CSCI 3104 Spring 2023
Instructors: Ryan Layer and Chandra Kanth Nagesh

Homework 26

Due Date .... April 20, 2023
Name .... **Blake Raphael**
Student ID **109752312**
Collaborators .... **Alex Barry, Brody Cyphers, and Ben Kohav**

Contents

1 Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Gradescope page** only (linked from Canvas). Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (I agree to the above, Blake Raphael). □

3 Standard 26 – Hash Tables

3.1 Problem 1 (1 Point)

Consider the following hash function. Let U be the universe of strings composed of the characters from the alphabet $\Sigma = [\texttt{A}, \ldots, \texttt{Z}]$, and let the function $f(x_i)$ return the index of a letter $x_i \in \Sigma$, e.g., $f(\texttt{A}) = 1$ and $f(\texttt{Z}) = 26$. Finally, for an m-character string $x \in \Sigma^m$, define $h(x) = ([\sum_{i=1}^{m} f(x_i)] \mod \ell)$, where ℓ is the number of buckets in the hash table. That is, our hash function sums up the index values of the characters of a string x and maps that value onto one of the ℓ buckets.

Suppose this is going to be used to hash words from a large body of English text.

List at least 4 reasons why $h(x)$ is a bad hash function relative to the ideal behavior of uniform hashing.

Answer. $h(x)$ is a bad hash function relative to uniform hashing for the following reasons:

1: Similar words are mapped to nearby buckets, instead of randomly distributed. When changing one letter, a different magnitude assignment does not occur.

2: Different length strings can have the same value which leads to collision.

3: Different strings containing the same characters can be mapped to the same bucket which is also collision.

4: Strings will not be evenly distributed since some words are more common than others. Shorter words are used more often and they often have similar values which leads to clustering.

□

3.2 Problem 2 (1 Point)

Consider a chaining hash table A with b buckets that holds data from a fixed, finite universe U. Recall the definition of worst-case analysis, and consider starting with A empty and inserting n elements into A under the assumption that $|U| \leq bn$.

(a) What is the worst case for the number of elements that collide in a single bucket? Give an exact answer and justify it. **Do not assume the uniform hashing assumption for this question.**

Answer. If we are not assuming uniform hashing, then the worst case is that all n elements will collide when inserting them into hash table A. This is because this becomes a linked-list essentially which has insert time of $\Theta(n)$.

□

(b) Calculate the worst-case total cost of these n insertions into A, and give your answer as $\Theta(f(n))$ for a suitable function f. Justify your answer.

Answer. We know that the worst-case runtime for inserting an element into a hash table is $\Theta(1)$ and if we are inserting n items into the hash table we can say $f(1) \cdot n$ which gives us $f(n)$ so we have $\Theta(f(n))$.

□

(c) **For this part only, assume the uniform hashing assumption, and that the elements added were chosen uniformly at random from U.** After the n insertions, suppose that m `find` operations are performed. What is the total cost of these m find operations? Give your answer as $\Theta(f(n))$ for a suitable function f, and justify your answer.

Answer. The cost of the find operations will be the average number of elements examined during the search. When n elements are inserted into b buckets, the length of the chain will be $\alpha = \frac{n}{b}$, so each find operation will take $\Theta(\frac{n}{b})$. With m find operations the total time will be $m \cdot \Theta(\frac{n}{b}) = \Theta(\frac{mn}{b})$.

□

3.3 Problem 3 (2 Points)

Hash tables and balanced binary trees can be both be used to implement a dictionary data structure, which supports insertion, deletion, and lookup operations. In balanced binary trees containing n elements, the runtime of all operations is $\Theta(\log n)$.

For each of the following three scenarios, compare the average-case performance of a dictionary implemented with a hash table (which resolves collisions with chaining using doubly-linked lists) to a dictionary implemented with a balanced binary tree.

(a) A hash table with hash function $h_1(x) = 1$ for all keys x.

Answer. This would lead to every key being a collision which would be $\Theta(n)$ which is a greater runtime than the Binary Tree with runtime of $\Theta(\log n)$.

□

(b) A hash table with a hash function h_2 that satisfies the Simple Uniform Hashing Assumption, and where the number m of buckets is $\Theta(n)$.

Answer. Assuming Simple Uniform Hashing and the number m buckets has runtime of $\Theta(n)$ would have the average runtime of $\Theta(1 + \frac{n}{m}) = \Theta(1)$ which is a smaller average runtime than the Binary Tree $\Theta(\log n)$.

□

(c) A hash table with a hash function h_3 that satisfies the Simple Uniform Hashing Assumption, and where the number m of buckets is $\Theta(n^{3/4})$.

Answer. Assuming Simple Uniform Hashing and the number m buckets has runtime of $\Theta(n)$ would have the average runtime of $\Theta(1 + \frac{n}{n^{3/4}}) = \Theta(1 + n^{1/4})$ which is a larger average runtime than the Binary Tree $\Theta(\log n)$.

□