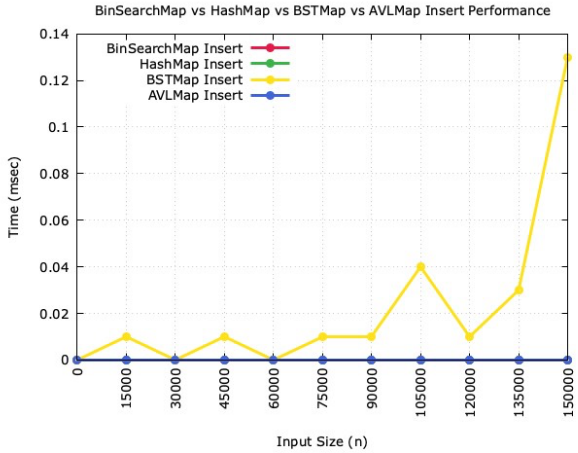
NAME: Ben Puryear

FILE: HW-8_WriteUp.pdf

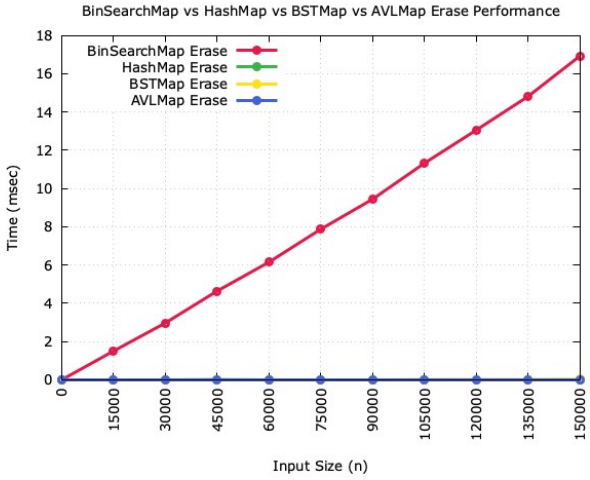
DATE: Fall 2021

DESC: This pdf goes over the basics accomplished throughout all the HW-8 related files

Graphs / Explanation:

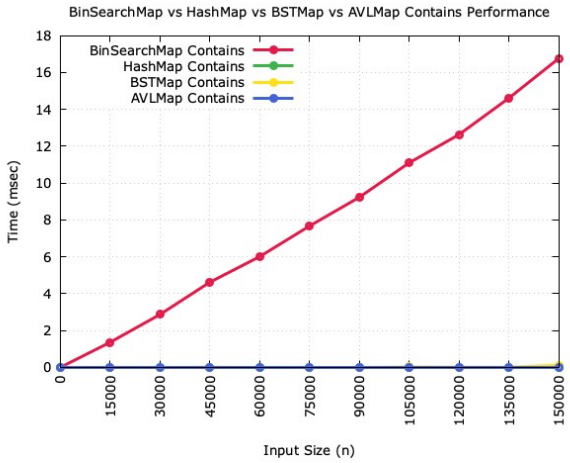


This first graph is the graph for insert. This graph clearly shows the difference between BSTMap's insert (O(n)) and AVLMaps insert (o(log(n))). This is because of how AVLs are constructed. AVLs are made to always be balanced, and the times when BSTs take a lot longer are when they are extremely onesided weight-wise.

