LEC 17

CSE 123

Hashing

BEFORE WE START

Talk to your neighbors:

Do you usually remember where you put things?

Instructors: Brett Wortzman
Miya Natsuhara

TAs:

Arohan	Neha	Rushil	Johnathan	Nicholas
Sean	Hayden	Srihari	Benoit	Isayiah
Audrey	Chris	Andras	Jessica	Kavya
Cynthia	Shreya	Kieran	Rohan	Eeshani
Amy	Packard	Cora	Dixon	Nichole
Trien	Lawrence	Liza	Helena	

Music: CSE 123 25wi Lecture Tunes

Questions during Class?

Raise hand or send here

sli.do #cse123

Announcements

- Creative Project 3 out, due ~~Wednesday, March 12~~ Friday, March 14 at 11:59pm
- Resubmission Cycle 6 due **tonight at 11:59pm**
  - ***P1***, C2, P2 eligible
- R7 / R-Gumball will open on Monday
  - ***all*** assignments will be eligible!
- Final Exam: Tuesday, March 18 at 12:30pm – 2:20pm
  - Left-handed desk request form, closes Tuesday, March 11
  - Details and resources posted later today
- Gumball & Gigi campus visit on Monday, March 17 12:00pm – 2:00pm

Data structures so far

- **Lists**
  - Maintain an ordered sequence of elements
  - Provides get(), add(), remove(), ...
  - Studied two implementations: ArrayIntList and LinkedIntList

- **Sets**
  - Maintain a collection of elements
  - Provides contains(), add(), remove(), ...
  - Implementations?

Set implementations

	ArraySet(?)	LinkedSet(?)	TreeSet	HashSet
contains()	O(n)	O(n)		
add()	O(n)	O(n)		
remove()	O(n)	O(n)		

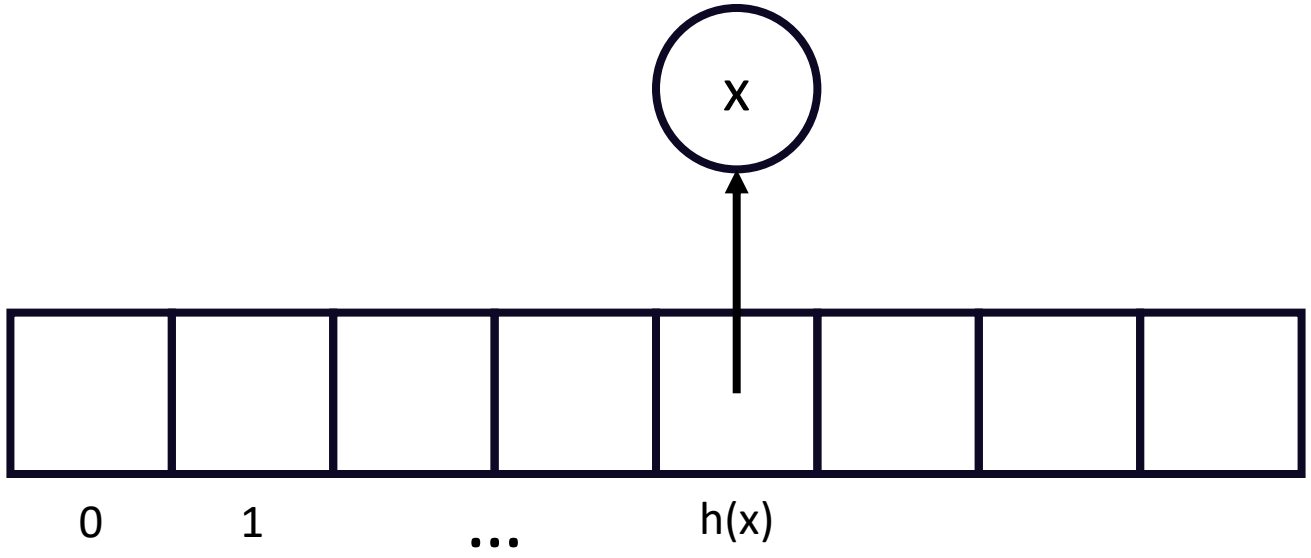
Set implementations

	ArraySet(?)	LinkedSet(?)	TreeSet	HashSet
contains()	O(n)	O(n)	O(log(n))*	
add()	O(n)	O(n)	O(log(n))*	
remove()	O(n)	O(n)	O(log(n))*	

* assuming tree is balanced

Hash Table

- Define a **hash function** h(x) that turns any value into an integer
  - Call this the values **hash code**
- Create a big array
- Store each value in the array at index h(x)



Set implementations

	ArraySet(?)	LinkedSet(?)	TreeSet	HashSet
contains()	O(n)	O(n)	O(log(n))*	
add()	O(n)	O(n)	O(log(n))*	
remove()	O(n)	O(n)	O(log(n))*	

* assuming tree is balanced

Set implementations

	ArraySet(?)	LinkedSet(?)	TreeSet	HashSet
contains()	O(n)	O(n)	O(log(n))*	O(1)**
add()	O(n)	O(n)	O(log(n))*	O(1)**
remove()	O(n)	O(n)	O(log(n))*	O(1)**

* assuming tree is balanced

** with some assumptions

What is Linear Probing?

