**IMPLEMENTATION OF ADVANCED DATA STRUCTURES AND ALGORITHMS**

**Assignment: SP0SL**

**Group: G19**

**Group Members:**

**Jayakarthigayan Sridharan**

**Sagarikha Srinivasan**

**SathyaNarayanan Srinivasan**

Skip – List Implementation:

Methods Implemented:

add, remove, contains, ceiling, findIndex, first, floor, isEmpty, last, size.

Approach:

The approach for inserting elements into the skip list and to know which of the above operations is completely randomized.

A random number is used in order to determine which of the above operations is to be performed (index – variable). A random number to be used during the operation is then generated. This is done for how many iterations the operation needs to be done.

Comparing treeSet and Skip-list:

A thread is used to run the two data structures in a parallel fashion. Timer is started during the start of the runnable method and then time taken to execute two threads is measured along with memory used. It is seen that from our results, skip list and tree set has the same execution time whereas skip list takes lesser memory to execute compared to tree set.

P.S: We used executor service to obtain a even more perfect result of running the two data structures parallely and since the results ie; timer and memory used for both the data structures were equal we felt threads provides a slight variance to indicate skip lists are slightly better than tree sets. But executor service takes lesser time to execute/perform operations compared to threads. For instance for iteration of 1million, executor service takes 40883 msecs.

Comparison Results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ITERATIONS | SKIP LIST TIME | TREE SET TIME | SKIP LIST MEM | TREE SET MEM |
| 10 | 64 ms | 64 ms | 20MB | 20MB |
| 100 | 554ms | 554ms | 46MB | 46MB |
| 1000 | 2605ms | 2605ms | 31MB | 44MB |
| 10000 | 17258ms | 17258ms | 222MB | 236MB |
| 100000 | 161005ms | 161005ms | 592MB | 625MB |