EECS 560: Lab 9 – Comparing the performance of Leftist-Heap and Skew-Heap

Shane Chu

November 12, 2016

1 Overall organization of the experiment

1.1 Code arrangement

To run the experiment, we simply construct an integer array with the number of generations we specified, which is the set of integers $\{50000, 100000, 200000, 400000\}$, hard coded inside the array. Then we construct a nested for-loop to record the time for Leftist-Heap and Skew-Heap both on its build time and operation time.

Each element inserted into the structure is generated by a random integer between 1 and $4 \times n$, where n is the number of generations. To build the structure, Leftist-Heap and Skew-Heap each inserts n number of elements. Note that each insert performs a merge function.

1.2 Data

The time-measured data on both data structure is arranged to be output as a text file, which is taken care by using fstream library. That way, we could simplify the process of visualizing the data.

1.3 Run the program

After the code is written, compile the file main.cpp under the folder using:

```
g++ -std=c++11 main.cpp
```

Then run the executable a.out to generate the data.

2 Data Generation

After running a.out, a text file result.txt is generated in the folder.

We process the output of the text file using programming language *python* and its library *matplotlib*. Further, we use *iPython notebook* (jupyter) so we could code and plot the result at the same time. The whole process is documented in parse-result.ipynb.

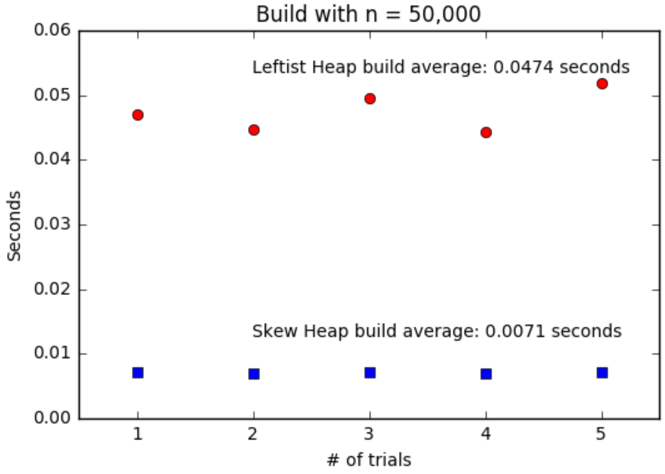
3 Results

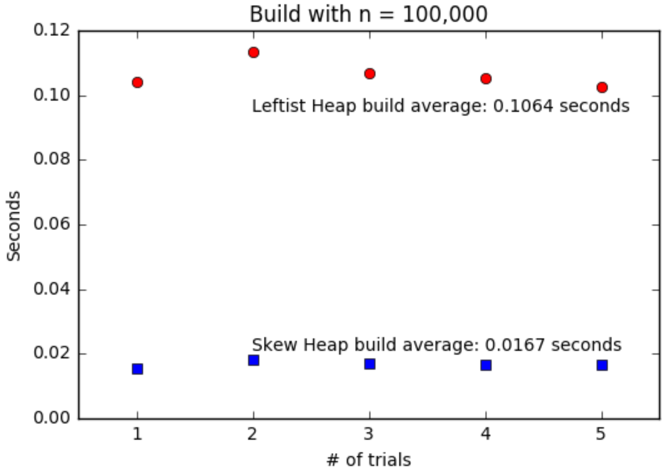
3.1 Data

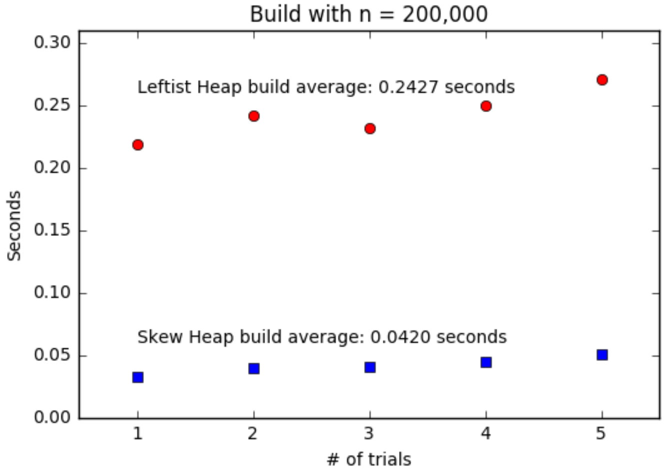
Please refer to the text file result.txt.

