

CS151 Intro to Data Structures

Hashmaps

Announcements & Outline

Hw7 Released due Monday December 1st (After thanksgiving break)

Final Exam: Monday December 15th 9:30AM Park 159

Today:

- HashMap Review
- Handling Collisions
- HW7 discussion

Hash Map Review

Hash Map:

- Efficient data structure with constant time* access, insertion, and removal
- * assuming no collisions or expansions

Hash Function Review

Book's `AbstractHashMap` hash method uses:

$$h_1(k) = k.hashCode() \text{ // java memory address}$$
$$h_2(x) = ((ax + b) \% p) \% N$$
$$h = h2(h1(k))$$

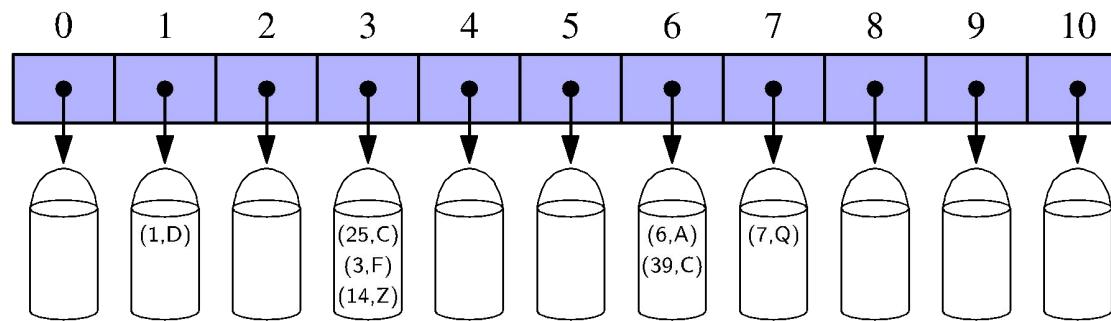
Performance Analysis

	ArrayMap	Collision Resistant Hash Map
get		
put		
remove		

Review: Handling Collisions

ChainHashMap:

- When more than one key hash to the same index, we have a bucket
- Each index holds a collection of entries

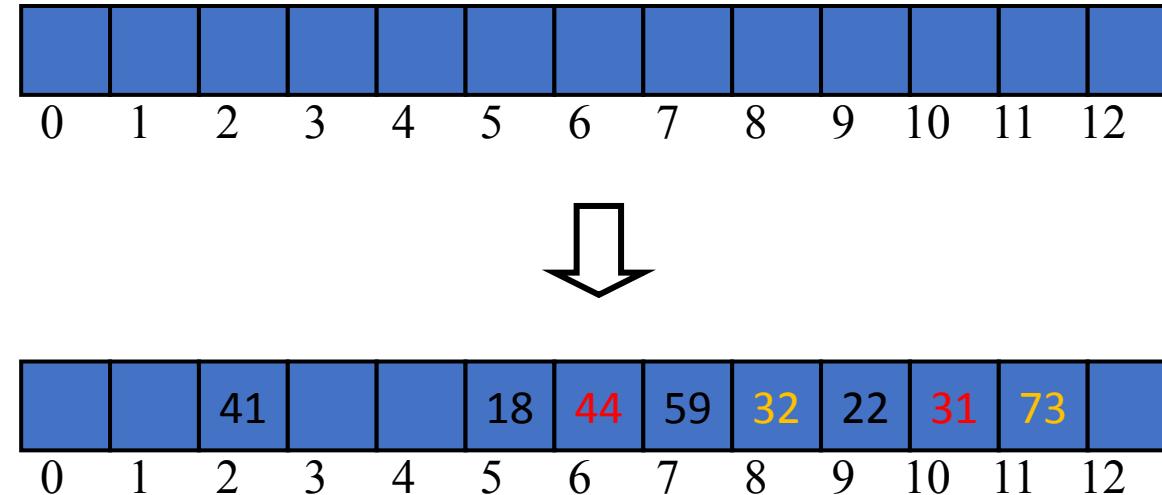


- Worst case:
 - all elements collide into the same bucket
 - $O(n)$ operations

Open Addressing and Probing

- Example: $h(x) = x \% 13$
- insert 18(5), 41(2), 22(9), **44(5)**, 59(7), **32(6)**, **31(5)**, **73(8)**

Keep “*probing*”
 $(h(k)+1)\%n$
 $(h(k)+2)\%n$
....
 $(h(k)+i)\%n$
until you find an
empty slot!



