Richard Aviles

Lab 07

This lab was conducted to find the time performance difference for the minmax-Heap and a min-Heap. I first created the minmax-Heap and then took the min-Heap structure from my previous lab 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.

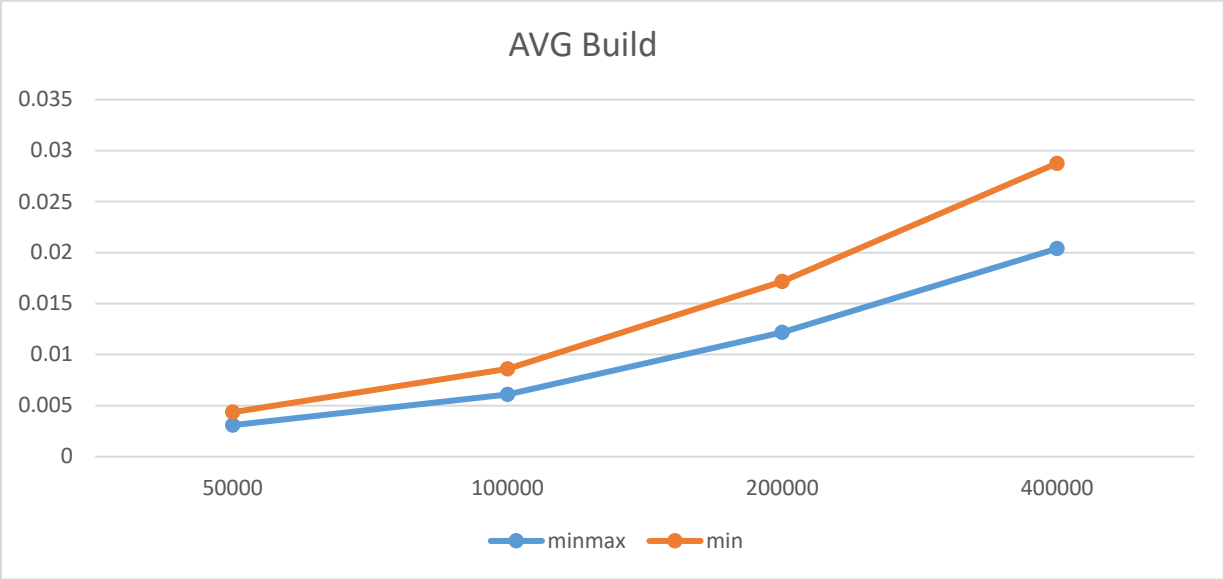
After timing the builds of both structures, I then used those already built structures to time how long it took for each structure to use the operations delete and insert. I used a separate array that held values for which operation was to be performed, thus creating a fair time for both. After each 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.

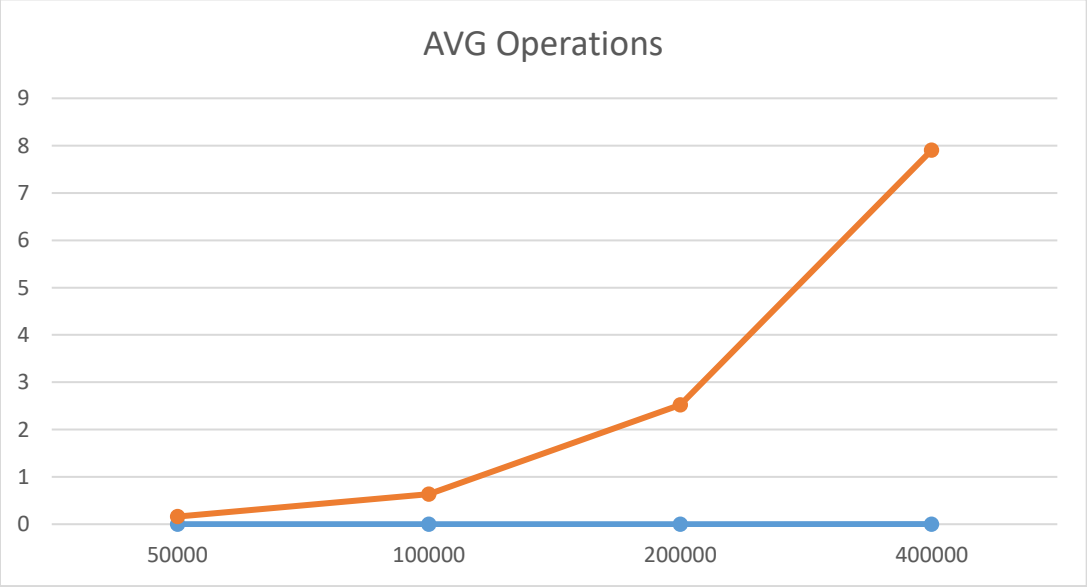
What I saw from the data is that in the build part of the lab, the min-Heap outperformed the minmax-Heap. For all the size input, the min-Heap was roughly twice as fast, but the timing it took was still extremely fast. When using the operations, delete and insert, the minmax-Heap structure was the far better option. The time difference was drastically different once the size started increasing, but was still very noticeable with the smaller sizes. Attached are two graphs showing the average times of both, and one can see from the operations side the huge difference in the two. From this I can conclude that all though the min-Heap is faster in building, the time difference in deleting and adding after the build is so different that I would have to say the use of the minmax-Heap is a better option, especially with large sized input. The smaller time difference in the build is not a big factor when the operations side is so different. If deleting was not a big deal or not commonly used, then its suffice to say either one is a good option.

	Build		Operation	
n	**minmax-min-Heap**	min-Heap	minmax-min-Heap	min-Heap
50000	0.003234	0.00137	0.00033	0.157351
	0.003046	0.001245	0.000355	0.161619
	0.003038	0.001266	0.000328	0.162624
	0.003037	0.001236	0.000401	0.160784
	0.003053	0.001237	0.000348	0.161768
100000	0.006147	0.002496	0.000775	0.624633
	0.006081	0.002496	0.000633	0.639343
	0.006135	0.002505	0.000622	0.634101
	0.00608	0.002479	0.000804	0.634045
	0.006097	0.002483	0.000621	0.6429
200000	0.012253	0.0051	0.001535	2.535617
	0.012144	0.004955	0.001524	2.490799
	0.012144	0.004967	0.001324	2.491513
	0.012171	0.004981	0.001231	2.515599
	0.012164	0.004982	0.001342	2.556814
400000	0.024543	0.010179	0.003473	8.001864
	0.01919	0.007979	0.002077	7.910352
	0.019138	0.007812	0.001902	7.823454
	0.01976	0.007817	0.002673	7.857452
	0.019333	0.008031	0.002379	7.913428

AVG

n	minmax-Heap Build	min-Heap Build	minmax-Heap Operations	min-Heap Operations
50000	0.0030816	0.0012708	0.0003524	0.160829
100000	0.006108	0.0025018	0.000691	0.635004
200000	0.01218	0.004997	0.0013912	2.51807
400000	0.0203928	0.0083636	0.0025008	7.90131


AVG Build
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
50000
100000
200000
400000
minmax
min


AVG Operations
9
8
7
6
5
4
3
2
1
0
50000
100000
200000
400000