CSCI 3104 Fall 2021
Instructors: Profs. Grochow and Waggoner

Problem Set 10

Due Date ..... TODO
Name ..... **john blackburn**
Student ID **jobl2177**
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.
- 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)

Problem 1.

- My submission is in my own words and reflects my understanding of the material.
- I have not collaborated with any other person.
- I have not posted to external services including, but not limited to Chegg, Discord, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (john blackburn). □

3 Standard 25- Hash Tables

3.1 Problem 2

Problem 2. 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. That is, for each of the three scenarios: describe clearly in terms of big-Oh notation the average-case runtime of the dictionary implemented with a hash table, and then clearly state whether the hash table implementation is better or the balance binary tree implementation is better, and why.

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

Answer. The binary tree implementation will be better in this case case because the worst case deletion and lookup times for the hash table implementation would be $O(n)$ compared to $\Theta(\log n)$ for the binary tree operations. If our hash function sends each value to bucket 1 we essentially now just have a doubly linked-list that we have to search each time we want any information from it. and in the worst case we would have to search the whole list to find what we want resulting in $O(n)$ run time for search and delete. The binary tree implementation is more effective in this case. $\log(n) \in O(n)$. It is a better solution because the worst case runtime of the binary tree implementation is faster than the worst case of the hash table implementation.

Limit Comparison Test work between the two implementations:

$\lim_{x\to\infty} \frac{n}{\log(n)}$

L'HOP:

$\lim_{x\to\infty} n = \infty$

Therefore we can see that $\log(n) \in O(n)$ and from that we can derive that the binary tree implementation is a better solution.

□

(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 $\Theta(n)$ (recall that n is the number of items).

Answer. Lookup and deletion times of a hash function that satisfies SUHA and resolves collisions with chaining all have run times of $\Theta(1 +$ (the number of items inserted/ the number of buckets)). In this case the number of items inserted is n and the number of buckets is $\Theta(n)$ so our runtime looks like $\Theta(1+(n/\Theta(n)))$ which can be simplified down to $\Theta(1 + n/n)$ which can be simplified to $\Theta(1)$ which is the best case runtime for a hash function and much faster than $\log(n)$. Therefore the hash function dictionary implementation is faster and more efficient than the balanced binary tree implementation for this case. It is a better solution because the worst case runtime of the hash table implementation is far faster than the worst case of the binary tree implementation.

Limit Comparison Test work between the two implementations:

$\lim_{x\to\infty} \frac{1}{\log(n)} = 0$

Therefore we know that $1 \in O(\log(n))$ and from that we know that the hash table implementation is a better solution in this case.

□

(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 $\Theta(\sqrt{n})$.

Answer. Lookup and deletion times of a hash function that satisfies SUHA and resolves collisions with chaining all have run times of $\Theta(1 +$ (the number of items inserted/ the number of buckets)) in this case the number of items inserted is n and the number of buckets is $\Theta(\sqrt{n})$ so our runtime looks like $\Theta(1 + (n/\Theta(\sqrt{n})) = \Theta(1+\sqrt{n})$. Now when comparing this to $\Theta(\log(n))$ we find that the binary tree implementation is more efficient in this case. It is a better solution because the worst case runtime of the binary tree implementation is faster than the worst case of the hash table implementation.

Limit Comparison Test work between the two implementations:

$\lim_{x\to\infty} \frac{1+\sqrt{n}}{\log(n)}$

L'HOP:

$\lim_{x\to\infty} \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n}}$

Simplifies to:

$\lim_{x\to\infty} \frac{\sqrt{n}}{2} = \infty$

Therefore we know that $\log(n) \in O(1 + \sqrt{n})$

and from that we know that the binary tree implementation is more efficient and a better solution in this case.

□