Timer.java and Shuffle.java has to be compiled and run for all programs

1. Empirical analysis between Insertion Sort and Merge Sort:

compile : javac InsertionSort.java, javac MergeSort.java

run: java InsertionSort, java MergeSort

Empirical analysis for different powers of 2,

Insertion sort exceeds 15 secs at 2^17

Merge sort exceeds 15 secs at 2^25

|  |  |  |
| --- | --- | --- |
| Power of 2 | Insertion Sort (sec) | Merge Sort (sec) |
| 10 - 14 | 0 | 0 |
| 15 | 1 | 0 |
| 16 | 6 | 0 |
| 17 | 24 | 0 |
| 21 | > 24 | 1 |
| 22 | > 24 | 2 |
| 23 | > 24 | 4 |
| 24 | > 24 | 9 |
| 25 | > 24 | 20 |

3. Quick Sort

compile : javac QuickSort.java

run : java QuickSort

Analysis from output of Dual-pivot and Standard Quick sort with duplicates and random numbers of different sizes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | TIME TAKEN(in msec) | | | |
| Size of the Array | DUPLICATE NUMBERS | | RANDOM NUMBERS | |
|  | Standard Quick Sort | Dual Pivot Quick Sort | Standard Quick Sort | Dual Pivot Quick Sort |
| 10000 | 8 | 1 | 8 | 5 |
| 100000 | 20 | 3 | 25 | 28 |
| 1000000 | 88 | 5 | 160 | 180 |
| 10000000 | 717 | 9 | 1583 | 1673 |

2. MergeSort variations with improvements:

compile : javac CormenMergeSort.java , MergeSortNoCopy, MergeSortWithArrBInsertion, MergeSortWithInsertion, MergeSortTake2

run : java CormenMergeSort , etc

|  |  |  |
| --- | --- | --- |
| Powers of 2 | Cormen Merge Sort (sec) | MergeSort with array B (sec) |
| 20 | 0 | 0 |
| 21 | 1 | 0 |
| 22 | 3 | 1 |
| 23 | 7 | 2 |
| 24 | 18 | 3 |
| 25 | 45 | 8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Powers of 2 | Threshold | Cormen Merge Sort with insertion sort (sec) | MergeSort with array B  with insertion sort (sec) | MergeSort with no copy array B  with insertion sort (msec) |
| 20 | 10 | 0 | 0 |  |
|  | 20 | 0 | 0 | 0 |
|  | 50 | 0 | 0 |  |
|  | 100 | 0 | 0 | 0 |
|  | 200 | 0 | 0 |  |
| 21 | 10 | 1 | 1 |  |
|  | 20 | 1 | 1 | 1 |
|  | 50 | 1 | 1 |  |
|  | 100 | 1 | 1 | 1 |
|  | 200 | 1 | 1 |  |
| 22 | 10 | 3 |  |  |
|  | 20 |  | 2 | 2 |
|  | 50 | 2 |  |  |
|  | 100 |  |  | 2 |
|  | 200 |  |  |  |
| 23 | 10 | 7 |  |  |
|  | 20 |  | 5 | 7 |
|  | 50 | 3 |  |  |
|  | 100 |  |  | 6 |
|  | 200 |  |  |  |
| 24 | 10 | 18 |  |  |
|  | 20 |  | 12 | 9 |
|  | 50 | 5 |  |  |
|  | 100 |  |  | 8 |
|  | 200 |  |  |  |

4. Fibonacci series :

compile : javac FibonacciBigInteger.java, javac FibonacciDac.java, javac FibonacciRecursive.java

run : java FibonacciBigInteger, java FibonacciDac, java FibonacciRecursive

Fibonacci Recursive:

Input : 46

Running Time : 51 secs

Output:

=========================================================================

46th fibonacci number-Recursive

Input Size: 46 Time: 51 secs

=========================================================================

Analysis:

|  |  |
| --- | --- |
| N | Running Time in secs |
| 45 | 31 |
| 46 | 51 |
| 47 | 83 |

Fibonacci Divide and Conquer:

Input : 66999999

Running Time : 61 secs

Output:

=========================================================================

66499999th fibonacci number-DAC

Input Size: 66499999 Time: 59 secs

=========================================================================

Analysis:

|  |  |
| --- | --- |
| n | Running Time in secs |
| 66499999 | 59 |
| 66999999 | 61 |

Fibonacci Dynamic Programming:

Input:1901000

Running Time:59 secs

Output:

=========================================================================

1901000 fibonacci number-Dynamic

Input Size: 1901000 Time: 59 secs

=========================================================================

Analysis:

|  |  |
| --- | --- |
| n | Running Time in secs |
| 1901000 | 59 |
| 1901500 | 61 |