**Algorithms :**

**Lab 7 : 18th February, 2019**

**Sort Algorithms**

Last week you added insertionSort to MyArrays class

Q. 1. Implement Merge Sort algorithm (add it to MyArrays class).

The code is:

**public static void** mergeSort(**int**[] a)  
{  
 **if** (a.**length** <= 1) { **return**; }  
 **int**[] first = **new int**[a.**length** / 2];  
 **int**[] second = **new int**[a.**length** - first.**length**];  
 *// Copy the first half of a into first, the second half into second*

for (int i = 0; i < first.length; i++)

{

first[i] = a[i];

}

for (int i = 0; i < second.length; i++)

{

second[i] = a[first.length + i];

}

*mergeSort*(first);   
 *mergeSort*(second);  
 *merge*(first, second, a);  
}

Header for merge is

public static void merge(int []first, int []second, int [] a)

The algorithm for merge method is:

SET iFirst = 0, SET iSecond = 0, and SET j = 0

WHILE iFirst < first.length and iSecond < second.length  
 IF first[iFirst] < second[iSecond]  
 a[j] = first[iFirst]

increment iFirst  
 ELSE

a[j] = second[iSecond]

increment iSecond

increment j  
ENDWHILE

WHILE iFirst < first.length

a[j] = first[iFirst]

increment iFirst and j

WHILE iSecond < second.length

a[j] = second[iSecond]

increment iSecond and j

Test mergeSort method in TestMyArrays

Q. 2. Arrays class in java api has a sort method. This uses a variant of merge sort. Profile selectionSort, insertionSort, mergeSort and Arrays.sort from the java api

Profile all these sort methods by getting the time taken to sort a large array of random numbers.

Complete the following table.

Size of array = 1000000

|  |  |
| --- | --- |
| **Sort** | **Time to sort in milliseconds** |
| selectionSort | 243564 |
| insertionSort | 74631 |
| mergeSort | 172 |
| Arrays.sort | 94 |

Obtained with Intel(R) Core(TM) i7-7700 CPU, 3600 MHz, Java 10, Windows 10

To find the time, get the time before the sort method call and the time after the method call. Look in System class for a method to give you current time in milliseconds.

Use the following code to create an array of random numbers.

SIZE must be large enough to allow you differentiate between the different sort algorithms.

Random random = **new** Random();  
**final int** SIZE = ;  
**int**[] a = **new int**[SIZE];  
**for** (**int** i = 0; i < a.**length**; i++) {  
 a[i] = random.nextInt(Integer.***MAX\_VALUE***);  
}

Can you explain the results?

Q.3. Repeat Q.2. but this time use an already sorted array instead of a random array.

Use the following code to create an already sorted array. (Note: often the array you are sorting may be partially sorted. To simulate this, insert a few items directly into the array, after the for loop below)

**final int** SIZE =   
**int**[] a = **new int**[SIZE];  
**for** (**int** i = 0; i < a.**length**; i++) {  
 a[i] = i;  
}

Can you explain the results?