Analysis of Separate Chaining and Quadratic Probing in Hash Tables

by Isaac Lee

Abstract:
An analysis of the properties of hash tables that use either a separate chaining or quadratic probing algorithms for inserting, finding, and removing their hash elements. 5 hash tables were made for each using the same hash function and hash key. 3 more tables were made, one with different hash function, one with a different hash key, and one with both. The number of reads (Times an element in the hash table is accessed) were recorded and written to a .csv file for analysis.

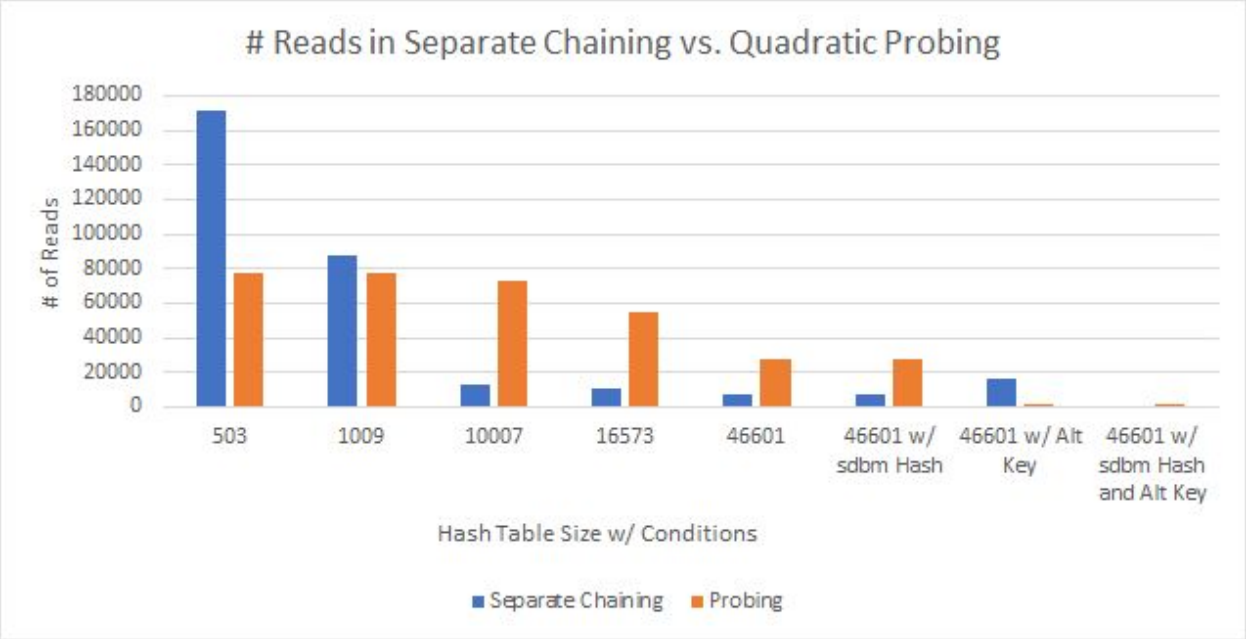
Data Set Attributes:

1. Rank - The rank of the video game based on sales number (Rank 1 ==> most sales)
2. Name - The name of the video game
3. Platform - The platform/console the video game was released on
4. Year - The year of the video game's release
5. Genre - The genre of the video game (Platformer, Simulation, Racing, Fighting, etc.)
6. Publisher - The publisher of the video game (Nintendo, Sony, Microsoft, etc.)
7. NA Sales - The total number of sales in North America
8. EU Sales - The total number of sales in Europe
9. JP Sales - The total number of sales in Japan
10. Other Sales - The total number of sales in all other regions
11. Global Sales - The total number of sales worldwide

Data Source:
This data was downloaded from https://www.kaggle.com/gregorut/videogamesales. I chose this data because I enjoy video games far more than I should, to the extent where I now run a gaming club and am considering a career in game development. This just seemed like it made sense to use.

Entry Order by Default:
Entries are ordered from the video game with the highest global sales (Rank 1), to the video game with the lowest global sales (Rank 16599).

Data/Graphs:
The following bar graph shows the number of reads for each hashing algorithm for each of the 5 differing sizes, as well as for the hashing tables that different conditions.



Analysis: As the hash table size increases, the number of separate chaining reads dramatically, almost in an exponentially decreasing manner, while the number of probing reads decreases steadily, in somewhat of a linear manner. Separate chaining shows to be more efficient with bigger hash tables, having both taken less time and less reads to complete. As for with the other hash function (I chose sdbm hash) and an alternate key (using genre instead of name), the number of reads for both algorithms is relatively similar for just the change in hash. With an alternate key, separate chaining becomes less efficient, while quadratic probing becomes more efficient. With both a different hash function and key, both algorithms become astonishingly efficient, taking less than 100 reads to complete. This may or may not be the result of some unknown error. Ultimately, the data shows that a large-size separate chaining hash table is optimal for my dataset.

Consider how removing and searching compares with inserting, in terms of number of hash elements read. Would they read more, less, or the same? With insertion, you only need to find an empty slot, which there are a lot of, whereas with search and removal, you need to find the exact thing you're looking for by reading every node. Therefore, search and removal requires more reads.

How often would each of the three operations (inserting, removing, searching) typically be used with your data set? Insertion would be used quite often, as my dataset has a lot of elements that can be sorted/keyed by a number of varying attributes (genre, sales, etc). Removal would not be used often. Usually it would be used to remove items of one attribute that have a certain quantitative or qualitative value in another attribute for the sake of sorting. Searching would be used often to look up which games have certain values for certain attributes given previous insertion and removal algorithms.

References

Dan ODan O 3, et al. “Why Isn't `Int Pow(Int Base, Int Exponent)` in the Standard C Libraries?”

Cplusplus.com, http://www.cplusplus.com/reference/cmath/pow/.

Stack Overflow, 1 Apr. 2010
https://stackoverflow.com/questions/2398442/why-isnt-int-powint-base-int-exponent-in-the-standard-c-libraries.Hardell, et al. “Simple Hash Functions.”

Stack Overflow, 1 Mar. 2011
https://stackoverflow.com/questions/14409466/simple-hash-functions.“Pow.”

The Prime Pages (Prime Number Research, Records and Resources),
https://primes.utm.edu/lists/small/10000.txt.