Homework 9: Hashing

Instructor: Mehmet Emre

CS 32 Spring '22

Due: 5/11 12:30pm
Name & Perm #:

Homework buddy (leave blank if you worked alone):
Reading: DS 12.2

1

From DS (12.2): There is a technique known as "double hashing".

1. (4 pts) What is the problem that double hashing is designed to solve? **Briefly explain.**

2. (4 pts) How does double hashing solve that problem? **Briefly explain.**

2

From DS 12.2: The abstract data type (ADT) known as a "dictionary" provides a way to look up keys and find values. We define these abstract operations for the ADT:

lookup(key) returns either value, or an indication that the value is not present

remove(key) removes the item with that key (if it exists)

insert(key,value) inserts the new key, and the value (or replaces the value if it is already present)

We could implement a dictionary by having an array of (key,value) pairs, sorted by key. We could use binary search for lookup. We'd have to figure out a way to put new values into the sorted list, and remove values. Or we could use hashing, as described in DS 12.2 (question continues on the next page)

1. (2 pts) If we use binary search on a sorted array, what is the worst case time for lookup(key) in terms of big-O? (No explanation needed; just state the answer)

2. (4 pts) If we use binary search on a sorted array, what is the worst case big-O time for remove(key)? This time, *briefly* explain your answer.

3. (2 pts) If we use hash tables instead, what is the worst case big-O time for lookup(key)? Just state it.

4. (2 pts) Is the worst case for lookup(key) for hash tables better or worse than the binary search approach?

5. (2 pts) If we use hash tables instead, what is the average (expected) case big-O time for lookup(key) assuming no collisions? Just state it.

6. (2 pts) Is the average (expected) case for lookup(key) for hash tables better or worse than the binary search approach?