Cuckoo Hashing: +1 Element

Group #7
Matt Johnerson, Michael Shihrer, Bradley White

November 9, 2016

The cuckoo hashing algorithm was developed and tested during the early 2000's, using equipment that would be considered severely outdated by today's standards[1]. Pagh and Rodler used their Linux PC, sporting an 800 MHz processor with 256 MB RAM, to test cuckoo hashing against three common hashing algorithms of their time: Chained hashing, linear probing, and double hashing. They performed a variety of hashing operations (insert, delete, etc.) using each of the previously mentioned algorithms and measured each result by the number of clock cycles needed to perform the operation.

Therefore, the '+1 element' to our project will be to implement Cuckoo Hashing on a modern computer and compare common hashing operations between cuckoo hashing, perfect hashing and the methods presented in [2]. The method would consist of h hashing functions that place elements into "bins" of size 2^n array cells, such that $0 \leq n \leq 3$. A perfect hash function for a set S is a hash function that maps distinct elements in S to a set of integers, with no collisions[3].

We chose to compare cuckoo hashing to different algorithms than those explored in [1] because we felt that it would be more interesting and useful to expand on the results and compare to a recent hashing algorithm that was derived from cuckoo hashing. Our implementation will be in Python and time taken for operations will be recorded for various load factors of the hash table.

References

[1] Pagh and, Rodler. *Cuckoo Hashing*. 2001.

[2] Erlingsson, Manasse, and Mcsherry. *A Cool and Practical Alternative to Traditional Hash Tables*. 2006.

[3] `https://en.wikipedia.org/wiki/Perfect_hash_function`