EECS 560 LAB 8 REPORT

Stephen Wiss 2524249

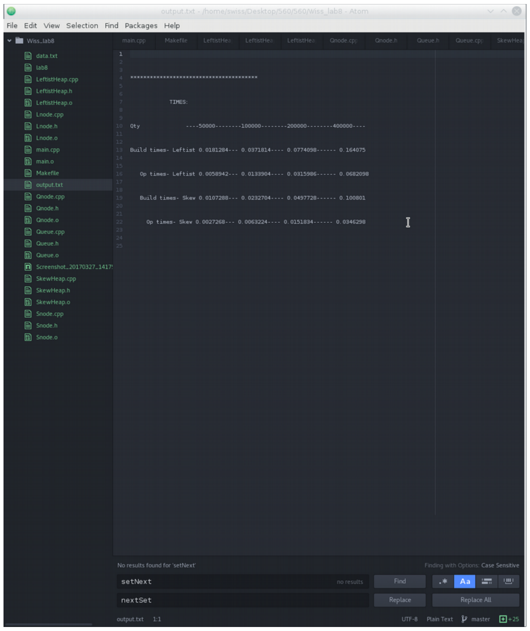
1. Organization/Setup

There were 5 trials for each of the different amounts of randomly generated numbers. The amount of numbers used increased to give a better representation of how performance was affected by data set size. The times' averages were stored into a group of 2-d arrays. These arrays were then format- printed to a textfile (output.txt).

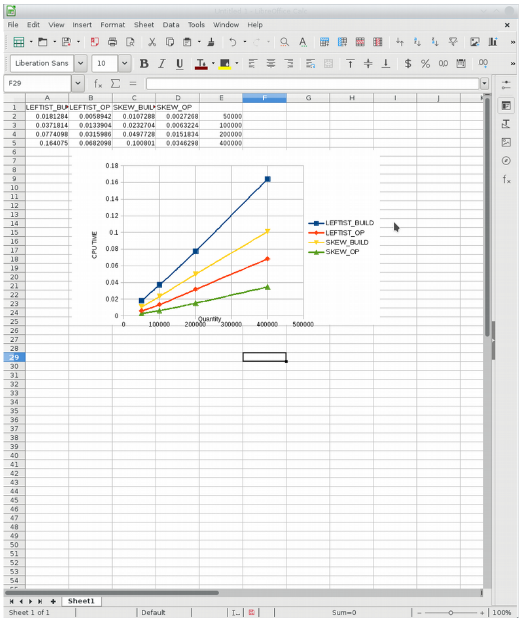
2.Data Collection/Observation

For every Quantity of Numbers used. (50k, 100k, 200k, 400k), The Heap was filled with an amount of randomly generated numbers equal to which ever amount the sequence was currently testing. After the Heaps were built, and the time taken to compute their builds was stored to an array. I used a random number generator to perform a sequence of operations on the Heaps. The number generated dictated which (DeleteMin or Insert) operation was performed. I made sure to perform a number of operations equal to 10 percent of the size of the Heap.

3. Conclusion/Screenshots


output.txt - /home/swiss/Desktop/560/560/Wiss_lab8 - Atom
File Edit View Selection Find Packages Help
main.cpp
Makefile
Qnode.cpp
Qnode.h
Queue.h
Queue.cpp

```


***************************************


             TIMES:


Qty              ----50000--------100000--------200000--------400000----


Build times- Leftist 0.0181284--- 0.0371814---- 0.0774098------ 0.164075


   Op times- Leftist 0.0058942--- 0.0133904---- 0.0315986------ 0.0682098


   Build times- Skew 0.0107288--- 0.0232704---- 0.0497728------ 0.100801


      Op times- Skew 0.0027268--- 0.0063224---- 0.0151834------ 0.0346298
```



LEFTIST_BU	LEFTIST_OP	SKEW_BUIL	SKEW_OP	
0.0181284	0.0058942	0.0107288	0.0027268	50000
0.0371814	0.0133904	0.0232704	0.0063224	100000
0.0774098	0.0315986	0.0497728	0.0151834	200000
0.164075	0.0682098	0.100801	0.0346298	400000

As shown in the screenshots provided. The build times grew at similar rates with the Leftist Build being slightly outperformed by the Skew. Both begin to grow quickly as the data set grows. The operations times was similar in that the Skew heap outperformed the Leftist heap, Although overall it's rate of growth was slower than the Build time's. The leftist heap becomes unbalanced and therefore less optimal to traverse because of the nature of the Data structure, I would blame this, in part for it's under performance in comparison to the Skew heap data structure. The Skew heap is after all, capable of adjusting itself in each merge.