CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Chandra Kanth Nagesh

Midterm S26

Due Date ..... Saturday Nov 19, 2022 4pm MT
Name ..... **Tyler Huynh**
Student ID **109603994**
Quiz Code (enter in Canvas to get access to the LaTeX template) **ul0UsWwUAs**

Contents

Instructions **1**

Honor Code (Make Sure to Virtually Sign) **2**

26 Standard 26: Hash Tables **3**

Instructions

- You may either type your work using this template, or you may handwrite your work and embed it as an image in this template. **If you choose to handwrite your work, the image must be legible, and oriented so that we do not have to rotate our screens to grade your work.** We have included some helpful LaTeX commands for including and rotating images commented out near the end of the LaTeX template.
- You should submit your work through the **class Gradescope 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 **may not collaborate with other students**. **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, Discord, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code. Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign)

Problem HC.

- 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.

I agree to the above, Tyler Huynh. □

26 Standard 26: Hash Tables

Problem 26. Consider a hash table designed to store integers, using the hash function $h(k) = k \ mod \ 5$ for all keys k for a table of size 5. (Resolve collisions by chaining with a linked list.) You have three scenarios:

- Scenario 1: keys are randomly drawn from integers that are divisible by 5.
- Scenario 2: keys are randomly drawn from integers of the form $5m + 1$ where m is an integer.
- Scenario 3: keys are randomly drawn from *all* integers

Do the following.

(a) For which scenario does the hash function $h(k)$ perform better? Please **explain/justify** your answer.

Answer. Scenario 3, will perform better because for both scenario one and scenario two these two scenarios will always hash the keys to the same bucket such that, scenario 1 will always hash it to bucket 0 creating a collision and scenario 2 will always hash it to bucket 1 also creating a collision. Scenario 3 since we are drawing from all integers the likelihood of a collision happening is less than both scenario 1 and 2 because we can hash to buckets from 0 to 4. □

(b) In each of the three applications, does the hash function $h(k)$ satisfy the uniform hashing property? Please **explain/justify** your answer.