ProbeHashMap

Let's look at an implementation of ProbeHashMap

Open Addressing and Probing

Linear Probing (what we just saw):

- Keep “*probing*” until you find an empty slot
 - $(h(k)+1) \% n$
 - $(h(k)+2) \% n$
 -
 - $(h(k)+i) \% n$
- Colliding items cluster together – future collisions to cause a longer sequence of probes

Open Addressing and Probing

Quadratic Probing:

- Keep “*probing*” until you find an empty slot

$$(h(k)+f(1)) \% n$$

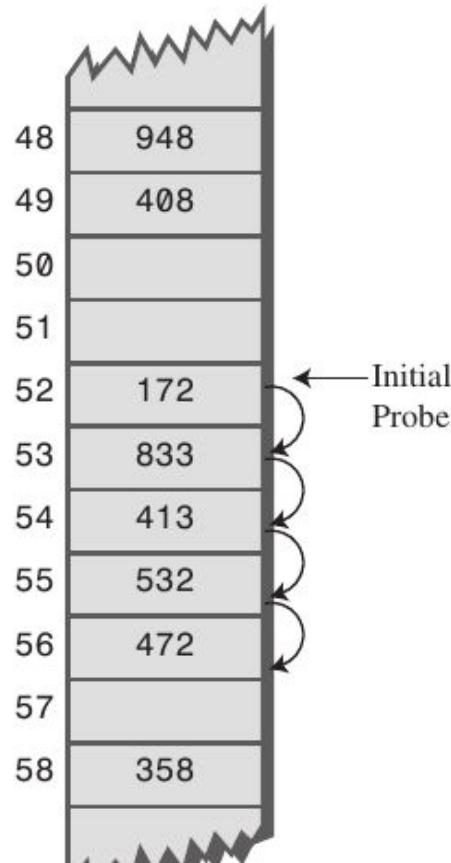
$$(h(k)+f(2)) \% n$$

....

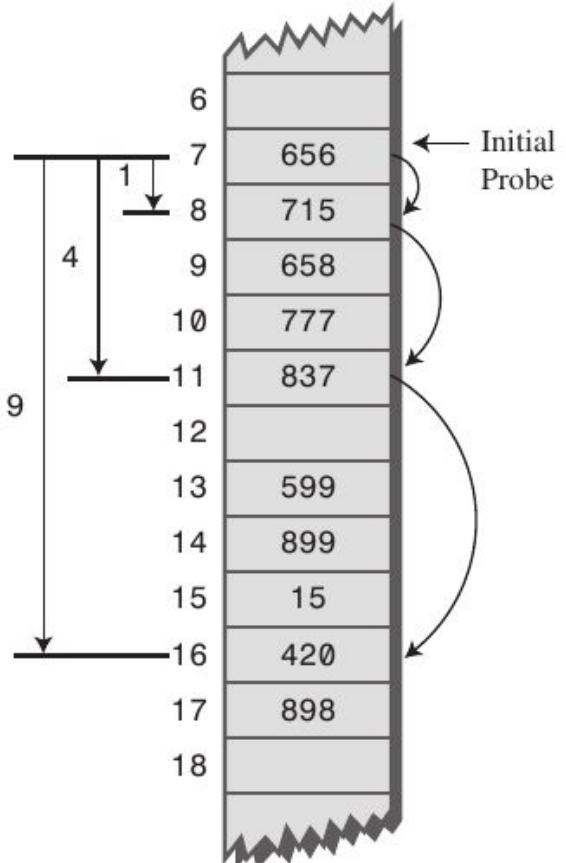
$$(h(k)+f(i)) \% n$$

where $f(i) = i^2$

Linear Probing vs Quadratic Probing



Linear Probing



Quadratic Probing

- Quadratic probing still creates large clusters!
- Unlike linear probing, they are clustered away from the initial hash position
- If the primary hash index is x , probes go to $x+1, x+4, x+9, x+16, x+25$ and so on, this results in ***Secondary Clustering***

Approach #3: Double Hashing

Let's try to avoid clustering.

