CSCI 3104 Fall 2021
Instructors: Profs. Chen and Layer

Problem Set 9

Due Date ..... April 8
Name ..... **Chengming Li**
Student ID **109251991**
Collaborators ..... **N/A**

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 run time 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. **Insertion:**$\Theta(1)$.
Deletion and Lookup:$\Theta(n)$
Since the hash function will generate the same output as 1, everything will store in the same bucket, or same location. And we solve the hash table using chaining, so all elements are stored in the same linked list. Thus, the deletion and lookup time is $\Theta(n)$ given n element. And we always insert the element at the head of linked list, the insertion is $\Theta(1)$.

$$\lim_{n\to\infty}\left(\frac{log_2 n}{n}\right) = 0$$

So, the Deletion and Lookup run time of this hash table under $h_1(x) = 1$ is worse than a dictionary implemented with a balanced binary tree. And $\Theta(\log n) \in O(n)$
But, the insertion time $\Theta(1)$ is better than a dictionary implemented with a balanced binary tree. □

(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. **The load factor is** $\frac{n}{m} = n/n = 1$ due to the number m of buckets is $\Theta(n)$
Under the Simple Uniform Hashing Assumption:
Insertion: $\Theta(1)$, because we pre pend the current element to our linked list at every collision.
Deletion and Lookup: $\Theta(1 + \alpha = 1 + 1) = \Theta(1)$

$$\lim_{n\to\infty}\left(\frac{log_2 n}{2}\right) = \infty$$

So, the Deletion and Lookup run time of this hash table under h_2 is better than a dictionary implemented with a balanced binary tree. And $\Theta(1) \in O(\log n)$
And, the insertion time is better than a dictionary implemented with a balanced binary tree.$\Theta(1) \in O(\log n)$
□

(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. **The load factor is** $\frac{n}{m} = n/n^{3/4} = n^{1/4}$ due to the number m of buckets is $\Theta(n^{3/4})$
Under the Simple Uniform Hashing Assumption:
Insertion: $\Theta(1)$, because we pre pend the current element to our linked list at every collision.
Deletion and Lookup: $\Theta(1 + \alpha = 1 + n^{1/4})$

$$\begin{aligned}
\lim_{n\to\infty}\left(\frac{log_2 n}{1+n^{\frac{1}{4}}}\right) &= \frac{\frac{1}{nln2}}{\frac{1}{4}n^{-\frac{3}{4}}} \\
&= \lim_{n\to\infty}\frac{4n^{-\frac{1}{4}}}{ln2} \\
&= \lim_{n\to\infty}\frac{4}{n^{\frac{1}{4}}ln2} \\
&= 0
\end{aligned}$$

So, the run time of this hash table under h_3 is worse than a dictionary implemented with a balanced binary tree. And $\Theta(\log n) \in O(1+n^{1/4})$
But, the insertion time $\Theta(1)$ is better than a dictionary implemented with a balanced binary tree. □

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.

- Since it's asking that the load factor cannot exceed 1/2, which means whenever the n elements are greater than half of current m. i.e. n>m/2, we double its number of buckets.
- Now, the buckets, or m, is 10. Thus, the bucket will double its size when n > 5, i.e. n=6. Assuming we add the sixth element, we double the size of buckets to 20.
- Now, the buckets, or m, is 20. Thus, the bucket will double its size when n > 10, i.e. n=11. Assuming we add the 11th element, we double the size of buckets to 40.
- Now, the buckets, or m, is 40. Thus, the bucket will double its size when n > 20, i.e. n=21. Assuming we add the 21th element, we double the size of buckets to 80.
- Now, the buckets, or m, is 80. Thus, the bucket will double its size when n > 40, i.e. n=41. We can compute the load factor after adding n = 36 elements to the hash table, because 36 <41.

The load factor $n/m = 36/80 = 0.45$ □