Richard Aviles

Lab 09

This lab was conducted to find the time performance difference for the leftist-Heap and a skew-Heap. I first created the two structures from my previous two labs and updated to fix any mistakes and allow for duplicate entries. I then measured the time it took to build the structures using different sizes, and the time it took to perform delete and adds.

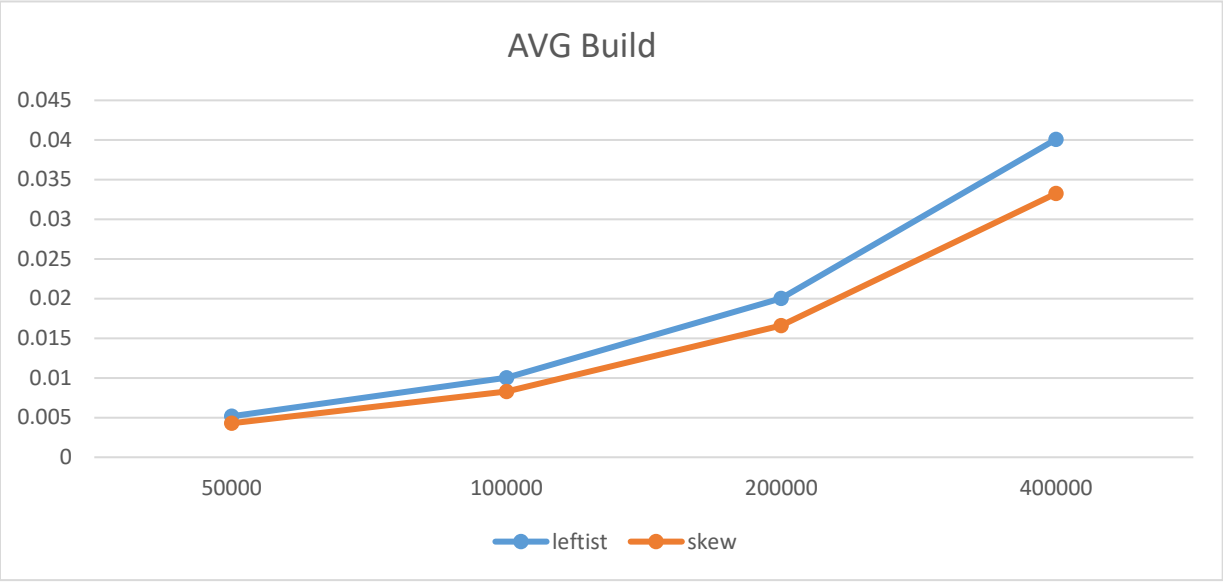
I started the lab by creating a for loop that ran 20 times. This was to calculate the five iterations of each of the four sizes: 50000, 100000, 200000, and 400000. I used the variable i in the loop to use a different seed for each iteration of the test. I generated an array to hold all the values that were to be inserted in the build at the beginning of each loop so that I could start the timer with the random numbers already generated. Each iteration used a different seed, thus giving different values to be used for each test run. I used the array to ensure that both structures I built contained the same information thus creating a fair timing.

