Chinmay Ratnaparkhi
Lab Assignment 08
EECS 650 Data Structures
April 13th, 2015

Overall Organization

The overall architecture of the project included two different kinds of data structures, namely, the Leftist Heap and Skew Heap. At first random numbers are generated for insertion in the aforementioned data structures, the number of random numbers goes on increasing for each iteration; it begins at 50,000 and doubles for every iteration, ending at 400,000. These numbers are sequentially inserted into the implemented data structures in the following manner :

1. Leftist Heap
    a. The timer is started
    b. “n” elements are inserted into the leftist heap
    c. The timer is stopped and time is recorded in an array.
2. Skew Heap
    a. The timer is started
    b. “n” elements are inserted into the skew heap.
    c. The timer is stopped and time is recorded in an array.

The above three steps are repeated three more times, with an increasing value of n ranging from 50,000 to 400,000. Final steps involve average calculation and printing the final results on the console.

Data Generation

The data for this project was generated using two functions, namely, rand() and srand(). The latter i.e. srand() ensures that the initial value is not constant through the multiple runs of the the program.

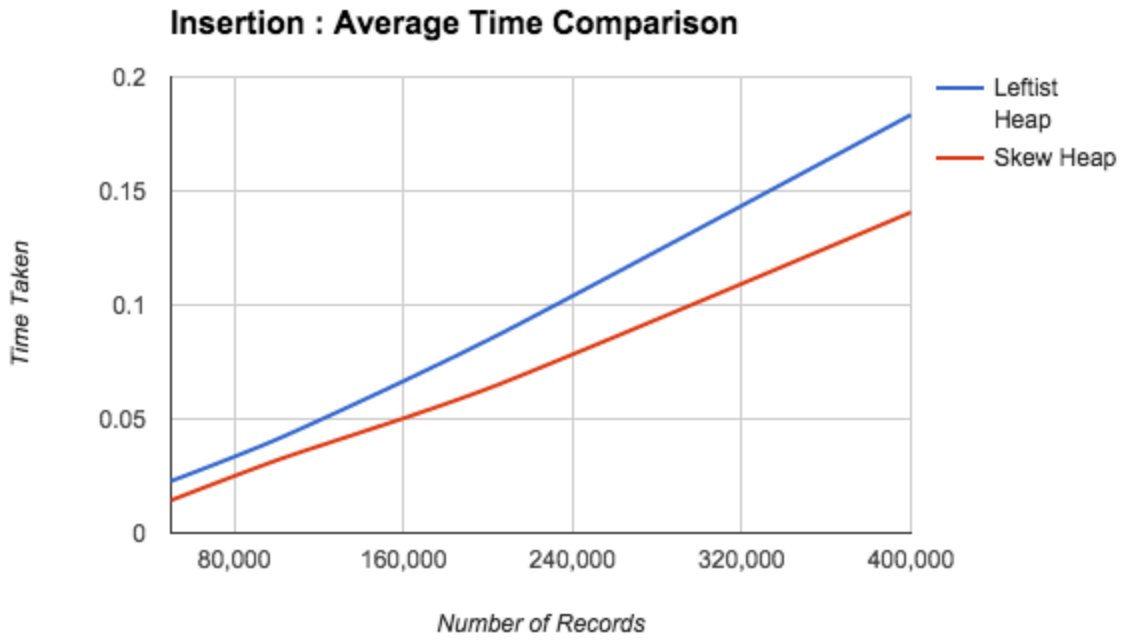
These values were stored in an array to ensure that all the two implemented data structures, Leftist heap and Skew heap were tested with the identical values. Implementation of an array also allowed the data structures to access the data in constant time. This was done so as to create uniformity in the testing and to minimize errors.

Summary of Results

Following are the times (in seconds) required to perform the insertions in the respective data structures.

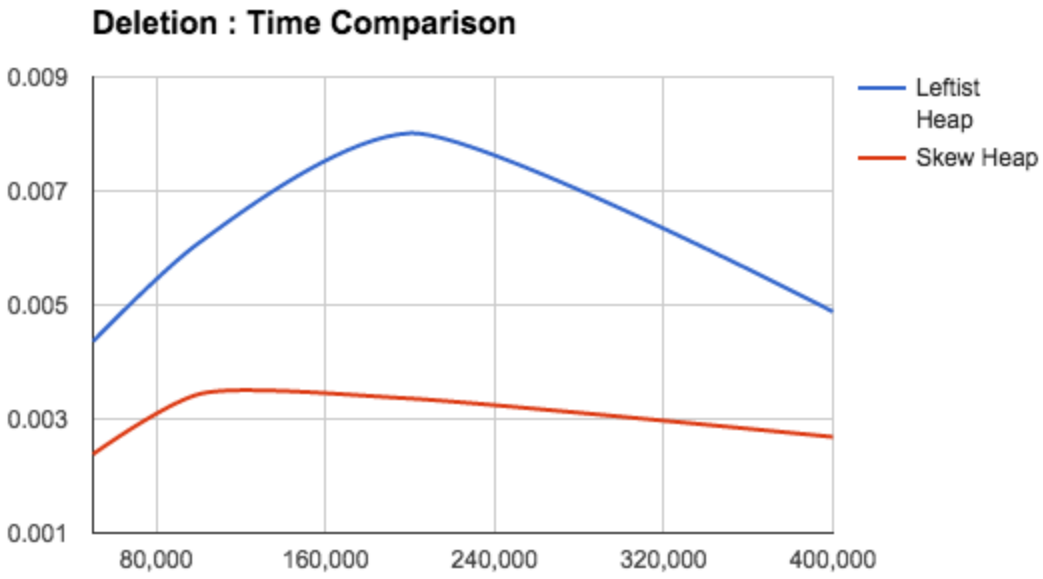
Insertion of Records

	50,000	100,000	200,000	400,000
Leftist Heap	**0.0227582**	**0.0410312**	**0.0846254**	**0.183396**
Skew Heap	**0.0143268**	**0.0318812**	**0.063361**	**0.140654**



Delete Operation

	50,000	100,000	200,000	400,000
Leftist Heap	**0.00436**	**0.006086**	**0.008014**	**0.004882**
Skew Heap	**0.002381**	**0.003434**	**0.003359**	**0.002685**



Observation and Conclusion

The results received were satisfactory to the expectations. It was apparent that the number of records (nodes) and the time required for execution are directly proportional, i.e. as the number of records goes on increasing, the time required to perform the insertions and deletions also goes on increasing.

It was also found that in general, the Skew Heap shows better performance than the Leftist Heap. It is apparent from the graphs that when the same number of operations is performed on both Leftist Heap and Skew Heap, the latter finishes the task in less time than the prior. It can be speculated that skew heap works faster than leftist heap since it does not have to calculate the rank to decide if a swap is necessary. Performing a simple pointer swap turns out to be cheaper than calculating the rank for each node to make a decision.