Cole Jurden
Lab 9 EECS 560
Mehrdad Hajiarbabi
11-17-2015

# Lab 9 Report

## Organization

The lab was organized using loops and writing directly to CSV files to minimize code and potential user error during data input. An outer loop defined 4 values of n: 50,000, 100,000, 200,000, and 400,000. An inner loop declared both heap structures (leftist and skew), kept track of seed values to generalize random number generation, and involved the actual timing and recording of each operation for each heap. Timing data was stored in an array to generate the average which was calculated for each n. It was also written directly to one of 6 individual CSV files: leftist build timing, skew build timing, average build timing, leftist insert/delete timing, skew insert/delete timing, and average insert/delete timing.

## Data Generation

The timing data was generated in two stages. First, the program timed how long it took for each heap (Leftist and Skew) to be constructed, respectively. This data was recorded for heaps of size 50,000, 100,000, 200,000, and 400,000. The values inserted were generated with a random number generator by 5 different seeds (1, 6, 11, 16, 21, 26) and were in the range of 1 to 4 times the final size of the heap. Timing data was collected for each individual seed, and average timing data was derived for all 5 seeds. The results are as follows:

Leftist:

| Num | 1 | 6 | 11 | 16 | 21 | 26 |
|---|---|---|---|---|---|---|
| **50000** | 0.014476 | 0.017818 | 0.015043 | 0.014589 | 0.015818 | 0.015204 |
| **100000** | 0.032663 | 0.029701 | 0.033645 | 0.028844 | 0.026867 | 0.028465 |
| **200000** | 0.066692 | 0.063192 | 0.058735 | 0.061504 | 0.059089 | 0.058757 |
| **400000** | 0.129148 | 0.129305 | 0.133427 | 0.122918 | 0.124583 | 0.117005 |

Skew:

| Num | 1 | 6 | 11 | 16 | 21 | 26 |
|---|---|---|---|---|---|---|
| **50000** | 0.010601 | 0.009813 | 0.008731 | 0.008471 | 0.008842 | 0.009973 |
| **100000** | 0.018928 | 0.020693 | 0.023162 | 0.018717 | 0.018629 | 0.020255 |
| **200000** | 0.044687 | 0.04581 | 0.038753 | 0.037114 | 0.042819 | 0.038567 |
| **400000** | 0.096169 | 0.088018 | 0.085415 | 0.085142 | 0.083735 | 0.088876 |

Average:

| Num | Left | Skew |
|---|---|---|
| **50000** | 0.023237 | 0.0141077 |

| | | |
|---|---|---|
| **100000** | 0.0450463 | 0.030096 |
| **200000** | 0.0919922 | 0.0619375 |
| **400000** | 0.189097 | 0.131839 |

Second, data was collected about the run time of insert and deletemin operations. The operations were executed 10% of the final size of the heap times. To determine which operation would be executed each time, a random number was generated between 0 and 1. If the number was < .5, deleteMin was performed on the heap. If >=.5, a random number between 1 and 4 times the final size of the heap was generated and inserted accordingly. Results are as follows:

Leftist:

| **Num** | **1** | **6** | **11** | **16** | **21** | **26** |
|---|---|---|---|---|---|---|
| **50000** | 4.70E-05 | 3.80E-05 | 3.70E-05 | 4.10E-05 | 4.00E-05 | 4.80E-05 |
| **100000** | 7.20E-05 | 7.40E-05 | 9.80E-05 | 7.20E-05 | 7.60E-05 | 8.70E-05 |
| **200000** | 0.000148 | 0.000186 | 0.000182 | 0.000143 | 0.000148 | 0.000144 |
| **400000** | 0.000413 | 0.000287 | 0.000286 | 0.000287 | 0.000287 | 0.000342 |

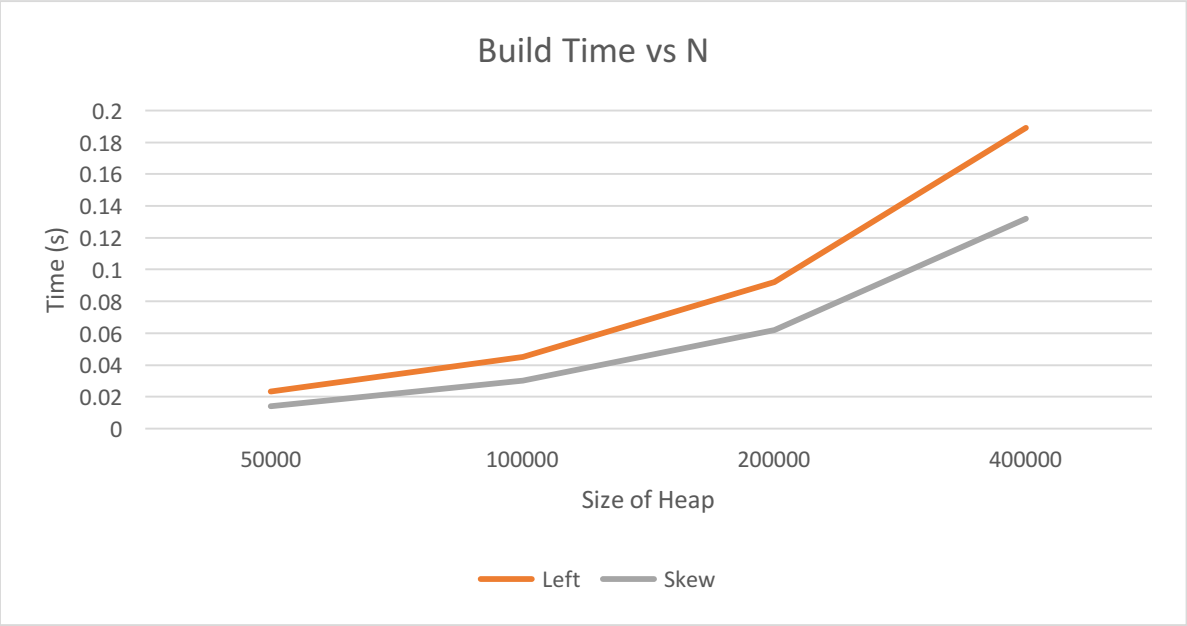
Skew:

