Assignment #6 – Questions

Keeley Abbott
Theodore Duchow-Pressley

May 29, 2015

1 Give an example of two words that would hash to the same value using `stringHash1()` but would not using `stringHash2()`.

Because `stringHash1()` is just a sum of the values of the characters in the string that is input, any string that has the same set of characters would end up having the same value. For example "tame" and "team" have the same hash in `stringHash1()`, but not in `stringHash2()`.

2 Why does the above make `stringHash2()` superior to `stringHash1()`?

As long as strings are limited to ASCII characters, there will always be a unique result for each unique string using `stringHash2()` (depending on the offset). The offset of each ASCII value of each character relative to its position in the string, ensures this uniqueness.

3 When you run your program on the same input file but one run using `stringHash1()` and on the other run using `stringHash2()`. Is it possible for your `size()` function to return different values?

No, because `size()` is returning the number of hashLinks in the table, which should be directly equivalent to the number of unique keys in the table. So, the hash function and size of table do not matter, because there should not be any variance when using the same word set.

4 When you run your program on the same input file using `stringHash1()` on one run and using `stringHash2()` on another, is it possible for your `tableLoad()` function to return different values?

No, because `tableLoad()` is returning the number of hashLinks divided by the number of buckets. Because this only relies on two things – (1) the size of the hash table and (2) the number of unique entries in the input file – this should not change.

5 When you run your program on the same input file with one run using `stringHash1()` and the other run using `stringHash2()`, is it possible for your `emptyBuckets()` function to return different values?

Yes, because different hash functions assign input keys to different buckets. So, even with the same table size and same number of unique entries, it is entirely possible to have differing numbers of empty buckets – even with the same input.

6 Is there any difference in the number of 'empty buckets' when you change the table size from an even number, like 1000 to a prime like 997?

Yes, because the table size is used in the modulo operator in the hashing functions. By changing the table size to a prime number, the chance of collision is decreased by eliminating possible common factors of the modulo operator – these collisions would result in assignment to the same bucket.

7 Using the timing code provided to you, run you code on different size hash tables. How does changing the hash table size affect your performance?

Increasing the table size directly correlates to increases in performance of `hashMap()`. This is because having fewer elements in each bucket means searching for a key can be accomplished in closer to constant time than linear time.
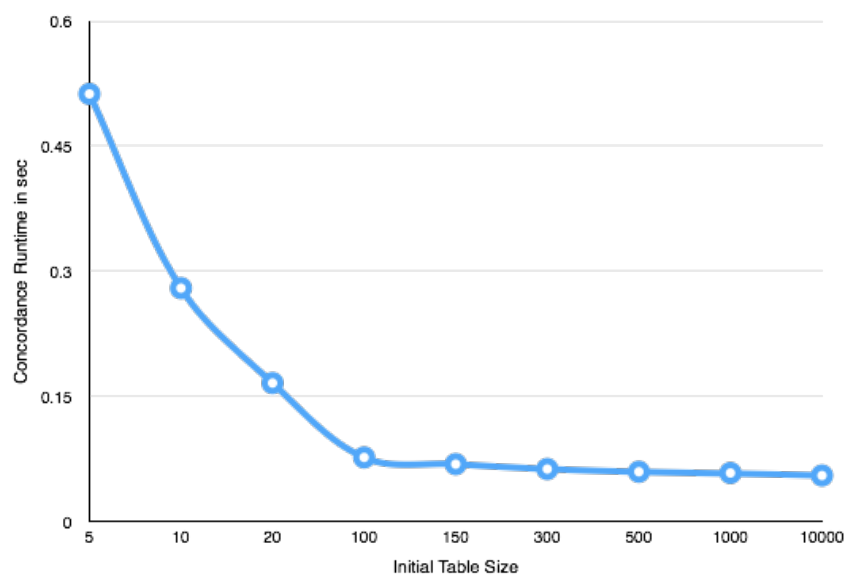

Figure 1: *Question 7 – Hash Table Sizing Performance*