CS 225

Data Structures

Oct. 28 – Hash Table Collisions

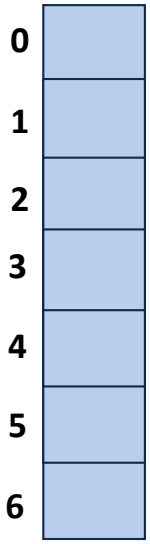
G Carl Evans

(Example of open hashing)

Collision Handling: Separate Chaining

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

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



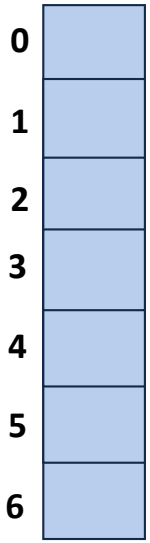
	Worst Case	SUHA
Insert	O(1)	O(1)
Remove/Find	O(n)	O(n/N)

(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**

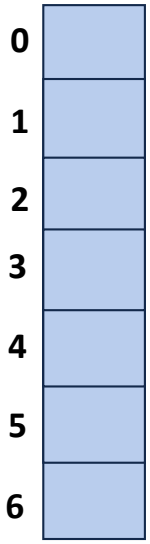


(Example of closed hashing)

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 …

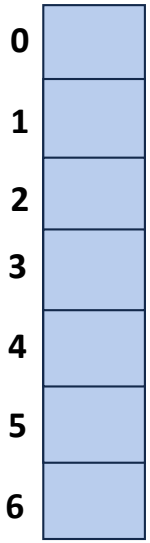
	Worst Case	SUHA
Insert		
Remove/Find		

(Example of closed hashing)

Collision Handling: Double hashing

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

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



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: **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)**

Separate Chaining:

- Successful: **1 + α/2**
- Unsuccessful: **1 + α**

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

Instead, observe:

- **As α increases:**

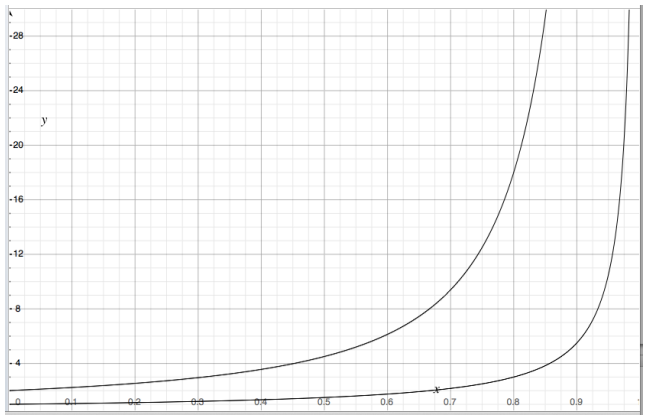
- **If α is constant:**

Running Times

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

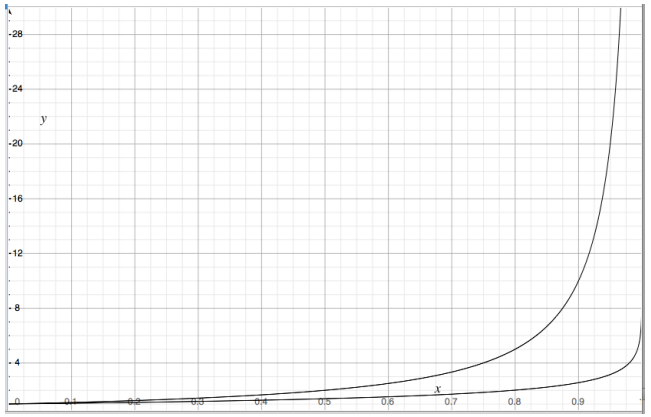
Linear Probing:

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



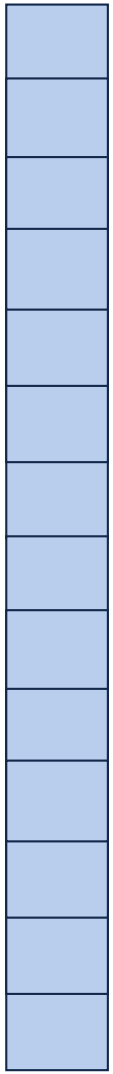
Double Hashing:

- 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?

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

Running Times

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