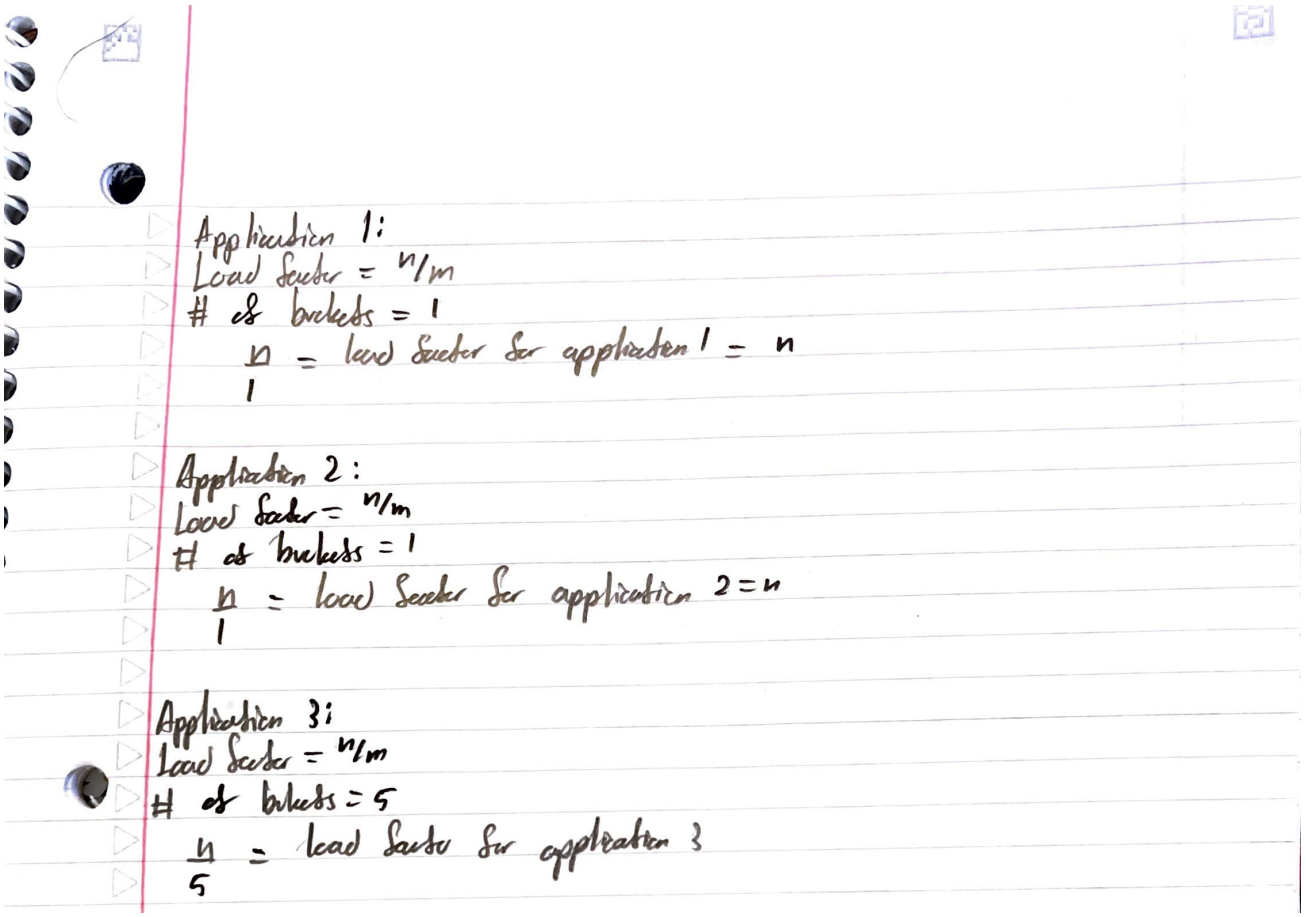
Answer. **Scenario 1:** Would not satisfy the uniform hashing property because it would satisfy the second condition where two keys k are independent, but it will fail the first one where the probability of hashing any key into any index should be $\frac{1}{5}$ however it is not as the probability of hashing the key into bucket 0 will always be 1, because the key will always be divisible by 5. The probability of hashing any other key into another would be 0. Thus it does not satisfy the uniform hashing property.

Scenario 2: Would not satisfy the uniform hashing property because it would satisfy the second condition where two keys k are independent, but it will fail the first one where the probability of hashing any key into any index should be $\frac{1}{5}$ however it is not as the probability of hashing the key into bucket 1 will always be 1, because the key will always have a remainder of 1 by the definition of the scenario and h(k). The probability of hashing any other key into another would be 0. Thus it does not satisfy the uniform hashing property.

Scenario 3: This scenario would satisfy the uniform hashing property because the second condition will be satisfied such that two keys within h(k) are independent from one another. This scenario would also satisfy the first condition because we are drawing the keys randomly from all integers, the probability of hashing the key into any of the 5 buckets will be $\frac{1}{5}$, where 5 represents m or the number of buckets. □

(c) Suppose you have n keys in total for each application. What is the resulting load factor for each application?

Answer.



□

(d) Suppose you have n keys in total for each application. What are the time complexities of the dictionary operations: add, delete, and find, respectively?

Answer. For all three applications the time complexity of the dictionary: adding will always be $\Theta(1)$ regardless if it satisfies the uniform hashing property or not because the hash function will always run in constant time when adding something to the hash table.

- **Application 1:**
  For this application because it does not satisfy the uniform hashing property the runtime for both delete and find using chaining to solve collisions in a linked list, respectively will be $\Theta(n)$.

- **Application 2:**
  For this application because it does not satisfy the uniform hashing property the runtime for both delete and find using chaining to solve collisions in a linked list, respectively will be $\Theta(n)$.

- **Application 3:**
  For this application since it does satisfy the uniform hashing property per it being able to hash values across all 5 buckets uniformly the runtime for delete and find will be $\Theta(1+\alpha)$, where α represents our load factor, our load factor by $\frac{n}{m}$ where n represents our keys and m represents the number of buckets we have which 5, such that:

  Deletion: $\Theta(1+\frac{n}{5})$
  Find: $\Theta(1+\frac{n}{5})$

□