Data Structures Lab 4

Joshua Marple

September 2014

Organization

My experiment was very simply organized. First, I generate $K * LoadFactor$ random elements, storing these in an array. Next, I load each of these arrays into each of the data structures we are interested in testing- first into open hash, then linear closed hash, then finally quadratic closed hash. Before loading these into the data structures though, I start a timer. This timer is stopped after each of the data structures and the time of execution is stored in a separate array. This provides reasonable timing of loading the set of data.

Data Generation

Our data was generated using the functions `rand()` and `srand()`. `srand()` was used to ensure that the starting value was different on each run. Since we stored all the random values to an array, we know that each of the data structures was tested on the same data set as the others. Also, since it was stored in an array, we guaranteed constant access time.

Summary

My results are included in table 1.

Load Factor	0.2	0.3	0.4	0.5
Open Hash Time	0.0326	0.049	0.069	0.089
Linear Closed Time	0.0225	0.0351	0.0492	0.067
Quadratic Closed Time	0.0244	0.0365	0.055	0.071

Table 1: Results of the experiment (averages)

Observation and Conclusion

These results mostly fit what we would expect. The first result, most easily seen, is that as the load factor increases, the time it takes to load the data into each of the data structures also increases. Furthermore, it is fairly evident that open hashing works the quickest, then linear closed, then quadratic closed. This is due to the fact that open hashing does not have to handle collision resolution, only needing to append the next element into the bucket. This makes it very simple to add more data. However, this time would be worse if we tried to search for an element or to delete an element, as you would have to search the bucket after hashing the value.

There are some things to keep in mind though about these results. First, these results are not perfect. The CPU may cause certain data structures to appear better than others due to using caching or prioritizing the thread at that time higher than at other times, or similar such mechanisms. These results then should be taken with a grain of salt.