Algorithms and Data Structures for Data Science
Hashing 2

CS 277
Brad Solomon

February 20, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Learning Objectives

Review what a hash table is and what its key weakness is

Introduce closed hashing strategies

A Hash Table based Dictionary

```
d = {}
d[k] = v
print(d[k])
```

A **Hash Table** consists of three things:

1. A hash function

2. A data storage structure

3. A method of addressing *hash collisions*

Open vs Closed Hashing

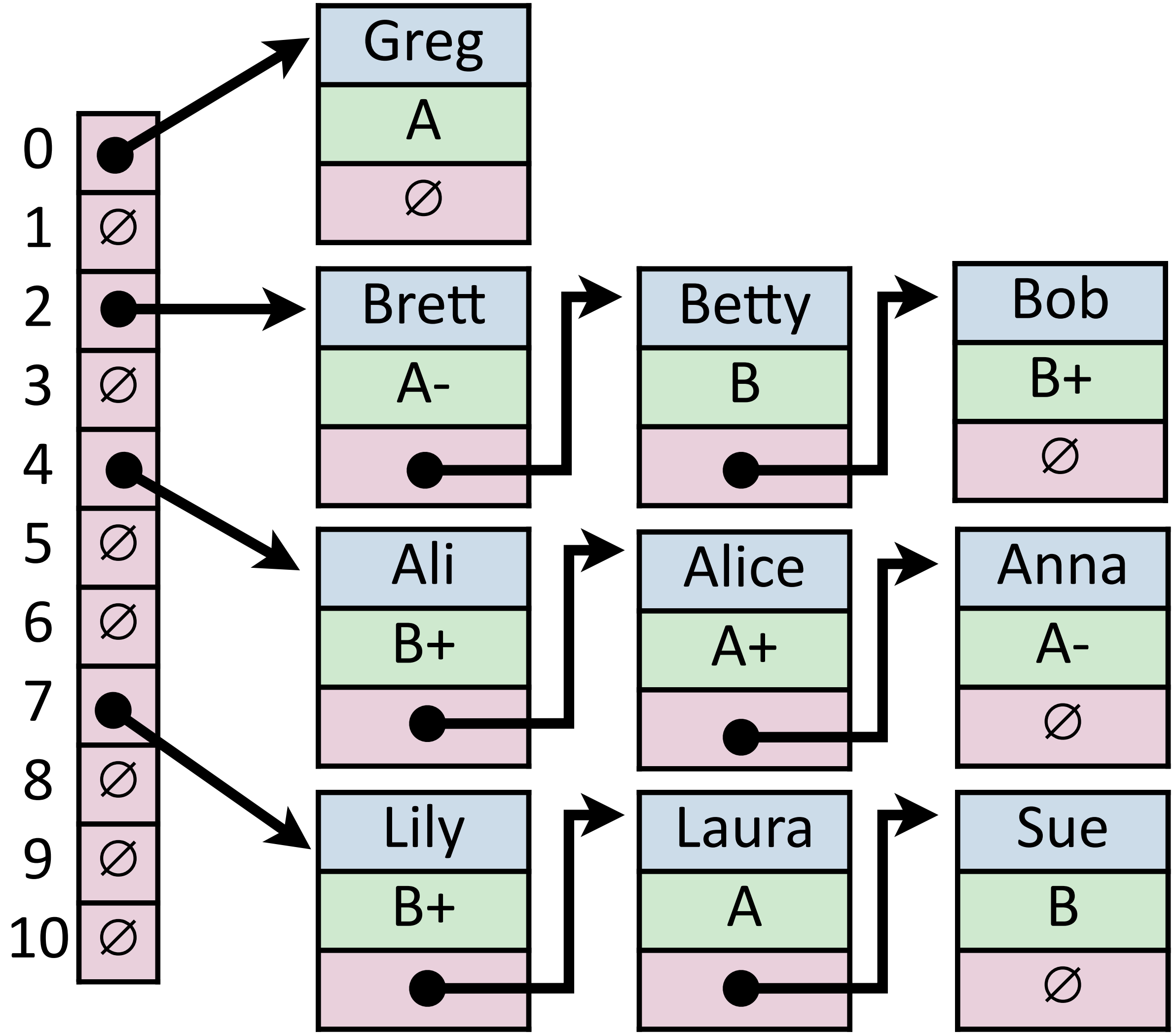
Addressing hash collisions depends on your storage structure.