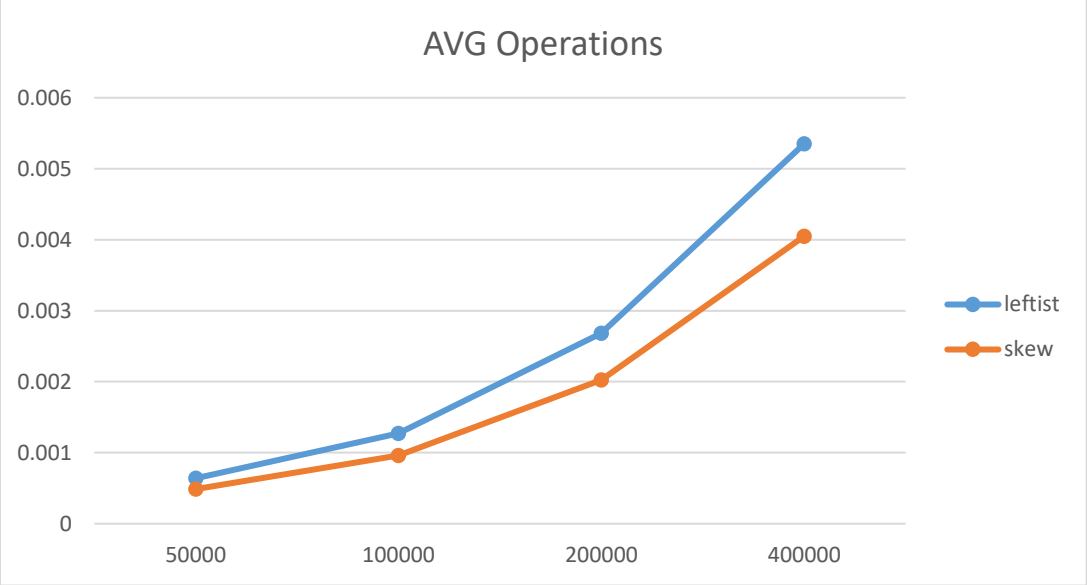
After timing the builds of both structures, I then used those already built structures to time how long it took for each structure to use the operations delete and insert. To determine which to use, I created an array to hold numbers varying from 0 to 1. Anything less or equal to 0.5 used the deleteMin function. All numbers higher than 0.5 invoked the insert operation. I used a separate array that held values for which operation was to be performed, thus creating a fair time for both. After each timing run, I deleted the structures built so that they were ready for building again at the next iteration. When the variable i reached %5 = 0, I then changed the size of the number of elements to be used for the next five iterations. All generated data can be seen in the attached charts.

What I observed from this lab is that the skew-Heap outperformed the leftist-Heap in both the build and the operations performed. The differences, however, were not much as can be seen from the data. The small differences in the two structures makes it feasible to use either one if given a choice. Even with the extremely large amount of input, both structures completed everything in less than one half a second. For deleting the minimum priority and building, these two have the best overall numbers.

	Build			Operation								
n	**leftist-Heap**	skew-Heap		leftist-Heap	skew-Heap							
50000	0.005241	0.004341		0.000693	0.000528							
	0.005191	0.004315		0.000639	0.00049							
	0.00521	0.004318		0.000601	0.000459							
	0.005204	0.004303		0.00067	0.000513							
	0.005015	0.00416		0.000595	0.000446							
100000	0.010004	0.008314		0.001269	0.00096							
	0.010021	0.008306		0.001293	0.000973							
	0.010021	0.008319		0.001284	0.000971							
	0.010006	0.00831		0.001167	0.000875							
	0.010026	0.008302		0.001351	0.00103							
200000	0.02002	0.016605		0.002616	0.001982							
	0.020035	0.016645		0.002762	0.002093							
	0.020083	0.016624		0.002738	0.002076							
	0.020012	0.016627		0.002654	0.001999							
	0.020023	0.016609		0.002638	0.00197							
400000	0.040083	0.033267		0.005284	0.003978							
	0.040106	0.033252		0.005367	0.004052							
	0.04006	0.033261		0.005483	0.004171							
	0.040047	0.033236		0.005306	0.003991							
	0.040118	0.033233		0.005317	0.004057							

		AVG										
	leftist-Heap Build	skew-Heap Build		leftist-Heap Operations	skew-Heap Operations							
n												
50000	0.0051722	0.0042874		0.0006396	0.0004872							
100000	0.0100156	0.0083102		0.0012728	0.0009618							
200000	0.0200346	0.016622		0.0026816	0.002024							
400000	0.0400828	0.0332498		0.0053514	0.0040498							


AVG Build
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
50000
100000
200000
400000
leftist
skew


AVG Operations
0.006
0.005
0.004
0.003
0.002
0.001
0
50000
100000
200000
400000
leftist
skew