Artur Janowiec
Tom Plano
Cliff Anderson
William Lawrence
Based on your results:

a. What are your observations about the relationship between file size and hash table size?

- 12361 items for the pokemon file. The file size is 521 KB.
- 18 items for the sources file. The file size is 1 KB.
- 119201 items for the American English file. The file size is 1236 KB.

So the number of items rose exponentially with file size, but correlation is not causation.

b. Do you think there is a universally ideal hash table size for this problem? If not, what do you think the hash table size should depend on?

The hash table size should depend on what the application needs.
If it works and you understand the implementation, don't try to make it better unless your application needs it. There are diminishing rewards at a certain point.

c. What should be the correlation between word frequency and chain length? How can we influence chain length in this respect?

The more unique words, the higher possibility of the chain length being larger for a single index spot in the array. However, if the key is the same, only the value will change because we increment each time the key appears.

d. What is your observation about how well disbursed the words are in the hash table (i.e., are most of the words hashed to just a few locations, or do most of the buckets have words)? Did varying the hash table size have an effect on disbursement of words? If so, in what way?

Well for the pokemon file, there is a huge cluster from buckets 50 to 300. After that, it starts to trail off, and each bucket only has around 1 or 0 as the value.

e. What might be a better hash code, or what made your hash code work?

A better hash code would probably be one where it looks like this, which is from the data structure book.

$$u_0 g^{n-1} + u_1 g^{n-2} + \ldots + u_{n-2} g + u_{n-1}$$

In this case, g is a constant, u is the unicode character, and n is the position of the character. By doing this, you can diversify the hash indexes better, and prevent really big buckets.

If you worked in pairs:

o How did you “divide up” the work so that each student still met the objectives for the assignment (i.e., learned, understood and applied the concepts).

-We all worked on different parts of the HashMap because the lab focused more on the data structure instead of the application.

o What was your contribution?

- I (Artur) worked on the creating the two iterators, which come from the values() and keySet() methods. I also tested to see if they worked.
- I (Will) wrote code for the application and did the presentation
- Cliff and Tom did the ADT

o How did you coordinate code changes/testing?

- Github, and I also learned how to merge files. It’s actually not that hard to merge files as long as both members worked on separate parts of the same file.

o Other observations about working with a partner?

- Nada.

• Where did you have trouble with this assignment? How did you move forward? What topics still confuse you? • What did you learn from this assignment? (Please be specific)

- For me, I just had to figure out that the way iterators work in Hashmaps are a bit different. You have to use a separate data structure to store the values and keys, and then you get the iterators from those data structures.
- In other data structures such as the ArrayList, you simply had to get a single iterator.

Metrics

As the size of the array increased, percentage of bucket used dropped
Bucket average size also went down
Number of buckets increased drastically