## Project 1: Empirical study of Merge sort

* **Implemented the logic for sorting all values of n not just power of 2 for “Even better Merge Sort”:**
* **If the value of ‘n’ is not a power of 2 then “n” is rounded off to the next greater power of 2.**
* **If ‘n’ is not a power of 2 the arrays are initialized with size as the next greater power of 2 from n.**
* **Initially all the array locations are initialized to the maximum value of an integer using INT\_MAX.**
* **The program then fills the array with the specified values till ‘n’.**
* **The given program is defined to sort an array with number of elements given as power of 2.Since we are rounding off each value of ‘n’ to the next greater power of 2 the logic of the given program works well.**
* **The program then checks if the values till the given integer ‘n’ are sorted.**
* **Note: An additional call to “Merge” routine is made since the given program worked for all odd powers of 2.For even powers of 2 the program ends sorting the two halves of the array so an additional call to merge is made to sort the entire array.**

**(Language of implementations for all the programs : C++)**

**Running time in Milliseconds for the 3 Programs for the specified inputs**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Test case 1:8192  (Running time in MS) | Test case 2:65536  (Running time in MS) | Test case 3: 1048576  (Running time in MS) | Test case 4: 8388608  (Running time in MS) |
| Old Merge Sort | 3ms | 23ms | 504ms | 4030ms |
| Better Merge Sort | 2ms | 18ms | 356ms | 3091ms |
| Even Better Merge Sort(For all values of n) | 1ms | 11ms | 210ms | 2036ms |