A way to resolve collisions by adding the element in the next available spot

Regular Hash Function

h(x) = x % size

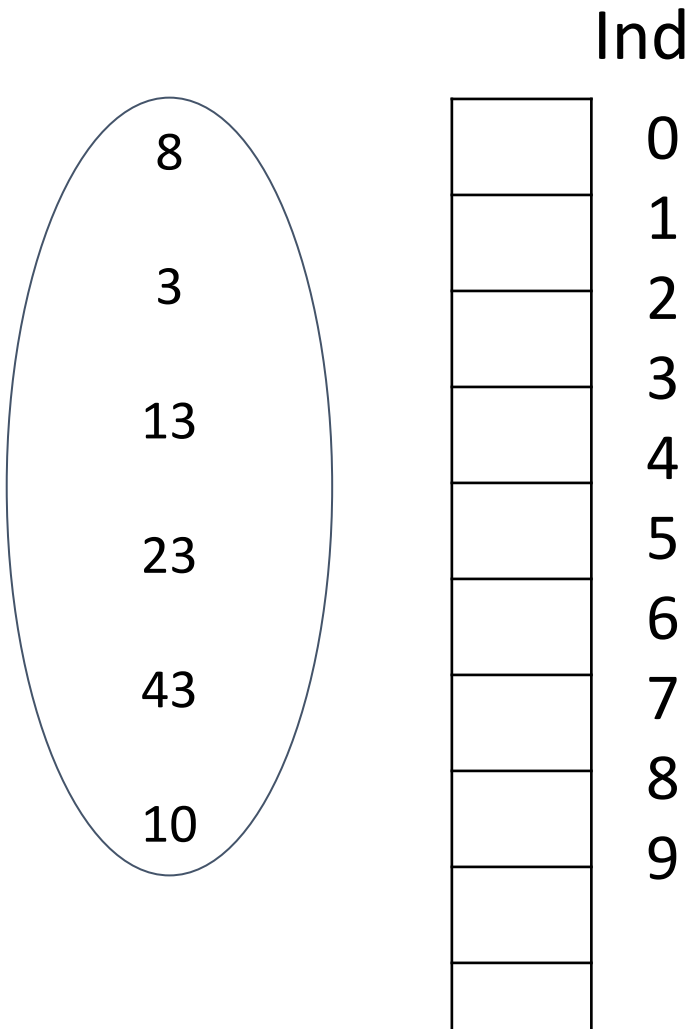
Hash Function (if collision)

h'(x) = [h(x) + f(i)] % size

f(i) = i

What is Linear Probing?

h(x) = x % size h'(x) = [h(x) + f(i)] % size f(i) = i



What is Quadratic Probing?

A way to resolve collisions by adding the element in the next available spot (quadratically)

Regular Hash Function

h(x) = x % size

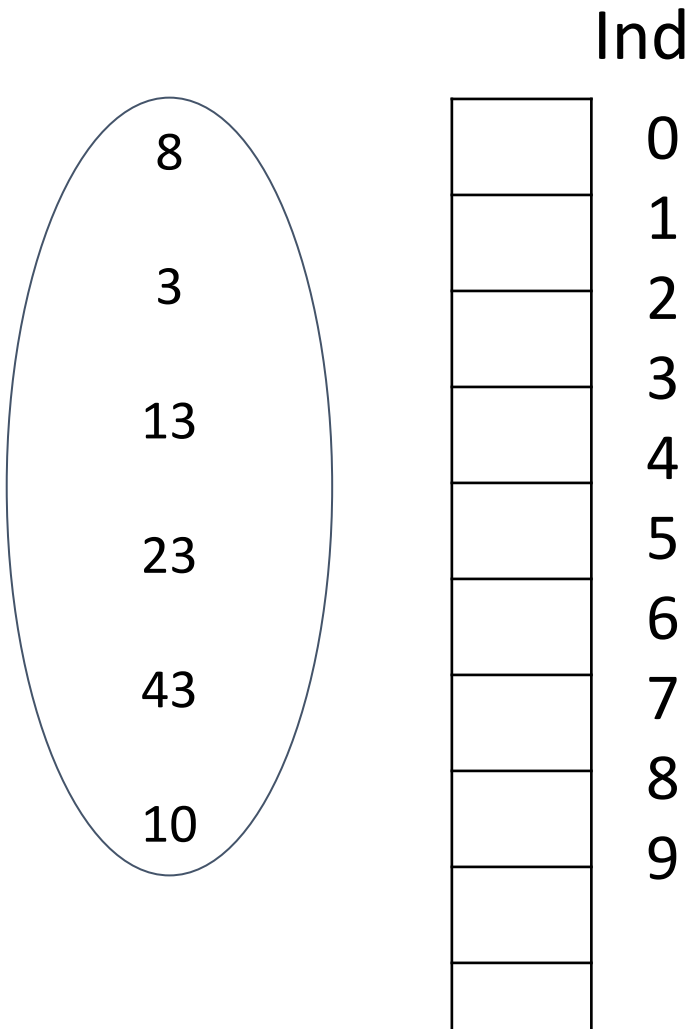
Hash Function (if collision)

