Algorithms and Data Structures for Data Science

Hashing 2

CS 277
Brad Solomon

December 1, 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Lab_hash is optional

Release date: December 1st

Due date: December 8th

ICES Evaluations

You are strongly encouraged to provide course feedback

go.illinois.edu/ices-online

Learning Objectives

Motivate and define a hash table

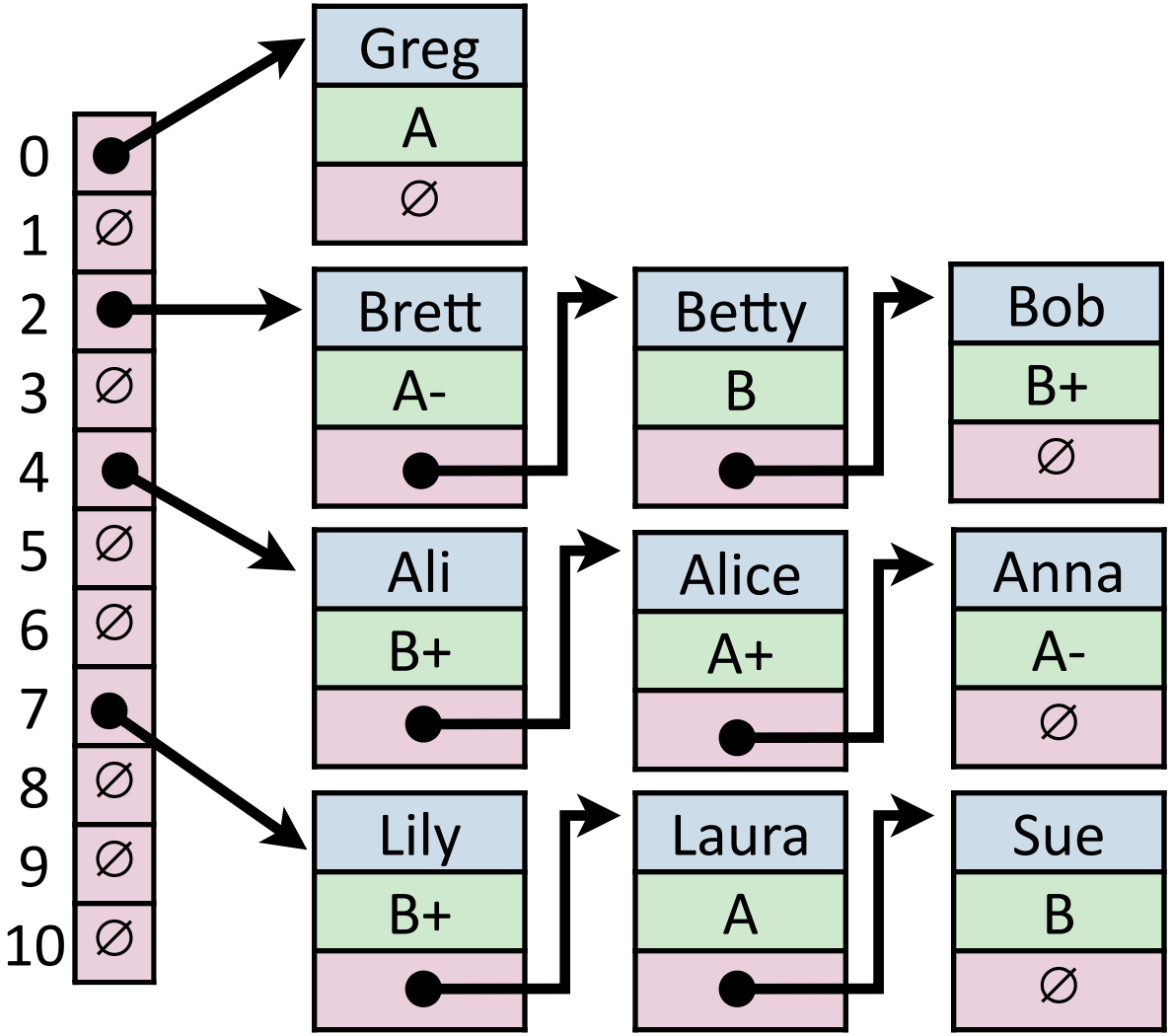
Discuss what a ‘good’ hash function looks like

Identify a key weakness of the hash table

Introduce strategies to ‘correct’ this weakness

Insertion (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

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

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

(Example of closed hashing)

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 }

|S| = n

h(k) = k % 7

|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

0	
1	
2	
3	
4	
5	
6	

`_find(29)`

`_remove(16)`

	Worst Case	SUHA
Insert		
Remove/Find		

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, 11, 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 …

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 }

|S| = n

$h_1(k) = k \% 7$

|Array| = m

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

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

(Expectation under SUHA)

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:

