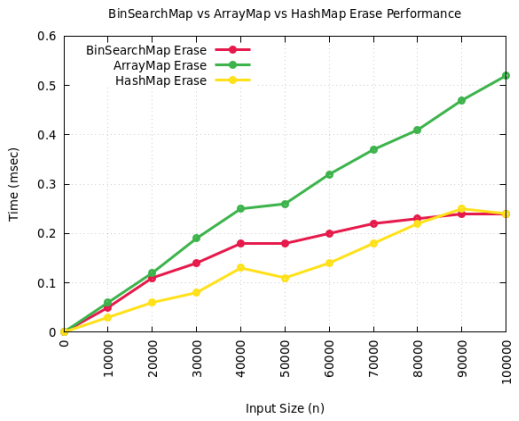
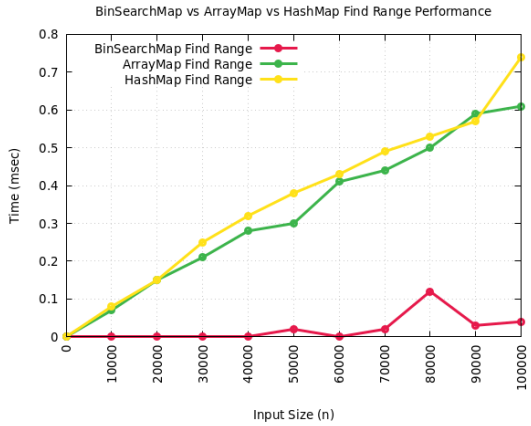
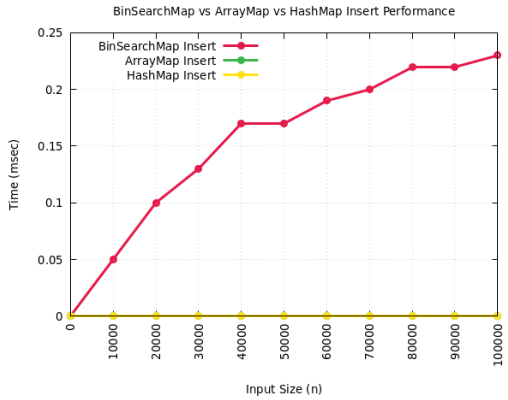
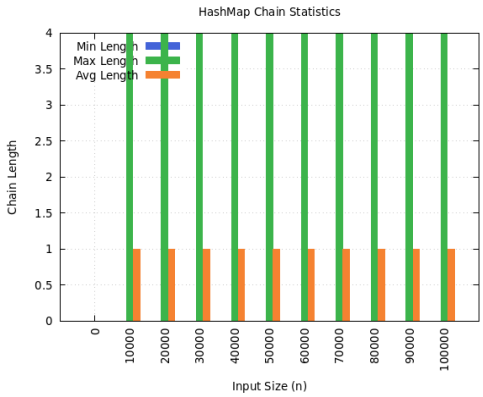
Colleen Lemak
Professor Bowers
CPSC 223
16 November 2021

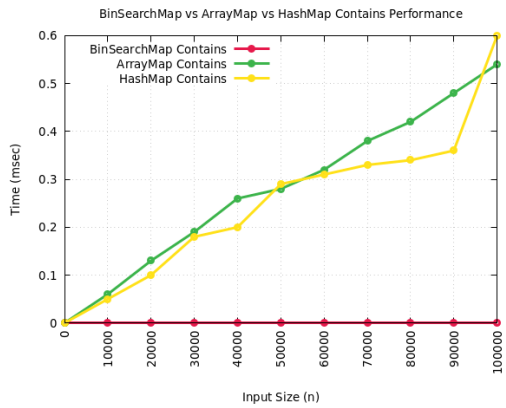
# Hw6_write_up



Using these graphs to interpret, we can see that using this K-V map, there is improved performance for contains, erase, insert, and operator[] and fetching in general. HashMap accessing is quite fast O(1) because we can hash right to the index and perform an operation or access there. The only one that is more expensive than O(1) is when you have to traverse through the whole or partial list like find keys or sorted keys. The HashMap chain statistics are interesting because of the ability to easily loop through to find out statistics about the table. With an increasing input size, HashMap typically is faster and more efficient than previously implemented maps like array and binsearch. Hashing straight to the value improves the time cost and makes for a better program assuming a good hash implementation.

Map Implementations

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

Once I understood the layout and concepts more, I was able to write functions much easier by drawing it out and tracing through the function's purpose.
I ran valgrind often, but although my tests passed, there was a memory leak in my resize and rehash function that I couldn't track down.