Jack Duncan
17 Feb. 2023

Project 2 Report

For Project 2, I created 3 implementations of hash functions: one using chaining to resolve collisions, another with quadratic probing, and the last with a second hash function. My spellCheck, convert, and hash methods remained the same while the constructors, add method, search method (called inArr), and various helper functions differed. The hash table that used chaining seemed to perform the best across the board while the double hash implementation generally performed the worst. I also tested for memory leaks, which did exist even though I used destructors; however, there were 0 errors from 0 contexts. In order to reach the misspelled count and time benchmarks, my spell check method made all uppercase words lower case, threw out strings with just numbers, got rid of all non '-' punctuation, and split string with 1 and 2 '-'s into 2 strings that were each checked.

My chaining hash table utilizes an array with a type linked list. The constructor constructs an array with a load factor of 1.0. The search (inArr) and add methods hash the input to a location, then at that location I call a search method for my link lists which linearly searches the list. The search time complexity is O(1 + n/m) because it is constant to reach the correct linked list, and n/m (num of elements / num of buckets) to traverse the linked list. The worst case space complexity is O(n + m), and this occurs if all items hash to the same bucket.

The other 2 implementations, quadratic probing and double hashing both construct an array with a load factor > 2.0, with a size of the next largest prime double the amount of elements in the list. The add and search methods hash to a location, then check if the location is empty or equals the new item. If empty, it adds the item to the slot. If not, then it probes. Quadratic probing probes quadratically while the step for double hashing is the result of hashing the item to a second function (I changed 37 to 31). The complexity for quadratic probing is the same as chaining with O(1 + n/m) while space complexity is O(n). For double hashing it is between O(1) and O(n), depending on the function. My function was not the most efficient, meaning it was not linear. The space complexity is the same as quadratic probing at O(n).

	Construction (ns)	Misspelled Count	Search (ns)
Chaining Simple	10,557	0	16,697
1	70,881,270	135	681,367
2	151,593,577	1807	40,843,154
Quad Probing Simple	12,630	0	17,872
1	73,351,108	135	565,267
2	154,086,364	1715	42,587,678
Double Hash Simple	11,050	0	24,083
1	102,599,269	135	639,835
2	211,779,330	1715	48,525,460