Pre-class Work - Lesson “Hash functions and open addressing” CS110

Create a Google document to record all your answers. Be ready to paste a link to your Google document into a class poll (make sure the document’s settings are such that one can assess its content from the link).

1 The Python dictionary

[time estimate: 6 minutes]

A Python dictionary is essentially a hash table.

1. How does it resolve collisions?

   Python dictionaries use open addressing to resolve collisions.

2. The Python dictionary is initially allocated a small amount of memory, what happens when it runs out of memory?

   A python dictionary doubles its memory whenever its memory capacity is reached (doubles hash tables size).

2 The other type of dictionary

[time estimate: 10 minutes]

Let’s use a Python dictionary (hash table) as a literal dictionary. In the dictionary, let’s store all correctly spelled words (there are only roughly 100,000 English words in common use).

1. How would we use this to find incorrectly spelled words in a long string of text?

   Given that we have all correctly spelled words in a dictionary, we can try to search for a match between the words in the string and the words in the dictionary. We can try something like this: lookdict_string(word_from_string). The algorithm then uses the input in the hash function to try to find a match. If a match is found we return None or False, else we return an error stating that the word is not spelled correctly.

2. How does the computational complexity of a dictionary (hash table) compare to using a sorted array of all correctly spelled words?

   Using a hash table has an average complexity of Θ(1/(1-a)). Using a sorted array would require us to go through every single word in the array, resulting in a Θ(n) or Θ(n/2) but we take out the constant 2.

3. How might you use a list of common mis-spellings to automatically correct a mis-spelled word?

   We can create a hash table with common mis-spellings and try to match the word with one of the elements in the hash table. The main problem I see with this is that the hash table would be very large, and while it seems beneficial to find a way to group together mis-spellings of the same word, I can’t come up with any good way to do so.

3 Cuckoo hashing

[time estimate: 6 minutes]

For an introduction to cuckoo hashing please read the references given at the end of this question. Focusing on the Cuckoo hashing variant that uses only a single hash table, answer the following questions:

1. How does search in a cuckoo hash table work?

   Unlike previously covered hash algorithms, cuckoo hash does not keep on looking for other buckets when it encounters a matching bucket that is already storing a value. Since cuckoo hashing does not allow for chaining, it kicks out any element inside the bucket and takes its place. We then have to find a new bucket for the kicked out element.

2. What is the worst case complexity of search in a cuckoo hash table?

   Cuckoo hashing has a O(n), which would entail a situation in which we have to kick out an element to insert another one and then to find it a new bucket we have to kick out another element repeatedly until we have done this for all elements in the hash table. The final step would either entail putting an element into the sole remaining empty bucket, or expanding the hash table to make room for it.

3. [Optional] Implement the query function in Python.

4. [Optional] Implement the insert function in Python.

4 References

- Cuckoo hashing. (2016, September 10). In Wikipedia, The Free Encyclopedia. Retrieved 16:34, October 5, 2016, from https://en.wikipedia.org/w/index.php?title=Cuckoo_hashing&oldid=738697882
- Curtis Lassam (2016, September 10). Cube Drone - Cuckoo Hashing [Video file]. Retrieved from https://www.youtube.com/watch?v=HRzg0SzFLQQ