| **Num** | **1** | **6** | **11** | **16** | **21** | **26** |
|---|---|---|---|---|---|---|
| **50000** | 4.70E-05 | 3.80E-05 | 3.70E-05 | 4.10E-05 | 4.00E-05 | 4.80E-05 |
| **100000** | 7.20E-05 | 7.40E-05 | 9.80E-05 | 7.20E-05 | 7.60E-05 | 8.70E-05 |
| **200000** | 0.000148 | 0.000186 | 0.000182 | 0.000143 | 0.000148 | 0.000144 |
| **400000** | 0.000413 | 0.000287 | 0.000286 | 0.000287 | 0.000287 | 0.000342 |

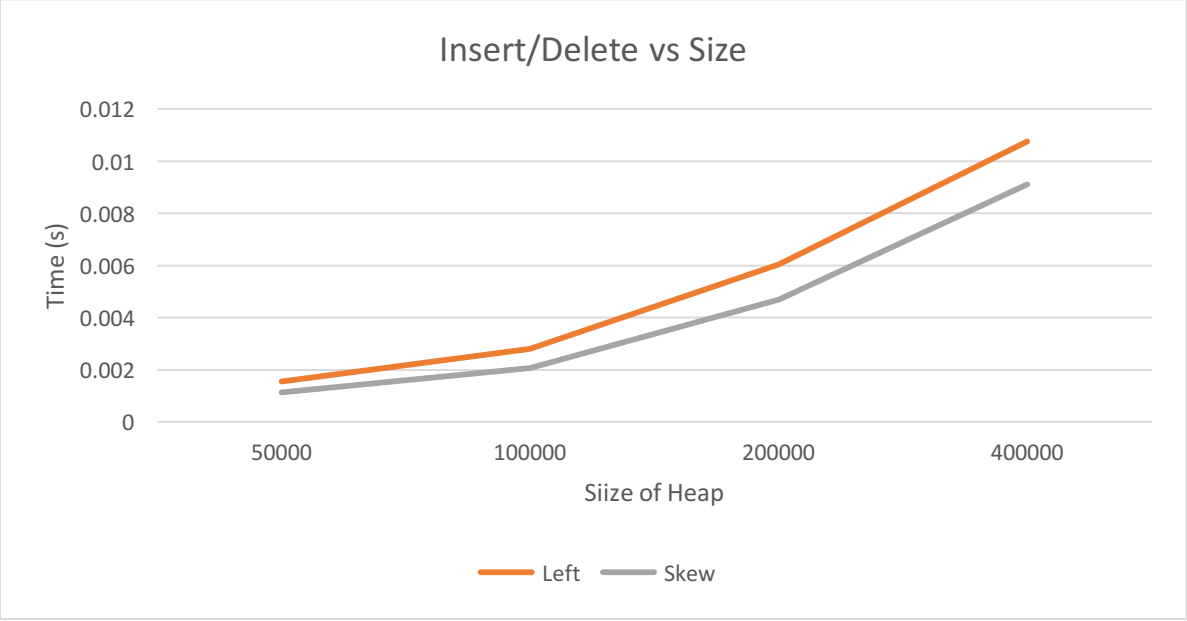
Average:

| **Num** | **Left** | **Skew** |
|---|---|---|
| **50000** | 6.28E-05 | 6.35E-05 |
| **100000** | 0.00011975 | 0.0001235 |
| **200000** | 0.00023775 | 0.00022375 |
| **400000** | 0.0004755 | 0.000496 |

The calculated average build time of each type of heap for each size generated the following graph:




It is apparent that Leftist built slower than skew for each of the sizes in question. This may be in part because skew swaps the left and right subtree in every merge call, while leftist merge only swaps according to the rank, requiring an extra function call to check and adjust rank at each merge.




Again, leftist requires more time to perform insert/delete operations than skew. However, the rate of divergence for the difference in time as size increases is less in this case versus build. I attribute that to the fact that in insertion and deletion, we are performing those operations less

and heap size is remaining relatively similar when compared to constantly increasing as in the building of the heaps.

In conclusion, largely because of the added aspect of calculating and considering rank, leftist heaps perform marginally less efficiently than skew heaps when building the heaps, inserting and deleting elements to/from the heap are concerned. As the size of the heap increases, so does the margin of difference between the timing of each operation, leftist becomes less efficient faster than skew, however both have time complexity of O(lgn) for all of the operations tested.