# Hash Table Analysis

Matthew Galitz

University of Virginia, Charlottesville, VA 22904

**Abstract.**
The purpose of the experiment is to optimize the Hash Table data structure and Word Solver algorithm using the Hash Table. Various strategies are compared to determine the one that is optimal.

## 1 Introduction

### 1.1 Problem

The problem is to determine the optimal word solving algorithm using the Hash Table data structure.

### 1.2 Description

The Hash Table is a data structure that maps keys to values. Objects are stored and retrieved using the hashCode() method. It features constant lookup times on average. Two strategies were employed to avoid address the occurrence of collisions: separate chaining and linear probing. The Hash Table was utilized to solve a word search puzzle. The strategy employed to find words was to store words from a dictionary into a hash table, loop through every row and column of a 50x50 word grid, and search for words in directions, north, south, west, east, northwest, northeast, southwest, southeast, of lengths 3 to 23 characters. If a word appears in the hash table, it is considered a found word. The word solver is also tested using separate chaining and linear probing.

## 2 Methods

Each hash table is made insert 1,000 items of random key and value for 5 trials each. The number of collisions is counted for each trial. The word solver is solved using hash tables of two different open addressing strategies. The time to solve the word puzzle is record for 5 trials on a 50x50 grid.

**Trials Run for Each Experiment**

| | # of Trials |
|---|---|
| **Hash Table (SC)** | 5 |
| **Hash Table (LP)** | 5 |
| **Word Solver (SC)** | 5 |
| **Word Solver (LP)** | 5 |

The average of the data for each trial is calculated using the following equation

$$\boldsymbol{A = \frac{\sum_{n=1}^{N} x_n}{N}}$$

Where $x_n$ is the time/collisions for trial $n$, $N$ is the total number of trials, and $A$ is the average time/collisions

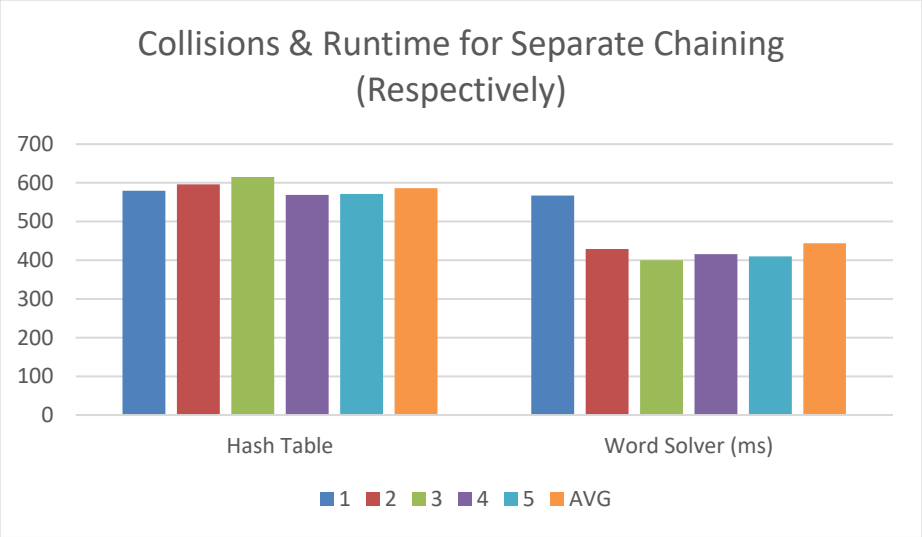
# 3 Results

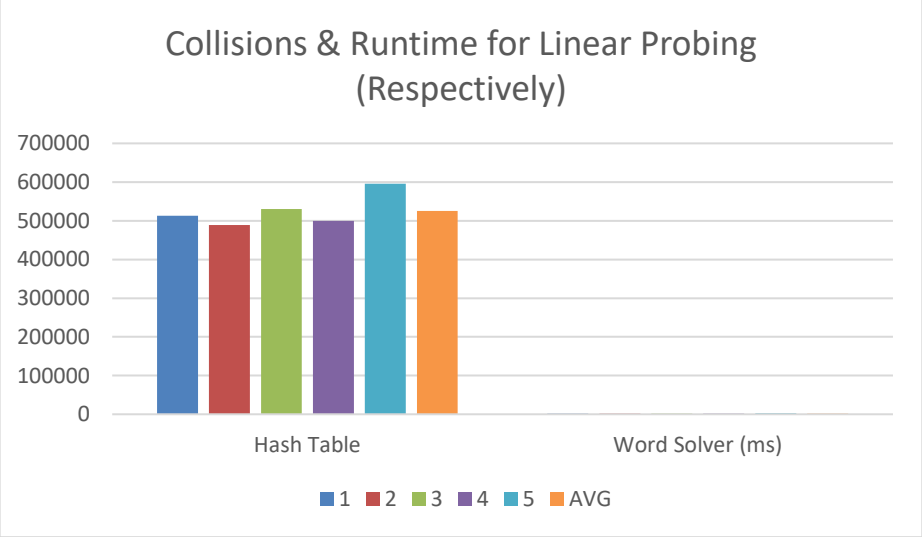
**Separate Chaining Collisions & Word Solver Runtime**

| | 1 | 2 | 3 | 4 | 5 | AVG |
|---|---|---|---|---|---|---|
| **Hash Table** | 580 | 596 | 615 | 569 | 571 | 586 |
| **Word Solver (ms)** | 567 | 429 | 400 | 416 | 410 | 444 |

**Linear Probing Collisions & Word Solver Runtime**

| | 1 | 2 | 3 | 4 | 5 | AVG |
|---|---|---|---|---|---|---|
| **Hash Table** | 512844 | 489002 | 529982 | 499892 | 595347 | 525413 |
| **Word Solver (ms)** | 2048 | 1729 | 1953 | 2184 | 2489 | 2081 |



## 4 Conclusion

The data indicates that the separate chaining strategy significantly reduces the number of collisions when compared to the linear probing strategy for inserting 1,000 items into a hash table. The data also shows that separate chaining is the more effective open addressing strategy to use for solving the word search of dimensions 50x50.

It is possible an error was made when counting the collisions. It is unlikely that one strategy is more effective to that degree. The code should be reviewed to verify this concern.