h'(x) = [h(x) + f(i)] % size

f(i) = i^2

What is Quadratic Probing?

h(x) = x % size **h'(x) = [h(x) + f(i)] % size** **f(i) = i^2**



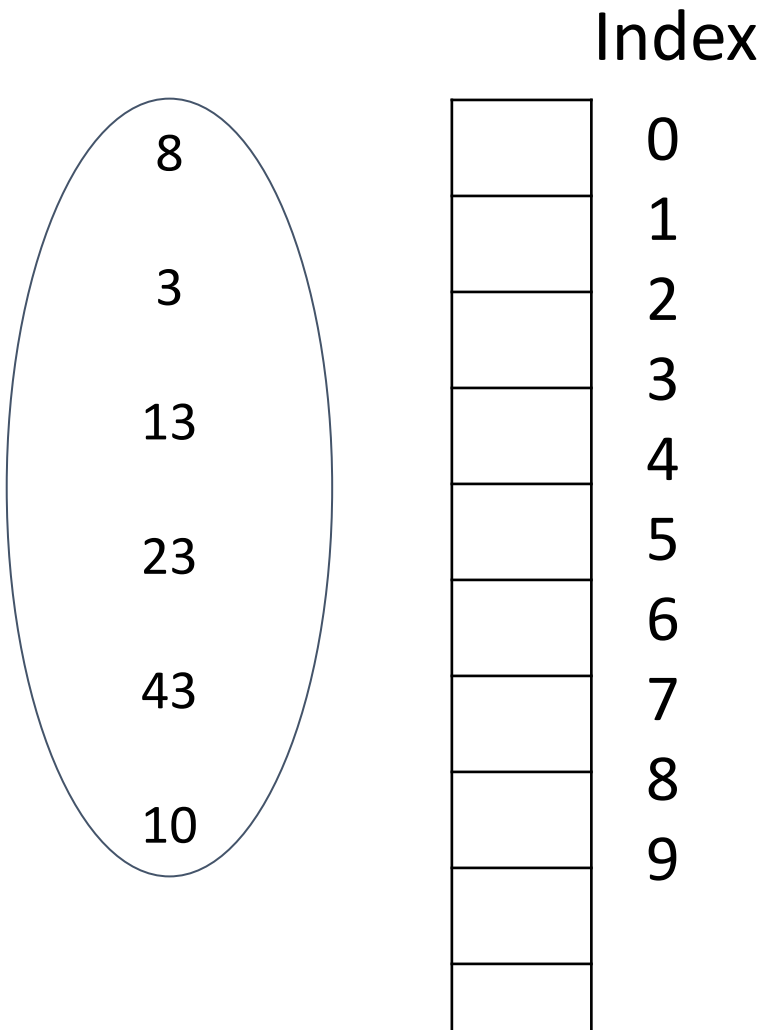
What is ⭐Chaining?

A way to resolve collisions by creating a LinkedList at that Index (also called a “bucket”)

- Combines both features of ArrayList Indexing and the ease of adding values using LinkedLists

What is ⭐Chaining?

h(x) = x % size



Recap (Comparison)

Linear Probing	Quadratic Probing	⭐ Chaining
A way to resolve collisions by adding the element in the next available spot	A way to resolve collisions by adding the element in the next available spot (quadratically)	A way to resolve collisions by creating a LinkedList at that Index (also called a “bucket”)

Why ⭐Chaining?

Clustering - A tendency for data to clump together when using solutions to Collisions like Linear and Quadratic probing

- Linear and Quadratic Probing often result in “Clustering”
- Inefficient use of space in the table
- This means the Runtimes will also be slower

1	2	3	4	5	6				7							8	9		

Why ⭐Chaining?

- Hashing can reduce it down to O(1)
- “Load Factor” - lambda ($\boldsymbol{\lambda}$)
  - the number of values in each LinkedList
- Finding the index in the Table is O(1)
- Finding value in LinkedList is O($\boldsymbol{\lambda}$) or essentially O(1)

Standard Java hashCode

```
hash = 0
for (each field) {
    hash = 31 * hash + hash(field)
}
```