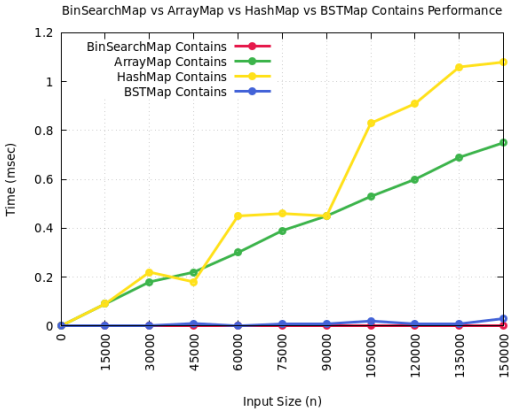
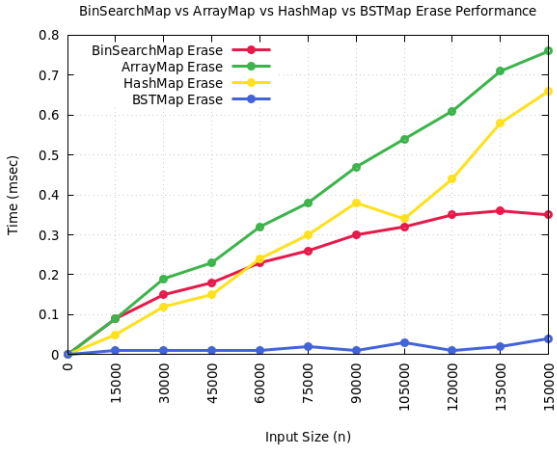
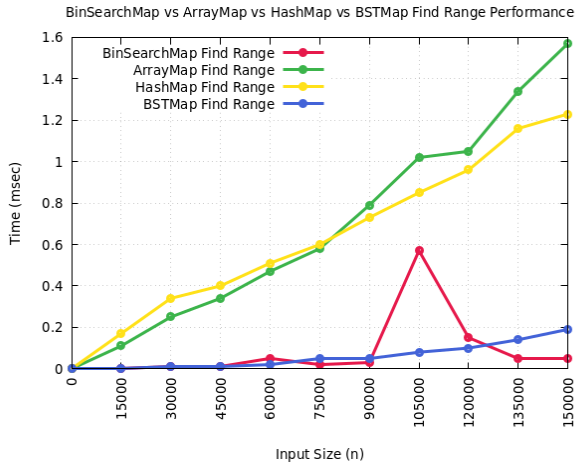
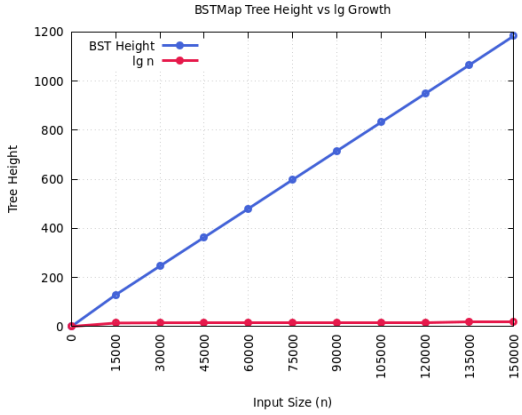
Colleen Lemak
Professor Bowers
CPSC 223
2 December 2021

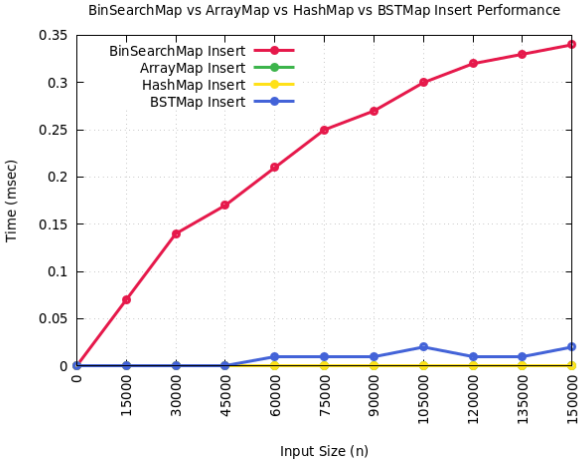
hw7_write_up

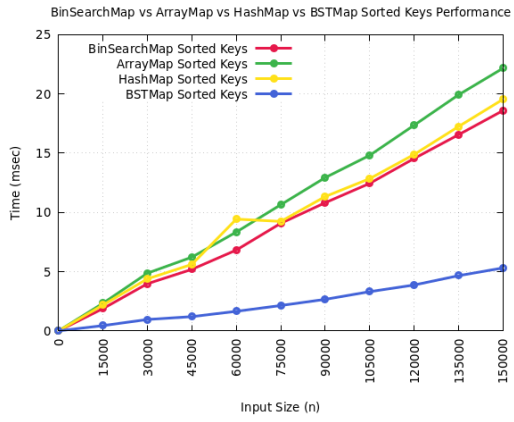












In insert(), BinSearchMap is consistently costly as n increases as expected, and HashMap and BSTMap are quite quick because unlike BinSearch, they can simply make a new node and attach it to the table or tree. For erase(), BSTMap performs very well because it is able to recursively track down the node to delete and then do pointer changes to hook nodes up correctly. However, with a tree, it can look more like a linked-list in the worst case, so these operations could take O(n). Additionally, BSTMap is quick in contains() because it takes advantage of the structure of the tree. Because the left side of the node should be less than the parent and the right should be greater than the parent, it is much easier to traverse through the tree assuming it is balanced and not in the worst case possible.

Operation	ArrayMap	LinkedMap	BinSearch	HashMap	BSTMap
insert	O(1)	O(1)	O(n)	O(1)	O(n)
erase	O(n)	O(n^2)	O(n)	O(1)	O(n)
contains	O(n)	O(n^2)	O(logn)	O(1)	O(logn)
find_keys	O(n)	O(n^2)	O(n)	O(n)	O(n)
sorted_keys	O(n^2)	O(nlogn)	O(n)	O(nlogn)	O(n)

Erase() was tricky, but after splitting it into various cases, I got it working. Printing in an abstract structure sometimes is difficult, but it wasn't much of a problem for debugging this program.