CSCI 3104 Fall 2021
Instructors: Profs. Chen and Layer

Problem Set 9

Due Date ..... April 8
Name ..... **Your Name**
Student ID ..... **Your Student ID**
Collaborators ..... **List Your Collaborators Here**

Contents

1 Instructions

- The solutions **must 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 Canvas page** only. 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.

2 Standard 24- Hash Tables

Problem 1. 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. For a hash function $h_1(x) = 1$, that is implemented with a doubly-Linked List with a head and tail pointer, with one element in every slot of the array. For the average-case run time of each operation of the has table, we have that:

- Insertion: We can insert at the head or tail of the linked list, thus operation takes $O(1)$
- Deletion: Assuming we want to delete a specific element, then the run-time of this operation is $O(n)$
- Lookup: Because we have to traverse through the linked list until we find a specific element, the run-time of this operation is $O(n)$

Comparing this hash table implementation with the definition of the run-time of a balanced binary tree, it follows that the balanced binary tree would be the better implementation in this scenario. □

(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. For the has function h_2, that is implemented with a doubly-Linked List, and satisfies the Uniform Hashing Assumption. Let $\alpha = n/m$ in which α is the load factor. For the average-case run time of each operation of the hash table, we have that:

- Insertion: We can insert at the head or tail of the linked list, thus operation takes $O(1)$
- Deletion: Let H be a hash table with underlying array $A[m]$ and suppose that for key k, $k \in H$, then the run time of deletion is $O(1 + \alpha)$
- Lookup: Since lookup follows the same method as deletion, then it follows that the run time for this is also $O(1 + \alpha)$

WE have that $m = n$, thus $\alpha = 1$ and the run time of the deletion and lookup is $O(1)$. Comparing this hash table implementation with the definition of the run time of a balanced binary tree, it follows that a hash table is the better implementation for $n > 10$. When $n = 10$, the run time of hash table and balanced binary tree would be the same and the implementation of a balanced binary tree would be more beneficial if $n < 10$. □

(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. For the hash function h_3, that is implemented with a doubly-Linked List with a head and tail pointer, and satisfies the Uniform Hashing Assumption. Let $\alpha = n/m$ in which α is the load factor. For the average-case run time of each operation of the hash table we have that.

- Insertion: We can insert at the head or tail of the linked list, thus operation takes $O(1)$
- Deletion: Let H be a hash table with underlying array $A[m]$ and suppose that for key k, $k \in H$, then the run time of deletion is $O(1 + \alpha)$
- Lookup: Since lookup follows the same method as deletion, then it follows that the run time for this is also $O(1 + \alpha)$

We have that $m = n^{3/4}$, thus $= n/n^{3/4}$ and the run-time of deletion and lookup is $O(1 + \frac{n}{n^{3/4}})$. Comparing this hash table implementation with the definition of the run time of a balanced binary tree it follows that the implementation of a balanced binary tree is better in this scenario. □

3 Standard 25- Doubling Lists and Amortized Analysis

Problem 2. Consider a hash table A that holds data from a fixed, finite universe U. If the hash table starts empty with $b = 10$ buckets and doubles its number of buckets whenever the load factor exceeds $\frac{1}{2}$, what is the load factor after adding $n = 36$ elements to the hash table?

Answer. When the load factor is above $\frac{1}{2}$ the number of buckets doubles. The load factor $\alpha = \frac{n}{m}$ when the elements are added $\alpha = \frac{36}{10} > \frac{1}{2}$, so the number of buckets is doubled so $\alpha = \frac{36}{20} > \frac{1}{2}$, so the number of buckets is doubled so $\alpha = \frac{36}{40} > \frac{1}{2}$, so the number of buckets is doubled so $\alpha = \frac{36}{80} < \frac{1}{2}$ which means the load factor after adding $n = 36$ elements to the has table is $\frac{36}{80} = 0.45$. □