Fuzzy Sorting of Intervals

Analysis of Running Time of Find-Intersection():

```
Find-Intersection(A, B, p, s, a, b)
    1.  i ← Random(p, s)                      Θ(1)
    2.  exchange A[i] with A[s]               Θ(1)
    3.  exchange B[i] with B[s]               Θ(1)
    4.  a ← A[s]                              Θ(1)
    5.  b ← B[s]                              Θ(1)
    6.  for i ← p to s - 1                    Θ(n+1)  where n is the total number of elements in A
    7.      do if A[i] ≤ b and a ≤ B[i]       Θ(n)
    8.          then if A[i] > a              Θ(n)
    9.                  then a ← A[i]         Θ(n)
    10.              if B[i] < b              Θ(n)
    11.                 then b ← B[i]         Θ(n)
```

Lines 1 through 5 involve Θ(1) operations. The line 6 iterates over the length of the array hence the time complexity of this line is Θ(n+1) . All the statements from 7 through 11 have a running time Θ(1) each, as these statements are in the for loop each will run for n times making the running time Θ(n). Hence Find-Intersection will therefore run Θ(n) times in the worst case.

Analysis of Running Time

Fuzzy sort partitions the input arrays into “left”, “middle”, and “right” subarrays. These partitions are similar to Quicksort which partitions an input array into one subarray with values less than or equal to the pivot and another subarray with values greater than the pivot. In the worst case the Fuzzy Sort will have no overlapping intervals, in this case it is similar to Randomized quick sort. Hence the worst case time complexity is Θ(n lg n).

With no overlaps the middle will take Θ(n) and the left and right will have a size (n/2) each which will take T(N/2) running time. The recurrence relation will be similar to quick sort that is,

$T(n) = 2T(n/2) + \Theta(n)$

Using Master method on this recurrence relation we have,

$a = 2$, $b=2$ and $f(n) = \Theta(n) = N$

$N^{\log_b a} = N^{\log_2 2} = N^1 = N$ which is equal to f(n)

Hence by case 2 of Master Method the time complexity is $T(n) = \Theta(n \lg n)$

Worst Case time complexity is Θ(n lg n).

In the best case Find-Intersection will always return a non-empty region of overlap. With this every interval will be within the “middle” region. Since the “left” and “right” subarrays will be empty running time of these will be Θ(1).

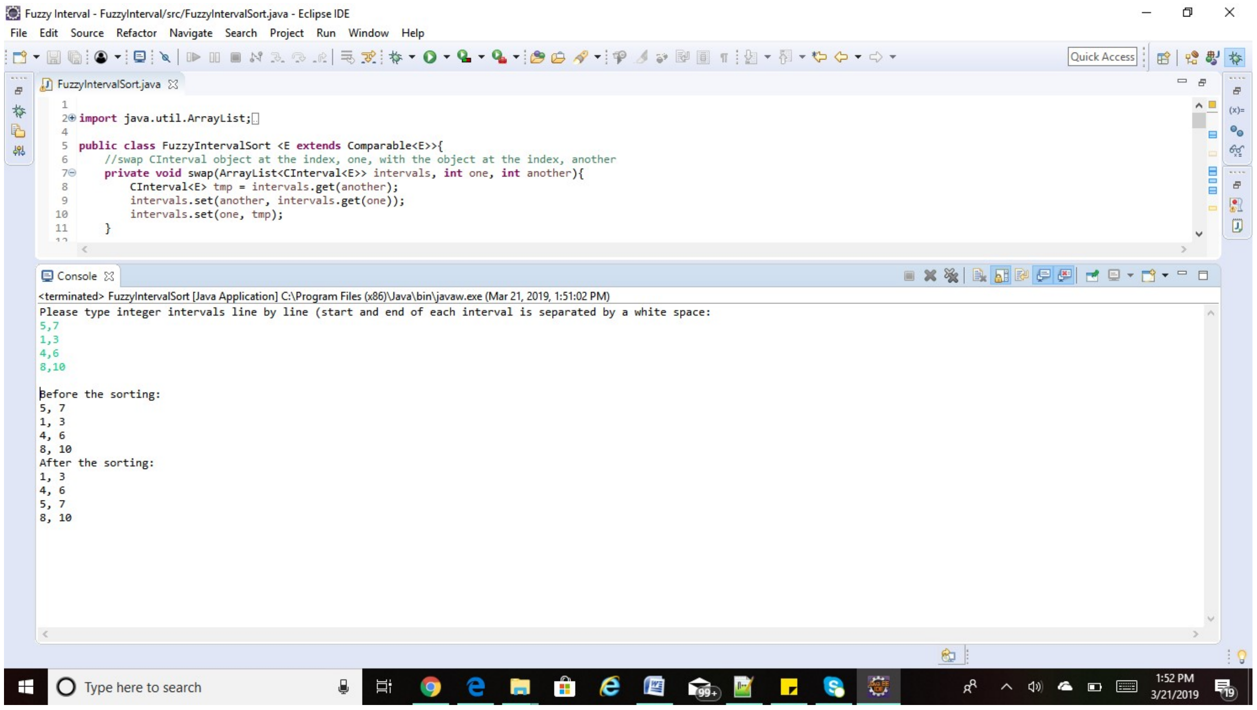
$T(n) = 2T(1) + \Theta(n)$

$T(n) = 2\Theta(1) + \Theta(n)$

T(n)= ϴ(n)

As a result, there is no recursion and the running time of Fuzzy-Sort is the time required for finding overlap which is ϴ(n). Hence if the input intervals all overlap at a point, then the expected running time is ϴ(n).

Console Output:



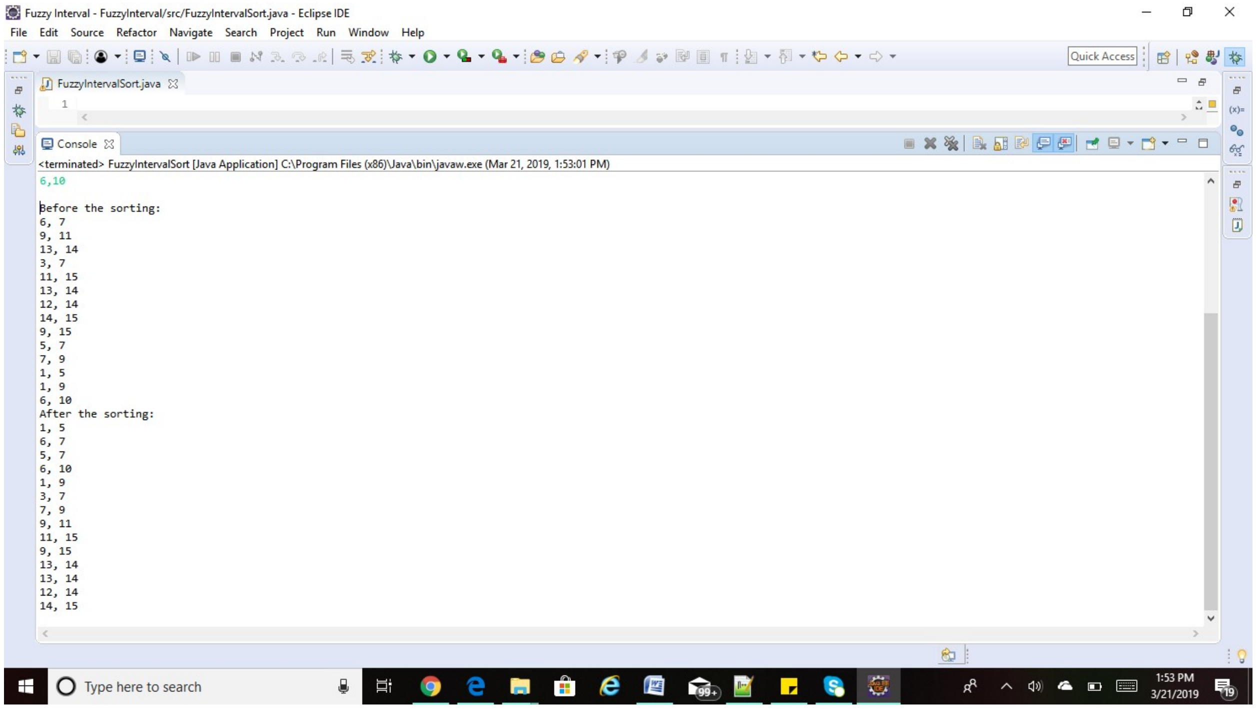
Fuzzy Interval - FuzzyInterval/src/FuzzyIntervalSort.java - Eclipse IDE

FuzzyIntervalSort.java

```java

import java.util.ArrayList;

public class FuzzyIntervalSort <E extends Comparable<E>>{
    //swap CInterval object at the index, one, with the object at the index, another
    private void swap(ArrayList<CInterval<E>> intervals, int one, int another){
        CInterval<E> tmp = intervals.get(another);
        intervals.set(another, intervals.get(one));
        intervals.set(one, tmp);
    }
```

Console

```
<terminated> FuzzyIntervalSort [Java Application] C:\Program Files (x86)\Java\bin\javaw.exe (Mar 21, 2019, 1:51:02 PM)
Please type integer intervals line by line (start and end of each interval is separated by a white space:
5,7
1,3
4,6
8,10

Before the sorting:
5, 7
1, 3
4, 6
8, 10
After the sorting:
1, 3
4, 6
5, 7
8, 10
```



Fuzzy Interval - FuzzyInterval/src/FuzzyIntervalSort.java - Eclipse IDE

Console

```
<terminated> FuzzyIntervalSort [Java Application] C:\Program Files (x86)\Java\bin\javaw.exe (Mar 21, 2019, 1:53:01 PM)
6,10

Before the sorting:
6, 7
9, 11
13, 14
3, 7
11, 15
13, 14
12, 14
14, 15
9, 15
5, 7
7, 9
1, 5
1, 9
6, 10
After the sorting:
1, 5
6, 7
5, 7
6, 10
1, 9
3, 7
7, 9
9, 11
11, 15
9, 15
13, 14
13, 14
12, 14
14, 15
```