CS 225

Data Structures

March 26 – Hash Table Collisions

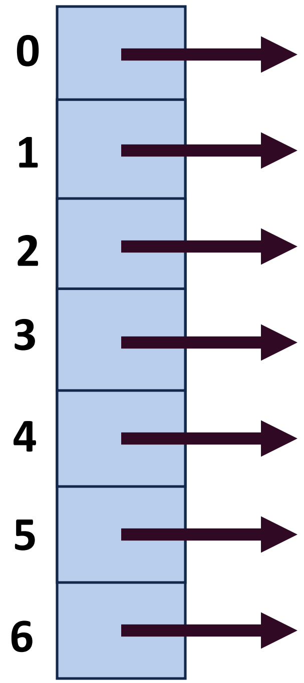
Wade Fagen-Ulmschneider

(Example of open hashing)

Collision Handling: Separate Chaining

S = { 16, 8, 4, 13, 29, 11, 22 } |S| = n

h(k) = k % 7 |Array| = N



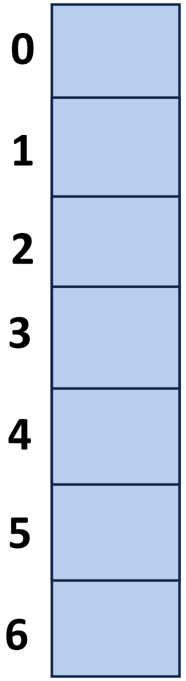
	Worst Case	SUHA
Insert	O(1)	O(1)
Remove/Find	O(n)	O(α)

(Example of closed hashing)

Collision Handling: Probe-based Hashing

S = { 16, 8, 4, 13, 29, 11, 22 } **|S| = n**

h(k) = k % 7 **|Array| = N**



(Example of closed hashing)

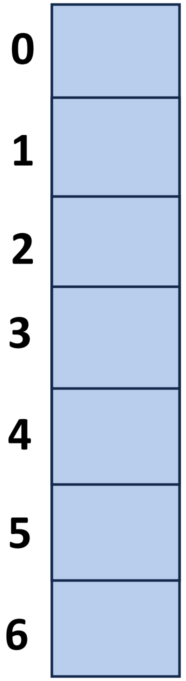
Collision Handling: Linear Probing

S = { 16, 8, 4, 13, 29, 11, 22 }

|S| = n

h(k) = k % 7

|Array| = N



Try h(k) = (k + 0) % 7, if full…

Try h(k) = (k + 1) % 7, if full…

Try h(k) = (k + 2) % 7, if full…

Try …

	Worst Case	SUHA
Insert		
Remove/Find		

A Problem w/ Linear Probing

Primary clustering:

Description:

Remedy:

(Example of closed hashing)

Collision Handling: Double hashing

S = { 16, 8, 4, 13, 29, 11, 22 } $|S| = n$

$h(k) = k \% 7$ $|Array| = N$

0	
1	
2	
3	
4	
5	
6	

Try $h(k) = (k + 0*h_2(k)) \% 7$, if full...

Try $h(k) = (k + 1*h_2(k)) \% 7$, if full...

Try $h(k) = (k + 2*h_2(k)) \% 7$, if full...

Try ...

$$h(k, i) = (h_1(k) + i*h_2(k)) \% 7$$

Running Times

The expected number of probes for find(key) under SUHA

Linear Probing:
- Successful: **$\frac{1}{2}(1 + 1/(1-\alpha))$**
- Unsuccessful: **$\frac{1}{2}(1 + 1/(1-\alpha))^2$**

Double Hashing:
- Successful: **$1/\alpha * \ln(1/(1-\alpha))$**
- Unsuccessful: **$1/(1-\alpha)$**

Separate Chaining:
- Successful: **$1 + \alpha/2$**
- Unsuccessful: **$1 + \alpha$**

(Don’t memorize these equations, no need.)

Instead, observe:

