Christian Shadis

CS-242 Project 5

Prof Mahadev

Due 12/6/19

**Hashing Discussion Log**

**Progress**:

I expected to spend about three hours on this project because I heard it was relatively easy compared to the other projects we’ve done. I started coding Tuesday 12/3, finished initial coding and began debugging Wednesday 12/4, and finished debugging Thursday 12/5. It took me about six total hours, about double what I expected.

**Collaborations**:

I workshopped logic with Sebastian Serafin, posted a question in Piazza, and asked Professor Mahadev for some insight on possible causes of a particularly frustrating bug.

**Supplementary** **Resources**:

None

**Challenges / new things learned:**

As previously mentioned, I had a bug which was caused by counting deleted dataTable checks as comparisons. This was particularly frustrating, and I spent about two hours trying to debug just that. Besides that, I had a bit of a struggle implementing the comparison counts for both the Linear Probing and the Quadratic Residue Search. Since I had forgotten what we learned in class about hashing from waiting too long to start the project, I had to re-teach myself everything and figure out by trial-and-error how to go about coding the hashing implementations.

Summary

By far, the open hashing was the easiest to code, and wound up using the least amount of comparisons for searches and deletions. This seems to be the ‘best’ implementation of hashing. QRS and Linear probing did very well too, and I was surprised at the tiny amount of comparisons needed for such a large data set. The collision counts were much higher than I expected, though it doesn’t seem that collisions are a big time-waster when running the program. All things considered, this seems to be an excellent data structure in terms of running time. Having such a large table probably uses a lot of memory, but the efficiency is excellent. It seemed to outperform BST’s in both simplicity of code and comparison counts. BST’s go through a hierarchy of comparisons, whereas hashing takes, on average, between 0 and 2 before reaching the desired conclusion. This was my favorite data structure to code.