Seconds**
    - Sorted List Length = 100 it takes **5.99 x 10-4 Seconds**
    - Sorted List Length = 5000000 it takes **9.53 x 10-1 Seconds**

|  |  |  |
| --- | --- | --- |
| List Type & Length | Normal Quick Sort  (return j) Time | Quick Sort Select Middle term (return (i+j)/2) Time |
| Length = 1 | 1.92 x 10-5 Seconds | 1.6 x 10-5 Seconds |
| Length = 25(Identical Numbers) | 3.56 x 10-5 Seconds | 1.01 x 10-4 Seconds |
| Length = 5000000(Identical Numbers) | 2.16 Seconds | 2.48 Seconds |
| List Length = 20 | 4.40 x 10-5 Seconds | 8.88 x 10-5 Seconds |
| List Length = 100 | 3.15 x 10-4 Seconds | 5.99 x 10-4 Seconds |
| List Length = 500 | 3.1 x 10-3 Seconds | 3.01 x 10-3 Seconds |
| List Length = 50000 | 8.35 x 10-2 Seconds | 6.11 x 10-2 Seconds |
| List Length = 5000000 | 4.79 Seconds | 3.19 Seconds |
| Sorted List Length = 20 | 3.76 x 10-4 Seconds | 1.24 x 10-4 Seconds |
| Sorted List Length = 100 | 1.2 x 10-3 Seconds | 5.99 x 10-4 Seconds |
| Sorted List Length = 5000000 | 1.43 Seconds | 9.53 x 10-1 Seconds |

**OBSERVATION –** We can see from the table above that when we edit the quicksort algorithm and choose the middle term first. It shows a trend that it is faster in the case of Unsorted and Sorted List . But in the case of List with identical numbers the time taken is similar.

4. For QuickSort when Pivot Method Selects random item

* + - List Length = 1 it takes **6.44 x 10-4 Seconds**
    - List Length = 25 ( Identical Numbers) it takes **2.95 x 10-4 Seconds**
    - List Length = 5000000 ( Identical Numbers) it takes **4.52 Seconds**
    - List Length = 20 it takes **4.34 x 10-4 Seconds**
    - List Length = 100 it takes **1.03 x 10-3 Seconds**
    - List Length = 500 it takes **3.3 x 10-3 Seconds**
    - List Length = 50000 it takes **7.8 x 10-2 Seconds**
    - List Length = 5000000 it takes **5.11 Seconds**
    - Sorted List Length = 20 it takes **3.06 x 10-4 Seconds**
    - Sorted List Length = 100 it takes **8.66 x 10-4 Seconds**
    - Sorted List Length = 5000000 it takes **2.19 Seconds**

|  |  |  |  |
| --- | --- | --- | --- |
| List Type & Length | Normal Quick Sort  (return j) Time | Quick Sort Select Random term (return (i+j)/2) Time | Quick Sort Select Middle term (return (i+j)/2) Time |
| Length = 1 | 1.92 x 10-5 Seconds | 6.44 x 10-4 Seconds | 1.6 x 10-5 Seconds |
| Length = 25(Identical Numbers) | 3.56 x 10-5 Seconds | 2.95 x 10-4 Seconds | 1.01 x 10-4 Seconds |
| Length = 5000000(Identical Numbers) | 2.16 Seconds | 4.52 Seconds | 2.48 Seconds |
| List Length = 20 | 4.40 x 10-5 Seconds | 4.34 x 10-4 Seconds | 8.88 x 10-5 Seconds |
| List Length = 100 | 3.15 x 10-4 Seconds | 1.03 x 10-3 Seconds | 5.99 x 10-4 Seconds |
| List Length = 500 | 3.1 x 10-3 Seconds | 3.3 x 10-3 Seconds | 3.01 x 10-3 Seconds |
| List Length = 50000 | 8.35 x 10-2 Seconds | 7.8 x 10-2 Seconds | 6.11 x 10-2 Seconds |
| List Length = 5000000 | 4.79 Seconds | 5.11 Seconds | 3.19 Seconds |
| Sorted List Length = 20 | 3.76 x 10-4 Seconds | 3.06 x 10-4 Seconds | 1.24 x 10-4 Seconds |
| Sorted List Length = 100 | 1.2 x 10-3 Seconds | 8.66 x 10-4 | 5.99 x 10-4 Seconds |
| Sorted List Length = 5000000 | 1.43 Seconds | 2.19 Seconds | 9.53 x 10-1 Seconds |

**OBSERVATION –** From the table above we can observe that when we edit the quicksort algorithm(findpivot) to select a random term the results are not assuring anything . As we are choosing a random number we can hit a jackpot by getting close to a sorted list or just choose a pivot which could be worst for us.When the length is 1 the edited algorithm works faster but in other cases algorithm is little slower than the original algorithm . This is due to the fact in the original Quicksort Algorithm we are selecting a fixed term.