- **As α increases:**

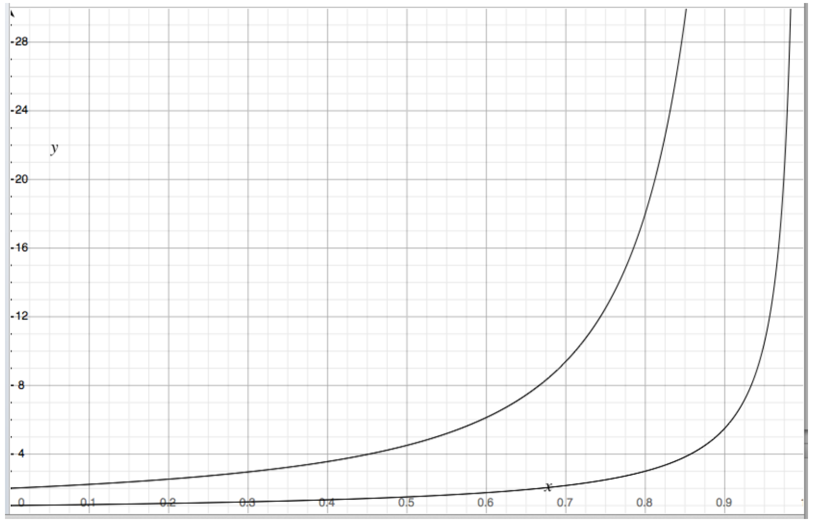
- **If α is constant:**

Running Times

The expected number of probes for find(key) under SUHA

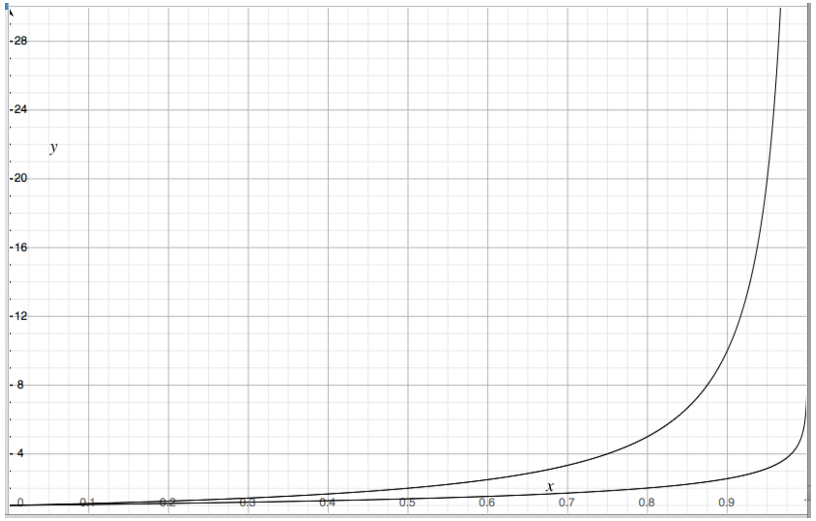
Linear Probing:

- Successful: **½(1 + 1/(1-α))**
- Unsuccessful: **½(1 + 1/(1-α))²**



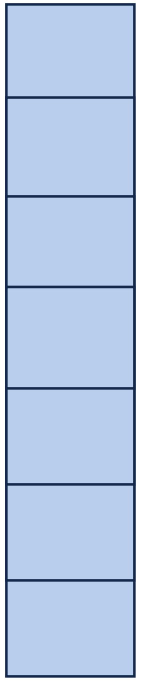
Double Hashing:

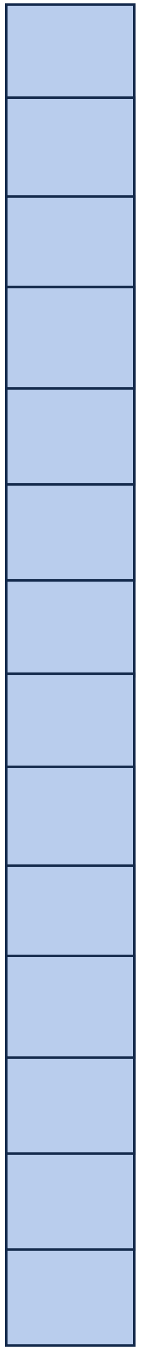
- Successful: **1/α * ln(1/(1-α))**
- Unsuccessful: **1/(1-α)**



ReHashing

What if the array fills?





Which collision resolution strategy is better?

- Big Records:

- Structure Speed:

What structure do hash tables replace?

What constraint exists on hashing that doesn't exist with BSTs?

Why talk about BSTs at all?

Running Times

	Hash Table	AVL	Linked List
Find	Amortized: Worst Case:		
Insert	Amortized: Worst Case:		
Storage Space			

std data structures

std::map

std data structures

std::map

::operator[]
::insert
::erase

::lower_bound(key) ➔ Iterator to first element ≤ key
::upper_bound(key) ➔ Iterator to first element > key

std data structures

std::unordered_map

::operator[]

::insert

::erase

~~::lower_bound(key) ➔ Iterator to first element ≤ key~~

~~::upper_bound(key) ➔ Iterator to first element > key~~

std data structures

std::unordered_map

::operator[]

::insert

::erase

~~::lower_bound(key) ➔ Iterator to first element ≤ key~~

~~::upper_bound(key) ➔ Iterator to first element > key~~

::load_factor()

::max_load_factor(ml) ➔ Sets the max load factor

Secret, Mystery Data Structure

ADT:

insert
remove
isEmpty