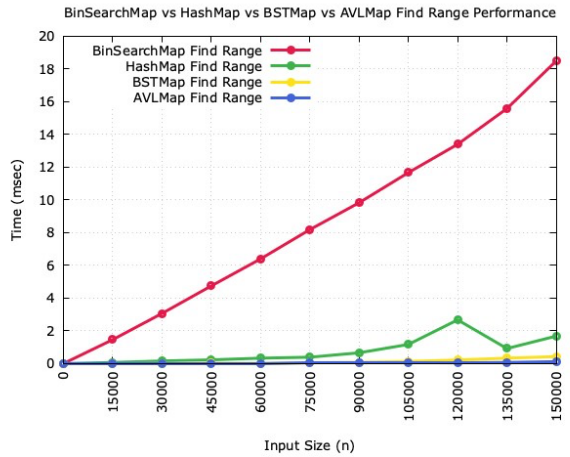


This next graph is for the performance of erase. This shows how well AVLmap outperforms BinSeachMap when it comes to erase. This also comes back to the

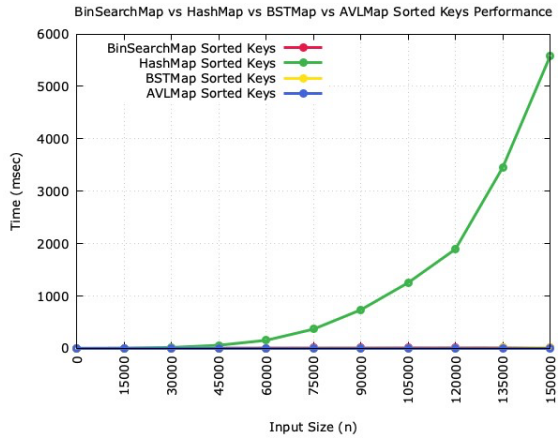
fact that AVL maps are balanced. A BinSeachMap has the potential to need to traverse basically a linked list in order to erase a key/value.



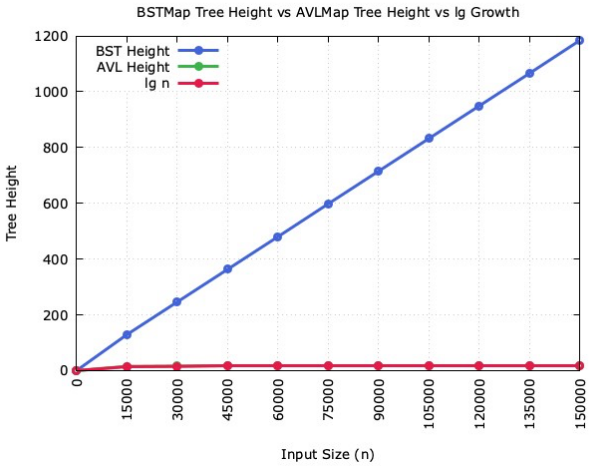
This graph shows the performance of contains. Similarly to the previous graphs, this also shows how many problems keeping a tree balanced can solve. BinSearchMap also has the potential to need to traverse a linked list in order to find it, causing it to be o(n), whereas AVLmaps not only being sorted but also balanced results with a very quick binary search.



This graph shows the performance speed of the find range function. Similarly to the previous graphs, this also shows how many problems keeping a tree balanced can solve. BinSearchMap also has the potential to need to traverse a linked list in order to find each of the keys in the range, where the AVLmap has a very efficient and cleaver algorithm in order to find them.



This graph shows the speed for the sorted keys function. This graph shows how when it comes to returning a sorted list of the keys, a sorted map has much better of a result than a balanced map. AVLMap takes both of these ideas and is a sorted balanced map!



This final graph shows the basic stats for and avl. This graph just blatantly shows how a balanced tree is much smaller than an unbalanced tree. Since a balanced tree fills up as much as it can, it is much more compressed than a BST.

Table:

operation	ArrayMap	LinkedMap	BinSearchMap	HashMap	BSTMap	AVLMap
insert	1	1	nlogn	1	N	Logn
erase	N	N^2	Nlogn	1	N	Logn
contains	N	N^2	Nlogn	1	N	Logn
find keys	N	N^2	Nlogn	N	N	nLogn
sorted keys	N^2	N^2	n	n	N^2	nlogn

Issues I faced:

The main issue that I faced was in erase and rebalance. The main issue I had was that rebalance would end up trying to find the height of a nullptr which would end with a seg fault.