CS 61BL Summer 2019

Lab 13

July 22, 2019

Name: SID:

Please complete this worksheet during your lab, and turn it in to your TA by the end of your section. You are encouraged to work with your neighbors collaboratively.

Section Number: (01) (02) (03) (04) (05) (06) (07) (08) (09) (10) (11) (12)

1 Hash Functions

For the following implementations of the `Integer`'s `hashCode()` function, mark each as valid or invalid. If it is invalid, explain why. If it is valid, point out a flaw or disadvantage.

Note: The `Integer` class extends the `Number` class, a direct subclass of `Object`. The `Number` class' `hashCode()` method directly calls the `Object` class' `hashCode()` method.

1.1 `return -1;`

→ ○ Valid ○ Invalid

but 100% collision rate

1.2 `return intValue() * intValue();`

→ ○ Valid ○ Invalid

but an integer C will collide with −C

1.3 `return super.hashCode();`

○ Valid → ○ Invalid

will return some integer corresponding to the objects memory ∴ same integer by .equals(..) may return different hash

Validity of Hash functions

- two objects that are equal by .equals(..) method should hash to same value
- hash function should return same value every time it's called on the same object

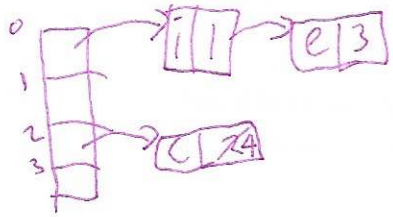
check for these

2 Ice Cream

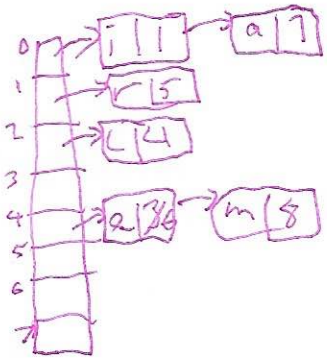
Assume we have a `HashMap<Character, Integer>` that starts with 4 buckets, resizes by doubling the number of buckets if the insert would cause the load factor to exceed 0.75, and implements external chaining using linked lists. Draw the `HashMap` as we've seen in lab after inserting: `('i', 1)`, `('c', 2)`, `('e', 3)`, `('c', 4)`, `('r', 5)`, `('e', 6)`, `('a', 7)`, `('m', 8)`. Determine `C`, the total number of hash collisions that occur, and `F`, the current load factor after all the insertions. Assume the hash code for `'a'` is 0, `'c'` is 2, `'e'` is 4, `'i'` is 8, `'m'` is 12, and `'r'` is 17.

C: 3

F: 6/8



resize



put

contains
- T → update value
- F → insert as new; resize if LF exceed

resize
- make new array
- repopulate it by hashing then mod/u to find new index

collision counter: ///

$F = \frac{N}{M} = \frac{\#\text{ of items}}{\#\text{ of buckets}}$

3 Modifications

For each of the following scenarios, mark **Always**, **Sometimes**, or **Never** and explain your reasoning.

(a) When you modify a key that has been inserted into a `HashMap`, will you be able to retrieve that entry again?

○ Always → ○ Sometimes ○ Never

if the new key hashes to the same value after and mod hash as the old key we can retrieve it but else we can't

(b) When you modify a value that has been inserted into a `HashMap`, will you be able to retrieve that entry again?

→ ○ Always ○ Sometimes ○ Never

key is used to find index, not value ∴ changing value doesn't affect lookup procedure

lookup procedure: hash key then mod hash value to find index