To probe, let's use a **second hash function**

- Keep “*probing*” until you find an empty slot

$$(h(k) + f(1)) \% n$$

$$(h(k) + f(2)) \% n$$

....

$$(h(k) + f(i)) \% n$$

Where $f(i) = i * h'(k)$

Approach #3: Double Hashing

Keep “*probing*” until you find an empty slot

$$(h(k)+f(1)) \% n$$

$$(h(k)+f(2)) \% n$$

....

$$(h(k)+f(i)) \% n$$

Where $f(i) = i * h'(k)$

A common choice for $h'(k) = q - (k \% q)$
where q is prime and $< n$

Example

k	$h(k)$	$h'(k)$	Probes
18	5	3	5
41	2	1	2
22	9	6	9
44	5	5	5 10
59	7	4	7
32	6	3	6
31	5	4	5 9 0
73	8	4	8

- Insert 18, 41, 22, 44, 59, 32, 31, 73

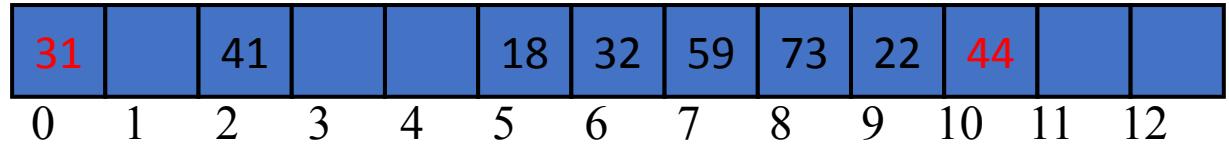
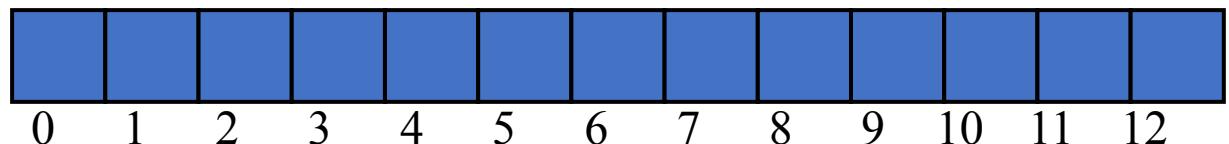
probe:

$$(h(k) + f(k)) \% n$$

$$h(k) = k \% 13$$

$$f(k) = i * h'(k)$$

$$h'(k) = 7 - k \% 7$$



Performance Analysis

	ChainHashMap Best Case	ChainHashMap Worst Case	ProbeHashMap Best Case	ProbeHashMap Worst Case
get				
put				
remove				

Which is better in practice?

Open Addressing vs Chaining

- Probing is significantly faster in practice
- locality of references – much faster to access a series of elements in an array than to follow the same number of pointers in a linked list

Performance Analysis

	ArrayMap	HashMap with good hashing and good probing
get		
put		
remove		

Performance of Hashtable

	array	linked list	BST (balanced)	HashTable
search	$O(n)$	$O(n)$	$O(\log n)$	$O(1)$
insert	$O(1)$ *	$O(1) / O(n)$	$O(\log n)$	$O(1)$
remove	$O(n)$	$O(1) / O(n)$	$O(\log n)$	$O(1)$

Load Factor

- HashMaps have an underlying array... what if it gets full?
 - For ChainHashMap collisions increase
 - For ProbeHashMap we need to resize!
- Load Factor = # of elements stored / capacity
- A common strategy is to resize the hash map when the load factor exceeds a predefined threshold (often 0.75)
 - tradeoff between memory and runtime

HW7 Discussion

Homework 7

- NYPD “Stop Question and Frisk” dataset
- How to work with large data

From Wikipedia, the free encyclopedia

A **Terry stop** in the United States allows the police to briefly **detain** a person based on **reasonable suspicion** of involvement in criminal activity.^{[1][2]}