- **Open Hashing:** store *k,v* pairs externally

- **Closed Hashing:** store *k,v* pairs in the hash table

Open Hashing: Separate Chaining

Key	Value	Hash
Bob	B+	2
Anna	A-	4
Alice	A+	4
Betty	B	2
Brett	A-	2
Greg	A	0
Sue	B	7
Ali	B+	4
Laura	A	7
Lily	B+	7



Simple Uniform Hashing Assumption

Given table of size m, a simple uniform hash, h, implies

$$\forall k_1, k_2 \in U \text{ where } k_1 \neq k_2 \text{ , } Pr(h[k_1] = h[k_2]) = \frac{1}{m}$$

Uniform: keys are equally likely to hash to any position

Independent: key hash values are independent of other keys

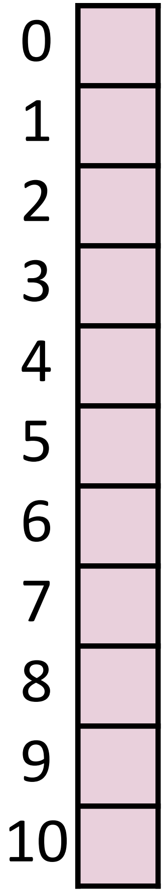
Separate Chaining Under SUHA

Under SUHA, a hash table of size *m* and *n* elements:

find runs in: ____________.

insert runs in: ____________.

remove runs in: ____________.



Open vs Closed Hashing

Addressing hash collisions depends on your storage structure.

- **Open Hashing:** store *k,v* pairs externally

- **Closed Hashing:** store *k,v* pairs in the hash table

Collision Handling: Probe-based Hashing

S = { 1, 8 , 15}

|S| = n

h(k) = k % 7

|Array| = m

0	
1	
2	
3	
4	
5	
6	

(Example of closed hashing)

Collision Handling: Linear Probing

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

h(k) = k % 7

|S| = n

|Array| = m

0	
1	
2	
3	
4	
5	
6	

h(k, i) = (k + i) % 7

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 …

(Example of closed hashing)

Collision Handling: Linear Probing

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

h(k, i) = (k + i) % 7 **|Array| = m**

`_find(29)`

0	22
1	8
2	16
3	29
4	4
5	11
6	13

(Example of closed hashing)

Collision Handling: Linear Probing

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

$|S| = n$

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

$|Array| = m$

`_remove(16)`

0	22
1	8
2	16
3	29
4	4
5	11
6	13

A Problem w/ Linear Probing

Primary clustering:

0	
1	**1**
2	**1’**
3	**3**
4	**1’’**
5	**3’**
6	
7	
8	
9	

Description:

Remedy:

(Example of closed hashing)

Collision Handling: Quadratic Probing

S = { 16, 8, 4, 13, 29, 12, 22 } **|S| = n**

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

0	
1	**8**
2	**16**
3	
4	**4**
5	
6	**13**

h(k, i) = (k + i*i) % 7

Try h(k) = (k + 0) % 7, if full…

Try h(k) = (k + 1*1) % 7, if full…

Try h(k) = (k + 2*2) % 7, if full…

Try …

(*Example of closed hashing*)

Collision Handling: Quadratic Probing

S = { 16, 8, 4, 13, 29, 12, 22 } **|S| = n**

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

0	**12**
1	**8**
2	**16**
3	**22**
4	**4**
5	**29**
6	**13**

`_find(11)`

`_remove(16)`

A Problem w/ Quadratic Probing

Secondary clustering:

0	**0**
1	**0’**
2	
3	
4	**0’’**
5	
6	
7	
8	
9	**0’’’**

Description:

Remedy:

(Example of closed hashing)

Collision Handling: Double Hashing

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

$h_1(k) = k \% 7$

$h_2(k) = 5 - (k \% 5)$

$|S| = n$

$|Array| = m$

0	
1	8
2	16
3	
4	4
5	
6	13

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

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 ...

Running Times

(Expectation under SUHA)

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

Open Hashing:

insert: ___________.

find/ remove: ___________.

Closed Hashing:

insert: ___________.

find/ remove: ___________.