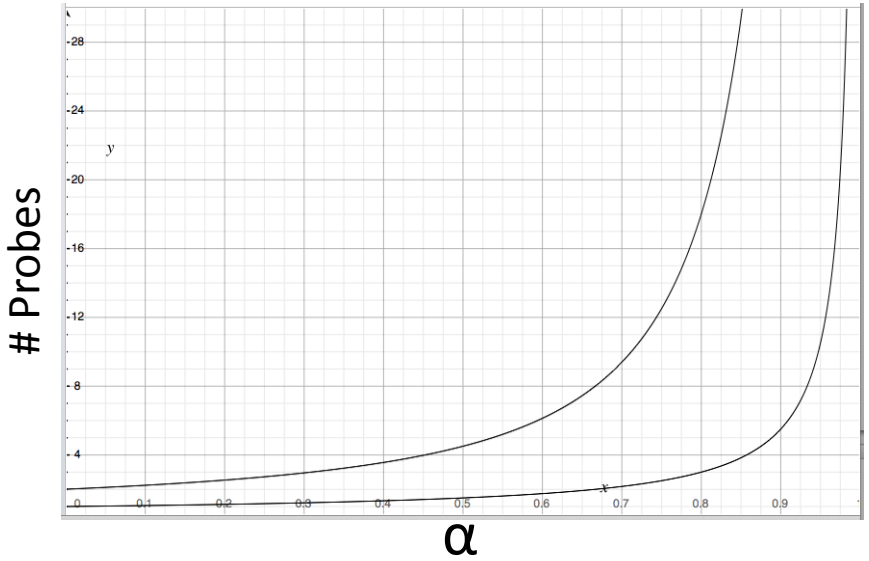
- If α is constant:

Running Times

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

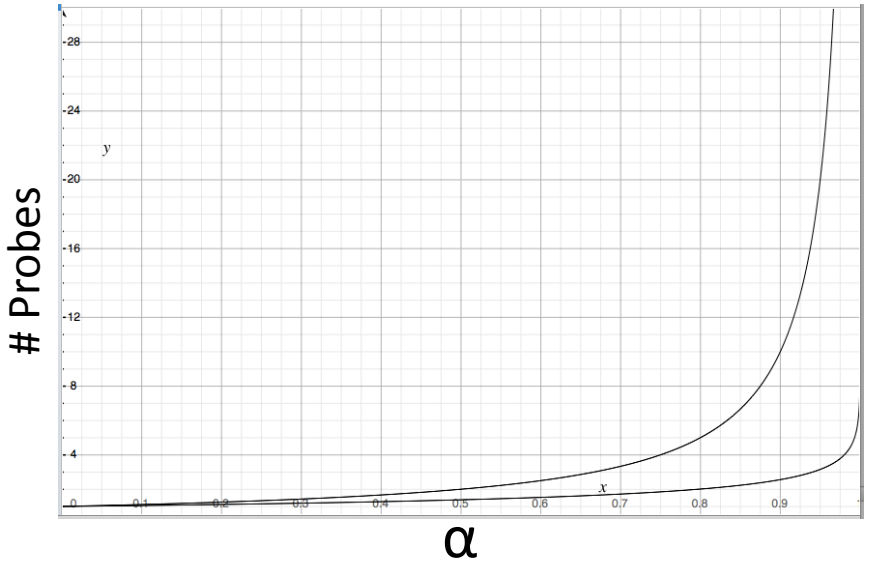
Linear Probing:

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



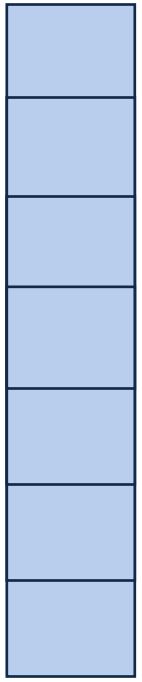
Double Hashing:

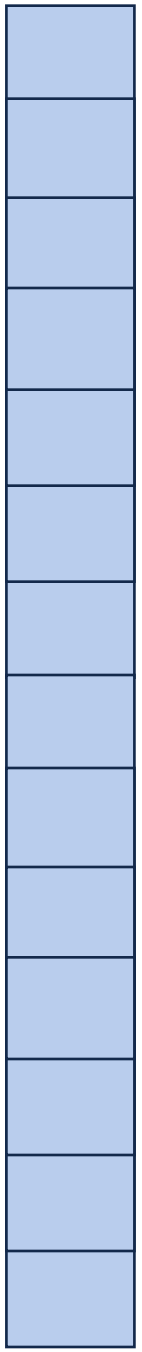
- Successful: **1/α * ln(1/(1-α))**
- Unsuccessful: **1/(1-α)**



ReHashing

What if the array “fills”?





Which collision resolution strategy is better?

- Big Records:

- Structure Speed:

What structure do hash tables implement?

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			

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: