

# DC ASSIGNMENT – 1

Name: K. Rahul

Roll No: S20200010091

Problem Statement: Implementing various distributed sorting algorithms on a line network.

- Odd-Even Transposition Algorithm for distributed sorting on a line network:

Output:

For partial order “<=” and n = 5:

```
~\Drives/Drive-D/clg/6/DC/2020-176-dc-assign01/Assign 1/Code/OddEven javac OddEven.java && java OddEven

=====
Enter the value of n (number of processing entities to run the algorithm on) : 5
The Initialized Processing Entities Before Sorting :
127 141 240 36 82

Select The Partial Order for Sorting:
For 'less than partial order (<=)' enter 0      For 'greater than partial order (>=)' enter 1:
0

=====
For Iteration number    0 :
127 141 36 240 82
=====

For Iteration number    1 :
36 127 82 141 240
=====

For Iteration number    2 :
36 82 127 141 240
=====

For Iteration number    3 :
36 82 127 141 240
=====

Total run time: 1 ms

Processing Entities After Sorting :
36 82 127 141 240
```

Time complexity:  $O(n^2)$

Space Complexity:  $O(n)$

- Sasaki's time-optimal algorithm for distributed sorting on a line network:

Output:

For partial “>=” and n = 5:

```

~\Drives\Drive-D\clg\6\DC\2020-176-dc-assign01\Assign 1\Code\Sasaki javac Sasaki.java && java Sasaki

=====n
Enter the value of n (number of processing entities to run the algorithm on) :5
The Initialized Processing Entities Before Sorting :
94 21 63 57 65

Select The Partial Order for Sorting:
For 'less than partial order (<=)' enter 0      For 'greater than partial order (>=)' enter 1:
1

=====n
For Round number : 0
[94* (area = -1)] [21,21 (area = 0)] [63,63 (area = 0)] [57,57 (area = 0)] [65* (area = 0)]
=====n
For iteration number: 1
[94* (area = -1)] [63,21 (area = 0)] [63,21 (area = 0)] [65*,57 (area = 0)] [57 (area = 1)]
=====n
For iteration number: 2
[94* (area = -1)] [63,63 (area = 0)] [65*,21 (area = 0)] [57,21 (area = 1)] [57 (area = 1)]
=====n
For iteration number: 3
[94* (area = -1)] [65*,63 (area = 0)] [63,57 (area = 1)] [57,21 (area = 1)] [21 (area = 1)]
=====n
For iteration number: 4
[94* (area = -1)] [65*,63 (area = 0)] [63,57 (area = 1)] [57,21 (area = 1)] [21 (area = 1)]
=====n
Processing Entities After Sorting :
94 65 63 57 21

Total run time: 19 ms

~\Drives\Drive-D\clg\6\DC\2020-176-dc-assign01\Assign 1\Code\Sasaki

```

Time complexity:  $O(n^2)$

Space Complexity:  $O(n)$

- An alternative time-optimal algorithm for distributed sorting on a line network :

Output:

For aprtial order “<=” and n = 5:

```

~/Dr/Drive-D/c/q/6/DC/2020-176-dc-assign01/Assign 1/Code/AlternatingAlgorithm  javac alternatealgorithm.java && java alternatealgorithm

=====
Enter the value of n (number of processing entities to run the algorithm on) : 5
The Initialized Processing Entities Before Sorting :
39 26 26 51 81
Select The Partial Order for Sorting:
For 'less than partial order (<=)' enter 0      For 'greater than partial order (>=)' enter 1:
0

=====
For Iteration number: 0
[39 | mod value = 0| ] [26 | mod value = 1| ] [26 | mod value = 2| ] [51 | mod value = 0| ] [81 | mod value = 1| ]
=====
For Round no: 1
[26 | mod value = 2| ] [26 | mod value = 0| ] [39 | mod value = 1| ] [51 | mod value = 2| ] [81 | mod value = 0| ]
=====
For Round no: 2
[26 | mod value = 1| ] [26 | mod value = 2| ] [39 | mod value = 0| ] [51 | mod value = 1| ] [81 | mod value = 2| ]
=====
For Round no: 3
[26 | mod value = 0| ] [26 | mod value = 1| ] [39 | mod value = 2| ] [51 | mod value = 0| ] [81 | mod value = 1| ]
=====
For Round no: 4
[26 | mod value = 2| ] [26 | mod value = 0| ] [39 | mod value = 1| ] [51 | mod value = 2| ] [81 | mod value = 0| ]
=====
Processing Entities After Sorting :
26 26 39 51 81
Total run time: 18 ms

~/Dr/Drive-D/c/6/D/2/Assign 1/Code/AlternatingAlgorithm

```

Time complexity:  $O(n^2)$

Space Complexity:  $O(n)$

### Comparison Table:

Table of execution time for different algorithms with different n.

Algo\Time	N= 10	N=20	N=30	N=50
Alternative	64ms	172ms	275ms	316ms
Sasaki	89ms	141ms	266ms	558ms
Odd-Even	18ms	69ms	79ms	122ms

### Observation:

According to the data in the table, it can be observed that the run time increases as the number of processing entities (PE) increase. However, it is not appropriate to compare the algorithms using these execution time values because the PE values are being initialized randomly, which means that the time complexity will be affected by the generated values being in ascending or descending order.

However in general, Odd-Even takes less time to execute because it does not use threading.