**CS 5V81.012 – Special Topics in Computer Science – F16**

**Optional Project 3**

**Team Members:**

**Alwin Johns(axj155830)**

**Divya Vudem(dxv151430)**

**Manoj Kumar Ellanti(mxe150630)**

**SrabontiChakraborty(sxc154030)**

**Question 1:**

**InsertionSort.java and MergeSort.java**has the code to this solution.

From the arrays of sizes 2^17, Insertion Sort is taking more than 15 seconds.

**Insertion sort with array size 2^17**

Time: 59721 msec.

Memory: 3 MB / 123 MB.

From the arrays of sizes 2^24, Merge Sort is taking more than 15 seconds.

**Merge sort with array size 2^24**

Time: 16154 msec.

Memory: 493 MB / 825 MB.

**Question 2:**

For the array of sizes 2^20, below is the time it took to execute using different algorithm

**2.1 Merge sort using 2 arrays L and R**

**MergeSort.java** has the code to this solution.

Time: 1164 msec.

Memory: 28 MB / 155 MB.

**2.2.Using Insertion Sort in Merge Sort**

**MergeSort2.java** has the code to this solution.

Time: 31368 msec.

Memory: 11 MB / 123 MB.

**2.3. Using an array B**

**MergeSort3.java** has the code to this solution.

Time: 13016 msec.

Memory: 384 MB / 629 MB.

**2.4. Using Insertion sort and an array B**

**MergeSort4.java** has the code to this solution.

Time: 30016 msec.

Memory: 384 MB / 629 MB.

**2.5.Avoiding unnecessary copying of elements**

**MergeSort5.java** has the code to this solution.

We have tried executing this code but could not achieve exact functionality. However, we have attached our code in this solution.

**3.Implementing quick sort Standard version v/s Dual Pivot implementation**

**QS.java** has the code to this solution.

We implemented the functionality for various inputs as follows:

for input size:1000 random elements which can be equal:

Dual Pivot method gives 9msec.

normal sorting method gives 8msec.

for input size:10000 random elements which can be equal:

Dual Pivot method gives 39msec.

normal sorting method gives 244msec.

**4.Computing Fibonacci Series using**

**a)Recursive method: FibonacciRec**

**for n=46:**  The running time exceeds 1 minute.

b)**Dynamic Programming: FibonacciDP**

**for n=**very large, we could not exactly determine the exact n value.

**c)Divide and Conquer: FibonacciDAC**

**for n=10005000,** we could get time exceeding 1 minute.