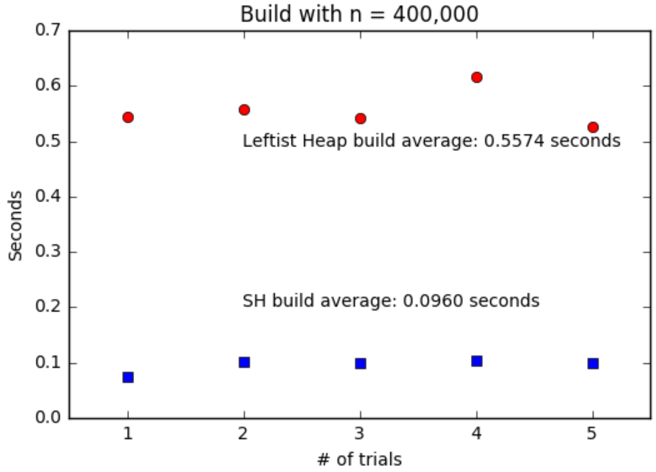
3.2 Build

Below are the graphs that show the results of building each data structures by inserting *n* elements (*n* is the number of random generations).



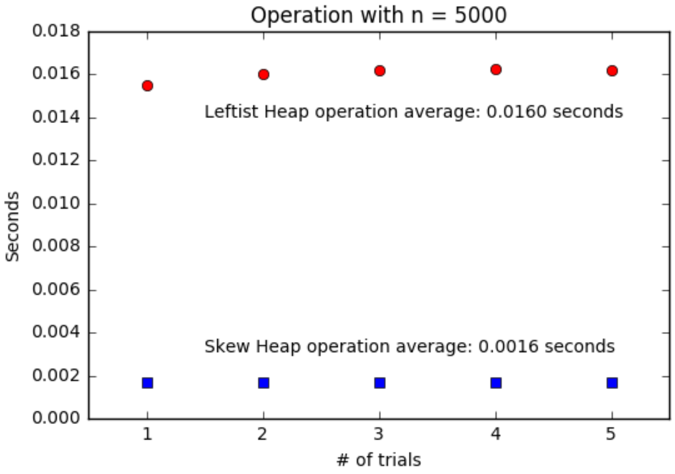


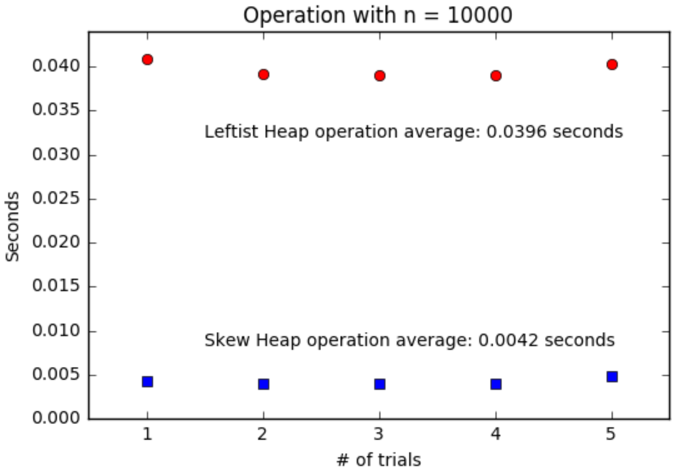


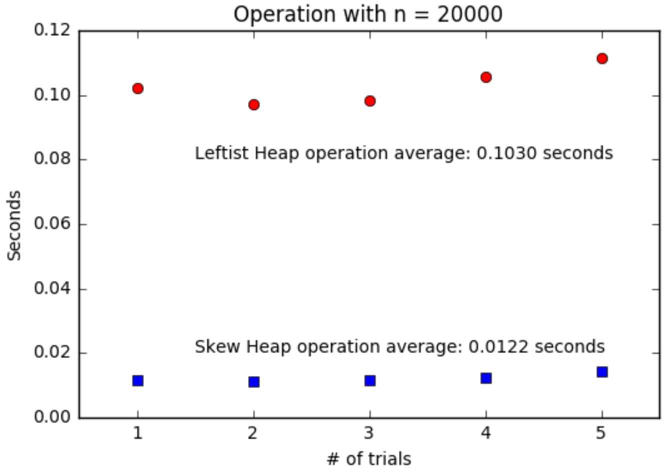


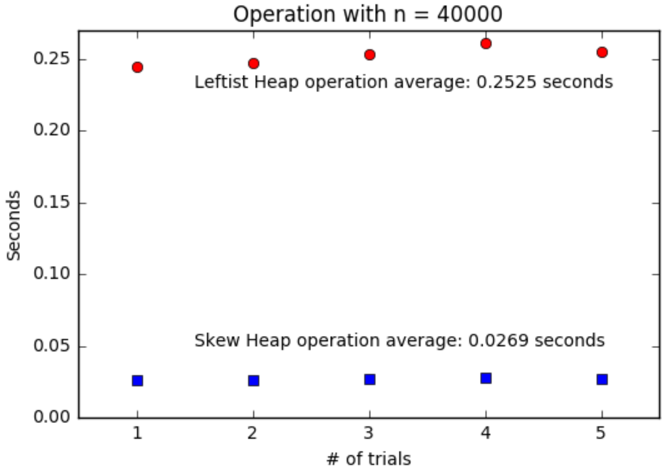
3.3 Operation

Below are the graphs that show the results of *n* operations on each data structures (*n* is the number of random generations divided by 10).

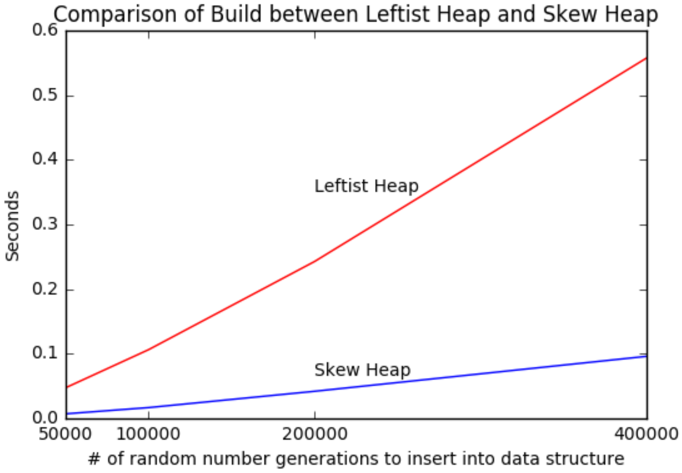


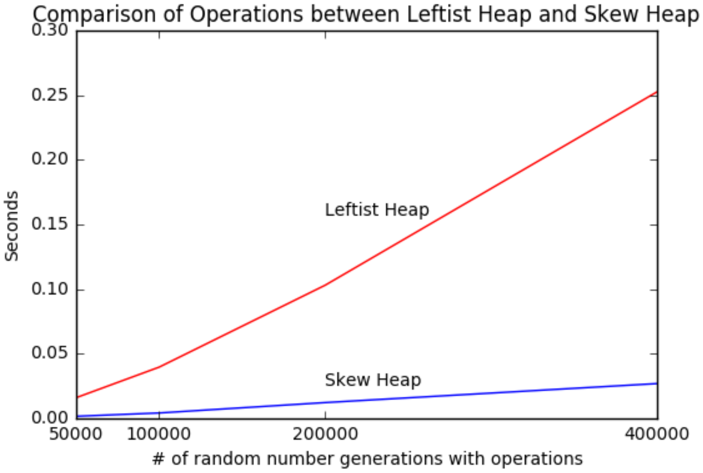






3.4 Performance Comparison





4 Conclusion

The *build* process takes longer time in Leftist-Heap than in Skew-Heap. Furthermore, as the input size increases, the time it takes to build a Leftist-Heap becomes larger than it takes to build a Skew-Heap. Theoretically, both takes $O(n\log(n))$ run time. The difference between them is that in Leftist-Heap, the data structure has to keep track of the rank of the each nodes, whereas in Skew Heap, the sub-trees swap sides every time it executes the merge and insert function.

The operations (*insert*) in Leftist-Heap takes longer time to complete than Skew-Heap. We can see that as the element of data structure increases, the difference between them grew larger. Theoretically,