Brandon Swanson
Assignment #6 Answers

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

Any anagrams such as the words in input3.txt

tea, ate, eat

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

Because hash 1 is simply the sum of the byte values (in ASCII) of the letter, so there are only C(26,r) possible hashes whereas has 2 has P(26,r) possibilities. This is due to the index multiplication causing the position of the letters to make a difference

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, there would still be the same amount of items in the map, they just may have a different distribution of buckets, i.e. the empty to utilized bucket ratio

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

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, if one of the hash functions does a better job of evenly distributing the returned and adjusted (mod) index across the size of the array, then one hash may have a less empty buckets wich would lead to better performance

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 ?

the theory for this is that having a prime numbered size will result in a better distribution of hashing indexes after performing the mod operation and should therefore result in less empty buckets. For my first test with input2.txt this did not prove to be the case. Below are the results

TableSize=1000 (originally)	TableSize=997 (originally)
concordance ran in 0.007852 seconds Table emptyBuckets = 1391 Table count = 908 Table capacity = 2000 Table load = 0.454000	concordance ran in 0.007860 seconds Table emptyBuckets = 1406 Table count = 908 Table capacity = 1994 Table load = 0.455366

I ran the test again this time changing the load factor to prevent a resize, the results are only minorly different.

TableSize=1000 (originally)	TableSize=997 (originally)
concordance ran in 0.006436 seconds Table emptyBuckets = 484 Table count = 908 Table capacity = 1000 Table load = 0.908000	concordance ran in 0.002765 seconds Table emptyBuckets = 483 Table count = 908 Table capacity = 997 Table load = 0.910732

7. Using the timing code provided to you. Run you code on different size hash tables. How does affecting the hash table size change your performance?

Starting with a much smaller size will of course lead to more re-hashing and moving values to the new table. So starting with a large capacity can reduce the time but if the starting size was too large it would be wasteful on memory. This is why some higher level language implementation of containers of this type have a capacity hint on initialization.