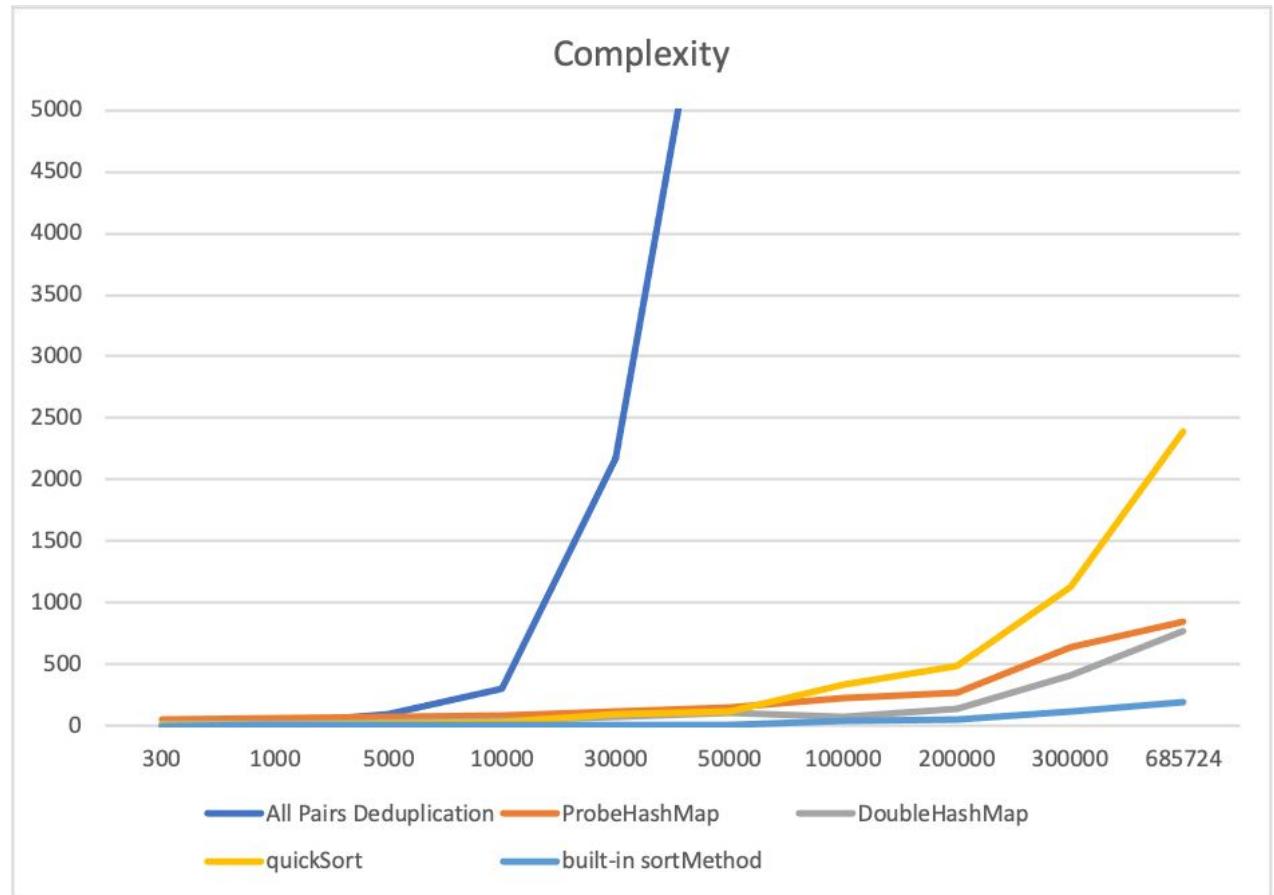
Reasonable suspicion is a lower standard than **probable cause** which is needed for **arrest**. When police stop and search a pedestrian, this is commonly known as a **stop and frisk**. When police stop an automobile, this is known as a **traffic stop**. If the police stop a motor vehicle on minor infringements in order to investigate other suspected criminal activity, this is known as a **pretextual stop**. Additional rules apply to stops that occur on a bus.^[3]

Homework 7

- How many times was the same person stopped for questioning?

Homework 7 Part 2: Complexity Analysis

- Line graph
- x axis: number of entries
- y axis: time in seconds



Summary

ChainHashMap - handles collisions by bucketing collisions in a Linked List

ProbeHashMap - handles collisions by finding the “next” open slot

1. Linear probe
2. Quadratic probe
3. Double Hash

Chain and Probe Hash Maps have equivalent runtime complexity (Big-O notation), but Probe is faster in practice