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

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: **$1 + \alpha$**

Instead, observe:

- As α increases:

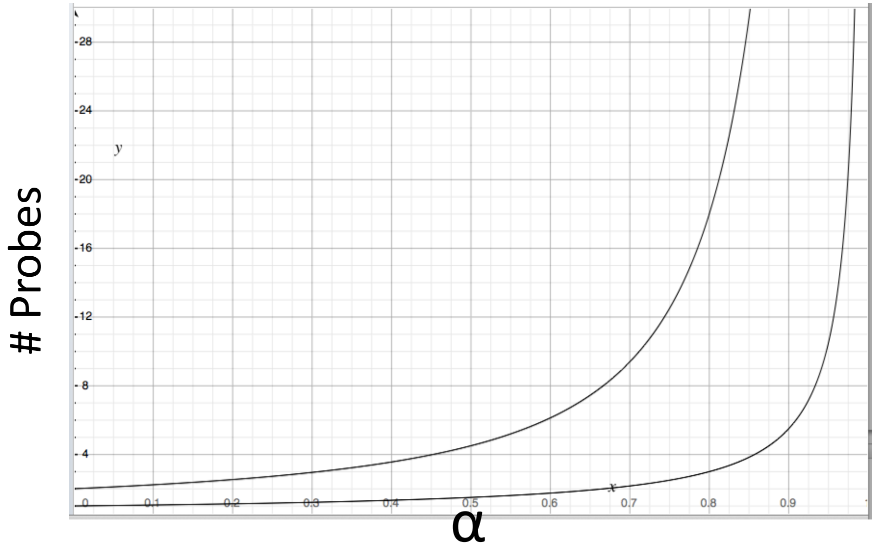
- If α is constant:

Running Times

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

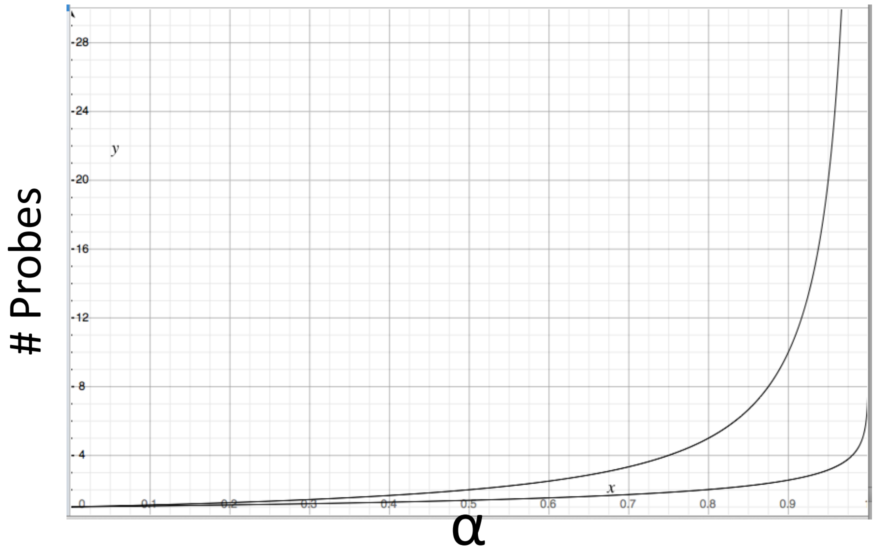
Linear Probing:

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



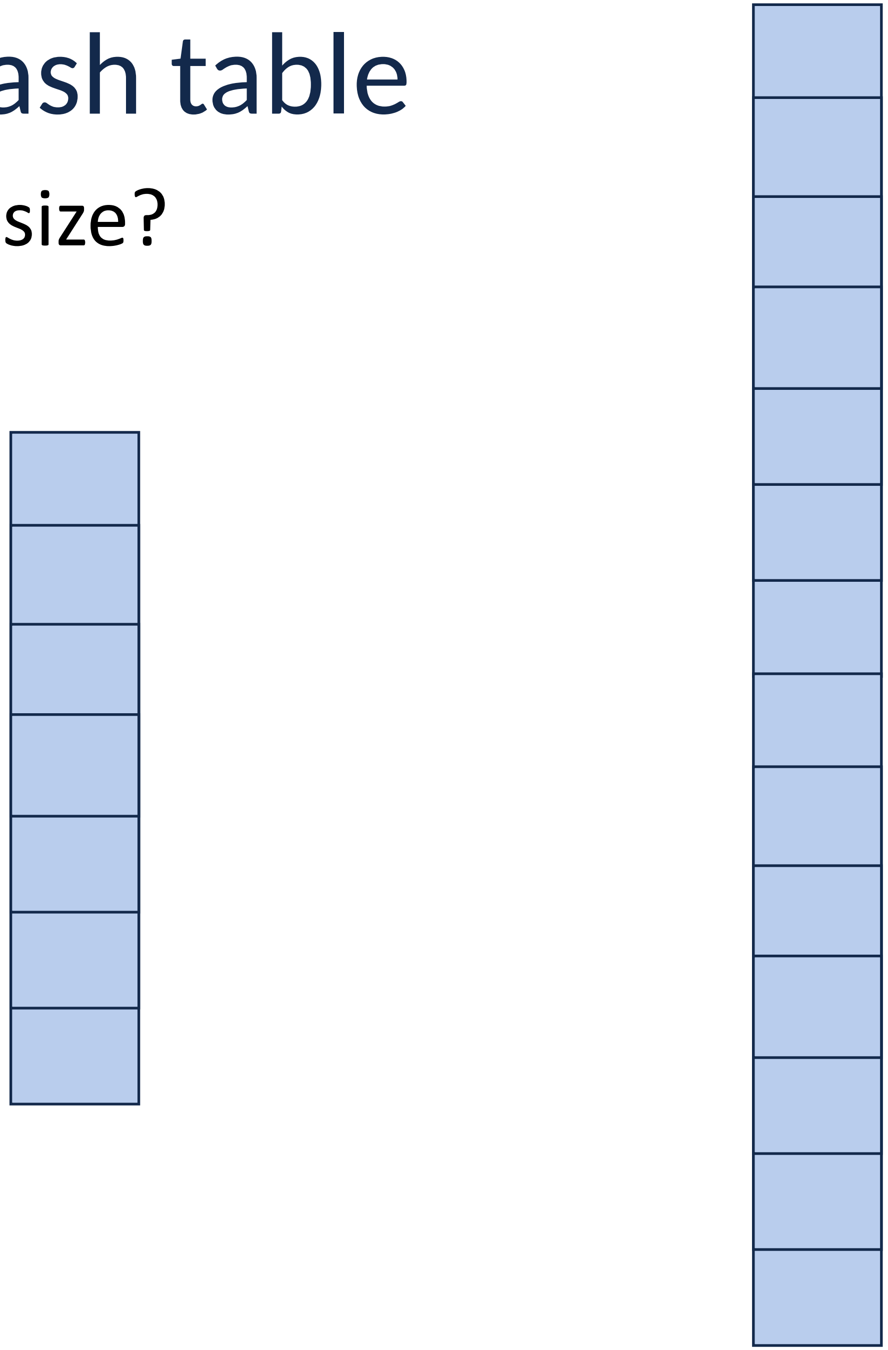
Double Hashing:

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



Resizing a hash table

How do we resize?



Running Times

	Hash Table	Array List	Linked List
Find	Amortized: Worst Case:		
Insert (Order Agnostic)	Amortized: Worst Case:	Amortized: Worst Case:	
Remove (By Value)	Amortized: Worst Case:		
Storage Space			

On Wednesday: More uses for hash functions!

Choosing a Hash Function

Python has a built-in hash! It’s pretty good if you run everything at once.

```
print(hash("I can pass in any string!”))


print(hash(205811))
```

Choosing a Hash Function

If you want something that is persistently deterministic, find a seeded hash

```
import mmh3

print(mmh3.hash("I can pass in any string!", 10)) #I got: -565691678
print(mmh3.hash("I can pass in any string!", 50)) #I got: -947521776
print(mmh3.hash("I can pass in any string!", 12)) #I got: 1680496801

```

Bonus Slides

Hash Function (Division Method)

Hash of form: $h(k) = k \,\%\, m$

Pro:

Con:

Hash Function (Multiplication Method)

Hash of form: $h(k) = \lfloor m(kA \,\%\, 1) \rfloor,\ 0 \leq A \leq 1$

Pro:

Con:

Hash Function (Universal Hash Family)

Hash of form: $h_{ab}(k) = \big((ak + b)\,\%\,p\big)\,\%\,m,\; a, b \in Z_p^*, Z_p$

$$\forall k_1 \neq k_2,\; Pr_{a,b}(h_{ab}[k_1] = h_{ab}[k_2]) \leq \frac{1}{m}$$

Pro:

Con: