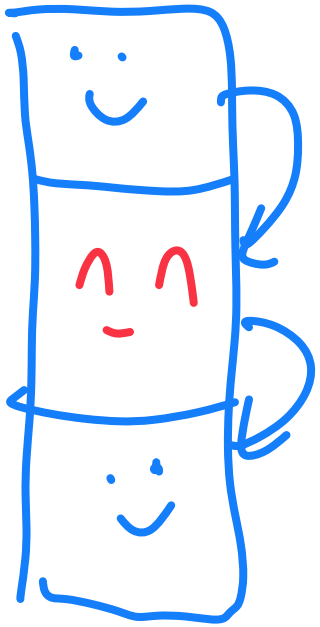
Data Structures and Algorithms

Hashing 2

CS 225

Brad Solomon

October 30, 2023



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Extra Credit Project Timeline

All extra credit projects need to be submitted by October 31st

All projects submitted by Saturday have been graded

On Sunday total number of submissions doubled

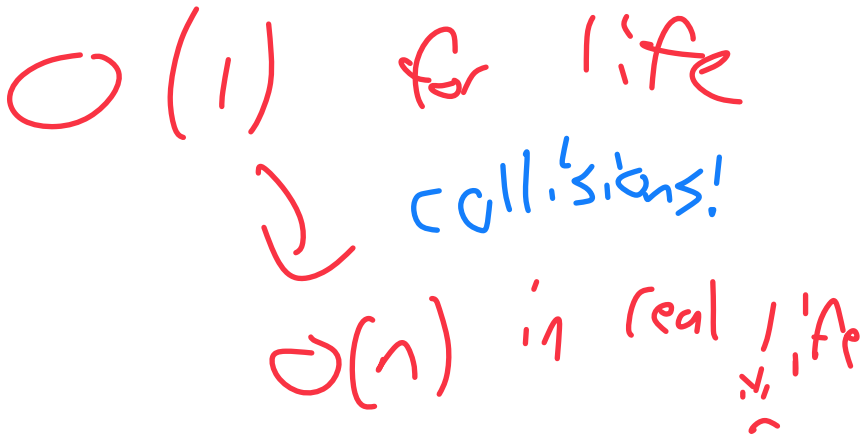
Mentor assignments by next Monday

Exam 4 Practice Exam

Releasing later today!

Learning Objectives

Review fundamentals of hash tables



Introduce closed hashing approaches to hash collisions

Determine when and how to resize a hash table

Justify when to use different index approaches

A Hash Table based Dictionary

User Code (is a map):

```
1  Dictionary<KeyType, ValueType> d;
2  d[k] = v;
```

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

1. A hash function

Data → int

2. A data storage structure

Array

3. A method of addressing *hash collisions*

Open vs Closed Hashing

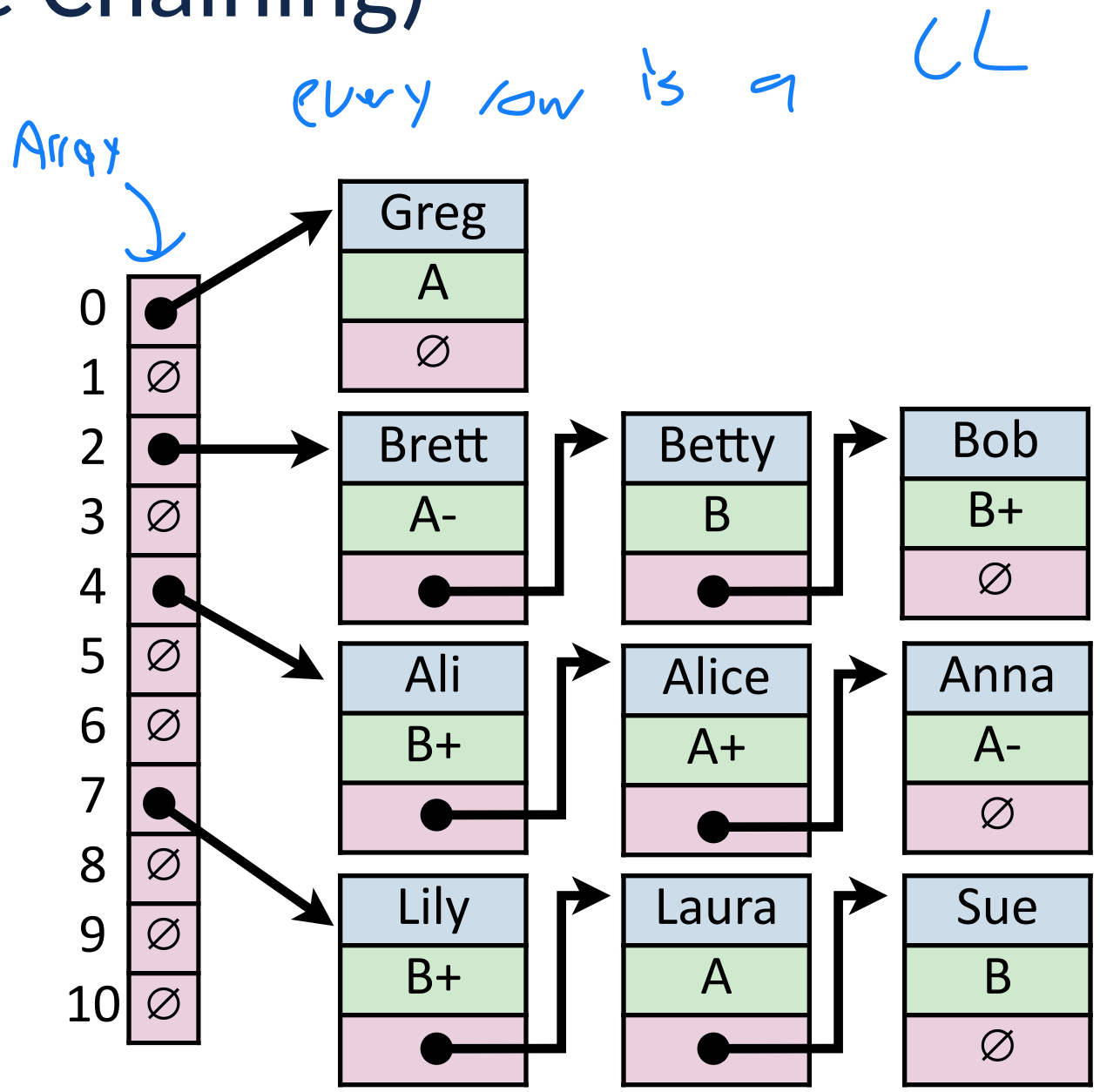
Addressing hash collisions depends on your storage structure.

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

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

Hash Table (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



Hash Table (Separate Chaining)

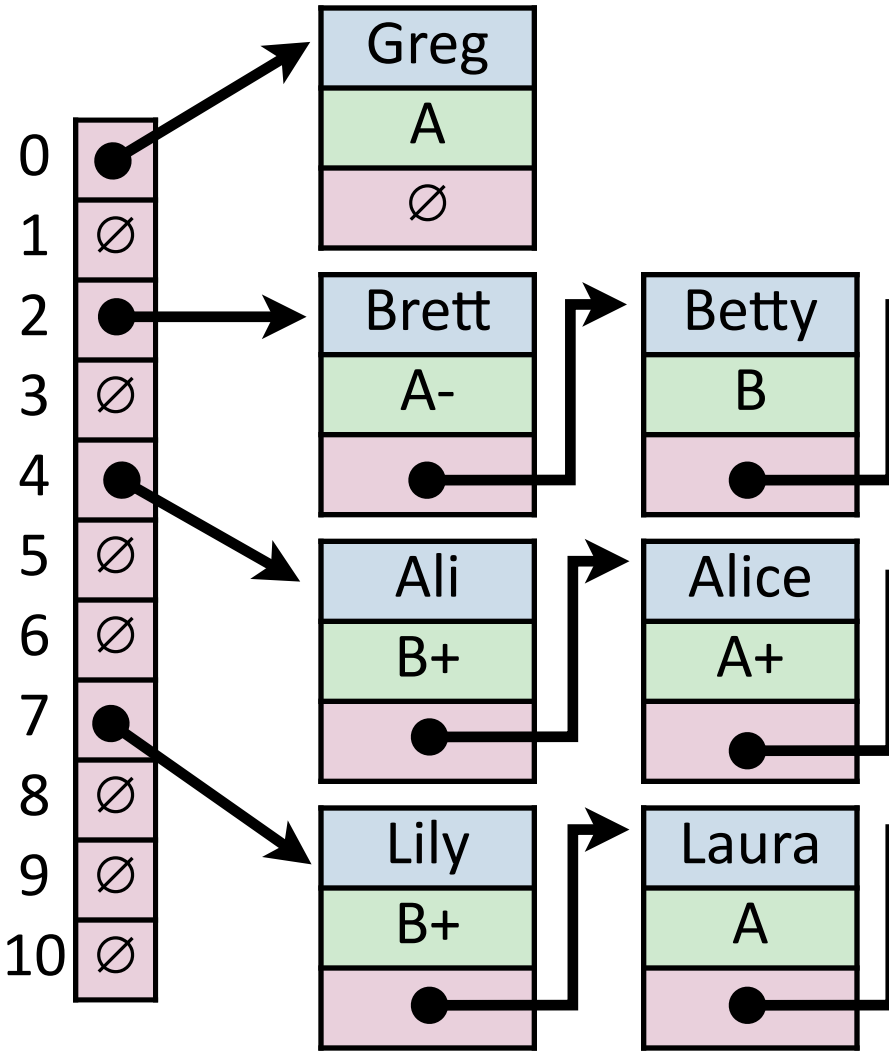
For hash table of size *m* and *n* elements:

Find runs in: O(n)

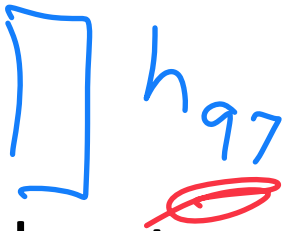
Insert runs in: O(1)

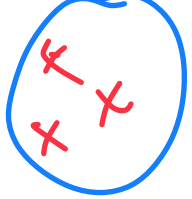
Remove runs in: O(n)

Big O



Hash Table





Worst-Case behavior is bad — but what about randomness?

1) **Fix *h***, our hash, and assume it is good for ***all keys***:

↳ SUHA

↳ Assume our hash is good!

↳ Not realistic!

uniform

independent

2) Create a ***universal hash function family:*** → Quicksort random pivot

Real world version of SUHA

↳ Assume I pick a random hash from a family of hashes!

Tangent: I can pick a random seed/hash/const as we construct table

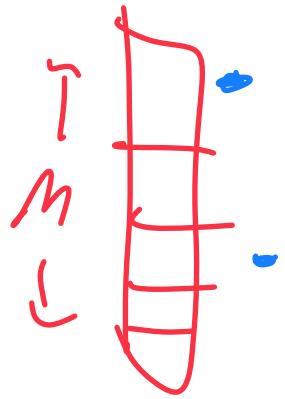
Simple Uniform Hashing Assumption

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

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

Uniform: All keys equally likely to hash any value

$Pr[h(k_1) = X] = \frac{1}{m}$



Independent: All keys hash independently of each other

Separate Chaining Under SUHA

Given table of size m and n inserted objects

Indicator function ↳ function either 1 or 0

Claim: Under SUHA, expected length of chain is $\frac{n}{m}$

Q_j = expected # of items hashing to position j

$$E[Q_j] = E\left[\sum_i H_{i,j}\right] = \sum_i \text{Prob}(H_{i,j}=1) \cdot 1$$

$$H_{i,j} = \begin{cases} 1 & \text{if item } i \text{ hashes to } j \\ 0 & \text{otherwise} \end{cases}$$

Expected value of indicator is the probability

$$E[R] = Pr(R) \cdot Val(R)$$

for any i = {0, 0, 1, 0, ...}

$$P[H_{i,j}] = \frac{1}{m}$$

$$E[Q_j] = n/m$$

Separate Chaining Under SUHA

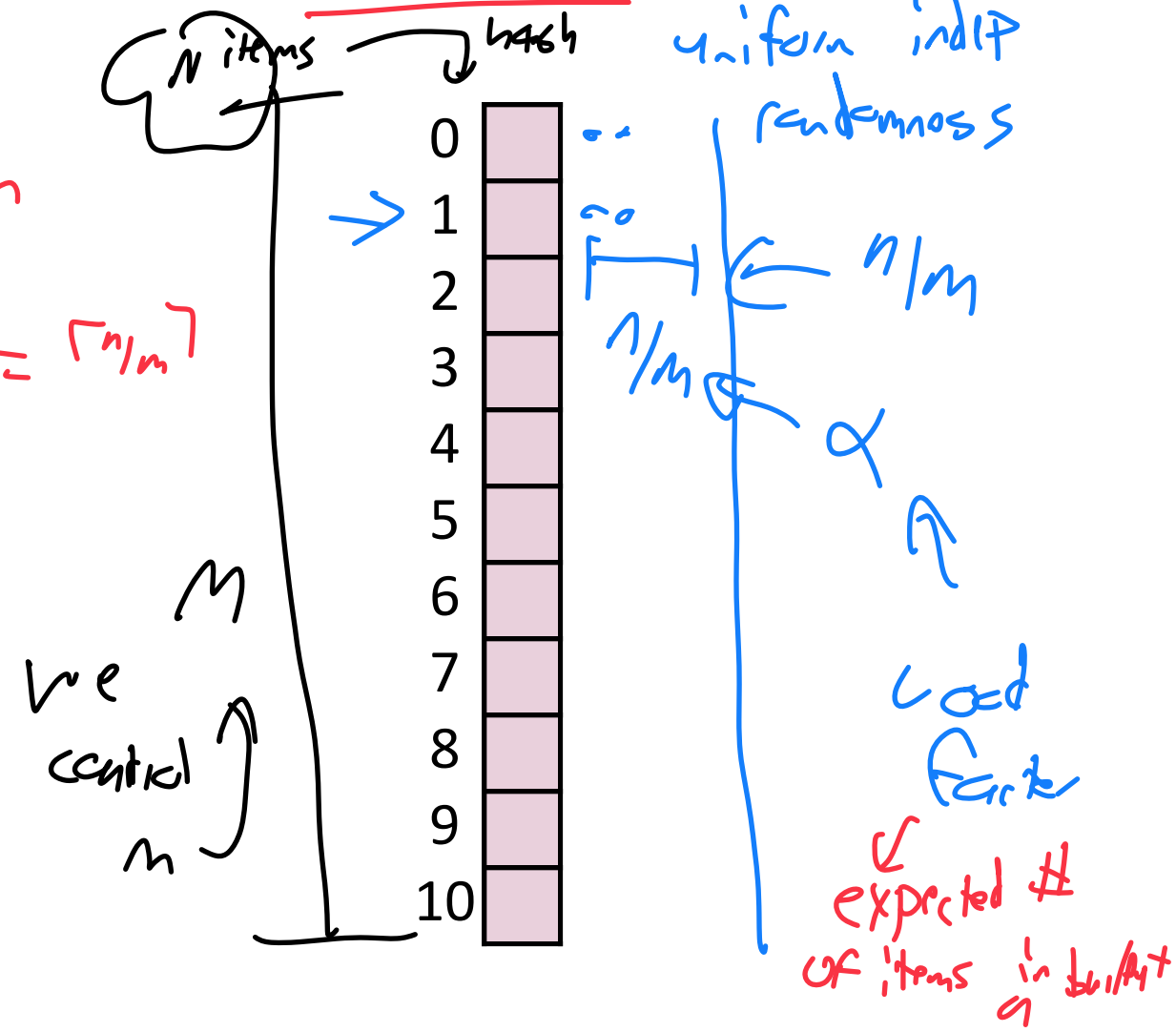
Under SUHA, a hash table of size *m* and *n* elements:

Find runs in: $1+\alpha$.

Insert runs in: $O(1)$.

Remove runs in: $1+\alpha$.

Expectations Not Big O



(Example of closed hashing)

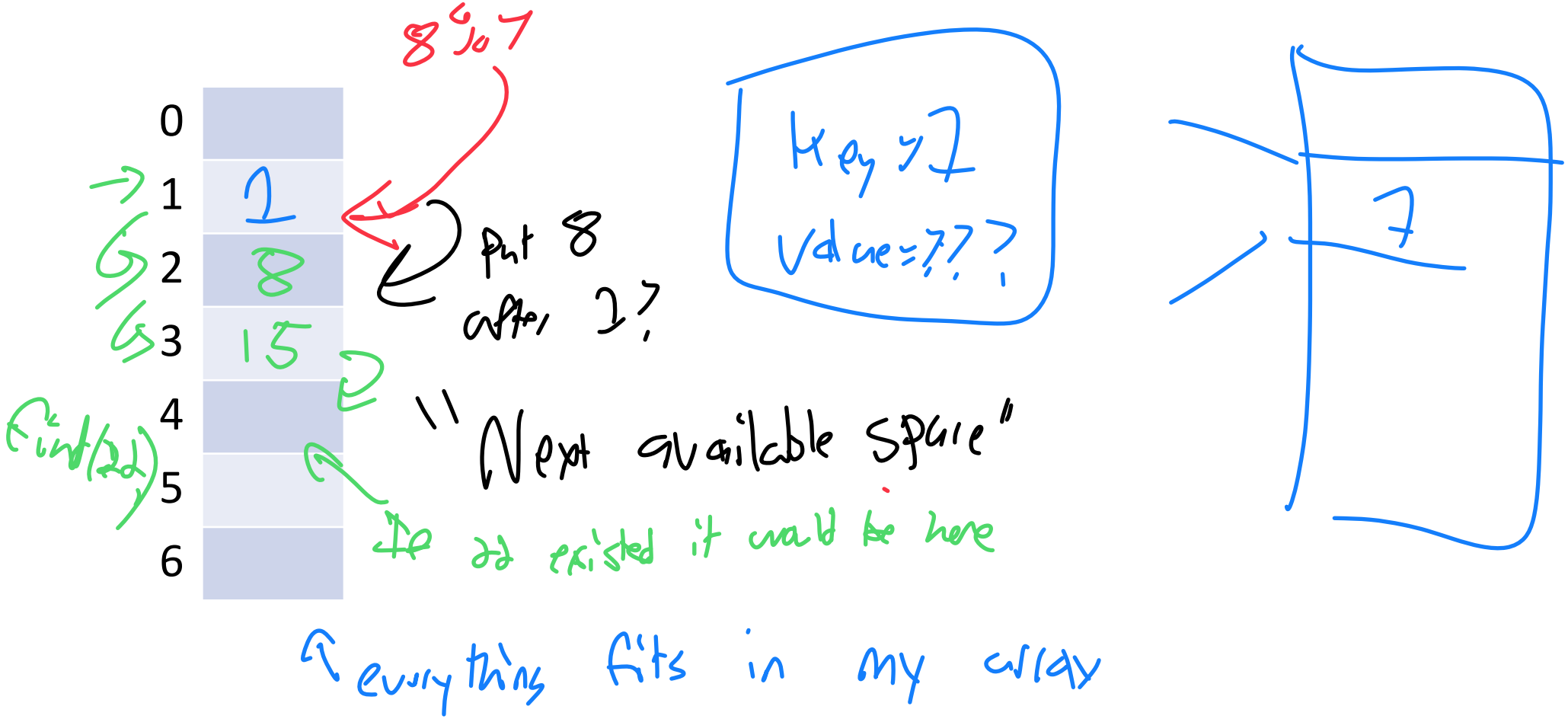
Collision Handling: Probe-based Hashing

S = { 1, 8 , 15}

|S| = n

h(k) = k % 7

|Array| = m

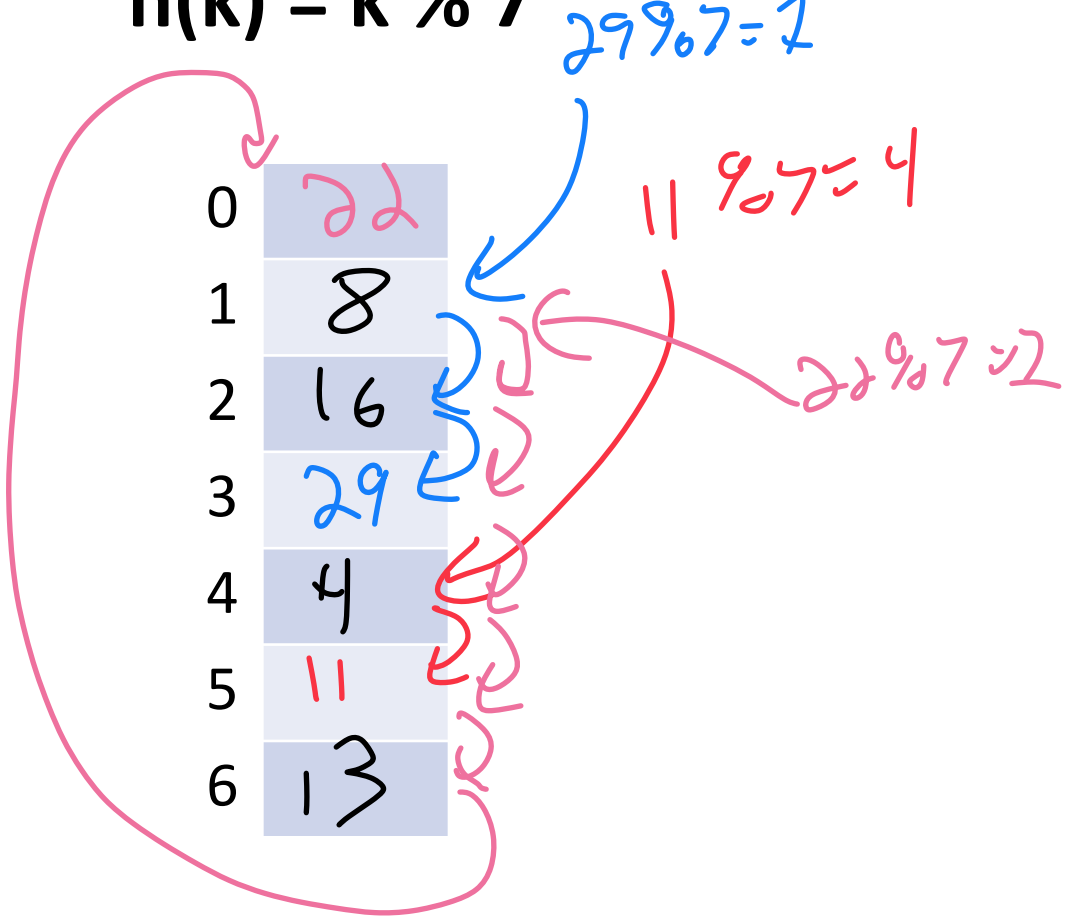


(Example of closed hashing)

Collision Handling: Linear Probing

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

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



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 ...

16
8
4
13
???

No way to search

(Example of closed hashing)

Collision Handling: Linear Probing

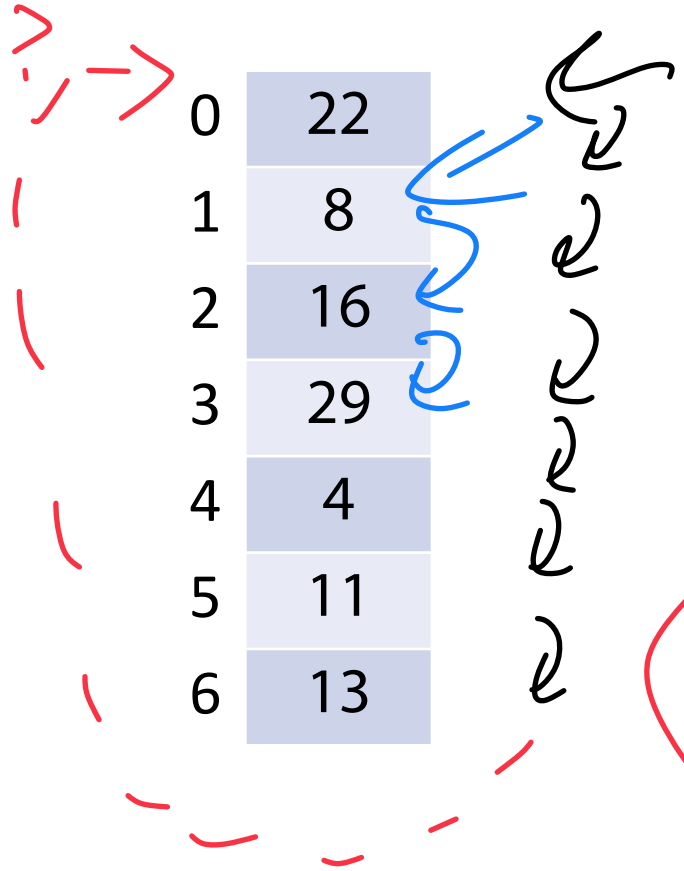
S = { 16, 8, 4, 13, 29, 11, 22 }

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

|S| = n

|Array| = m

find is O(n) :(



_find(29)

↳ 29 % 7 = 1

↳ find 29

Look at next space until match

or

See every item

or

_find(30) = 0

See empty space

↳ Stop after looking at entire list

↳ If item existed, it would be here!

(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**

If remove is array remove → tombstoning

_ggggg / 1 bit extra (bool) / Null

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

stops!

b/c tombstone keep going!

Allocating 1 bit to say if any item existed there ever

```
_remove(16)
1) find(16)
2) Remove
```

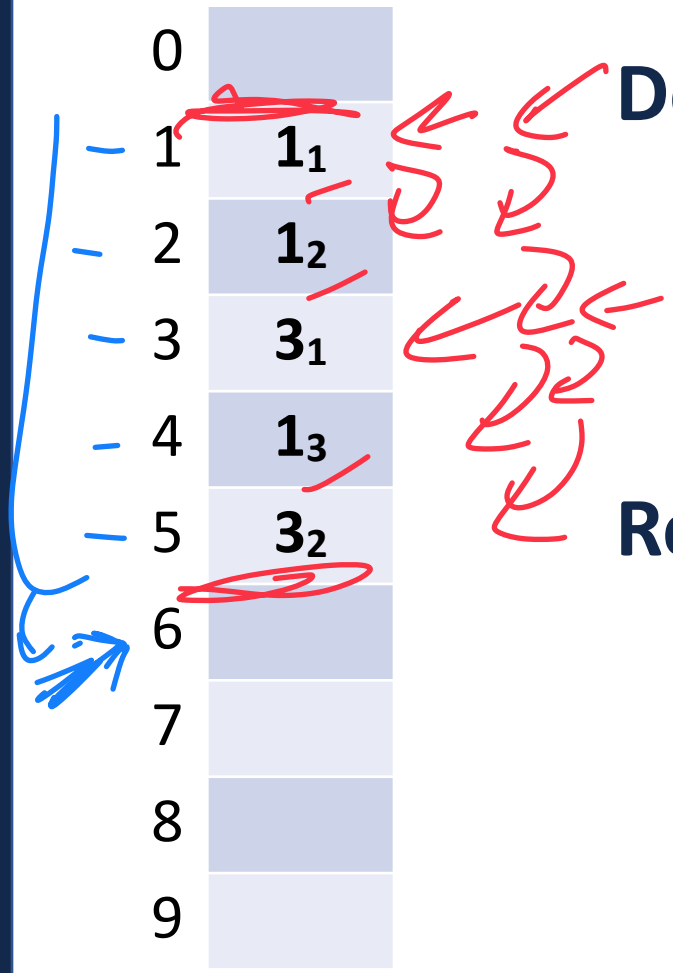
- Introduce empty spaces "bx mistake"

_find(29)

After removing we tombstone

A Problem w/ Linear Probing

Primary clustering: "Rich got richer"



Description: Collisions create long runs of filled in buckets

↳ Every item has 1/m chance to hash any where

↳ But "next available" doesn't work that way!

Remedy:

↳ Pick a better "next available"!

(Example of closed hashing)

Collision Handling: Quadratic Probing

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

|S| = n

h(k) = k % 7

|Array| = m

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

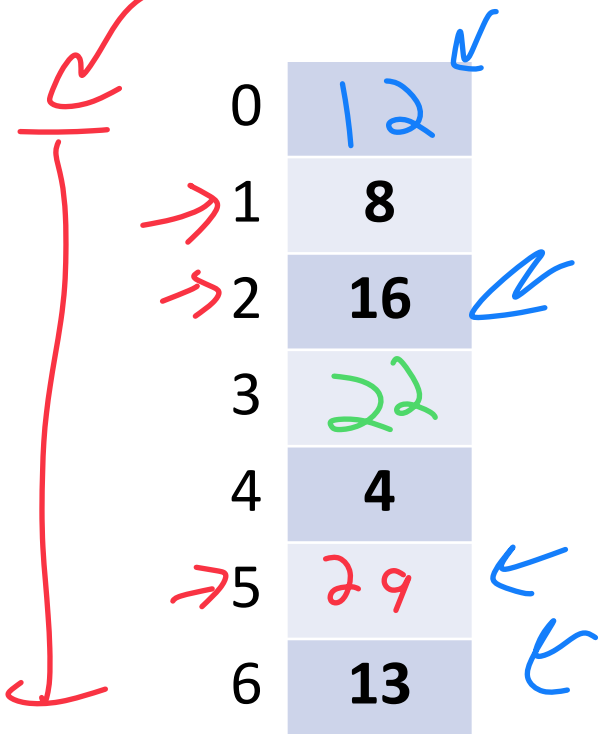
Try h(k) = (k + 0) % 7, if full…

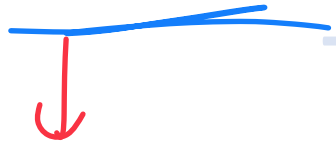
Try h(k) = (k + 1*1) % 7, if full…

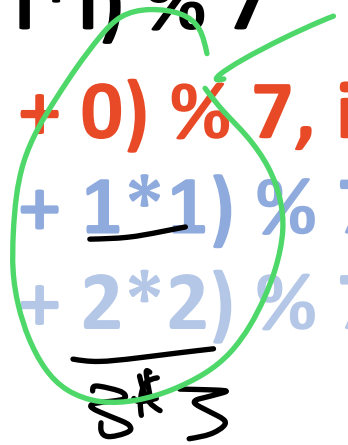
Try h(k) = (k + 2*2) % 7, if full…

Try …

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

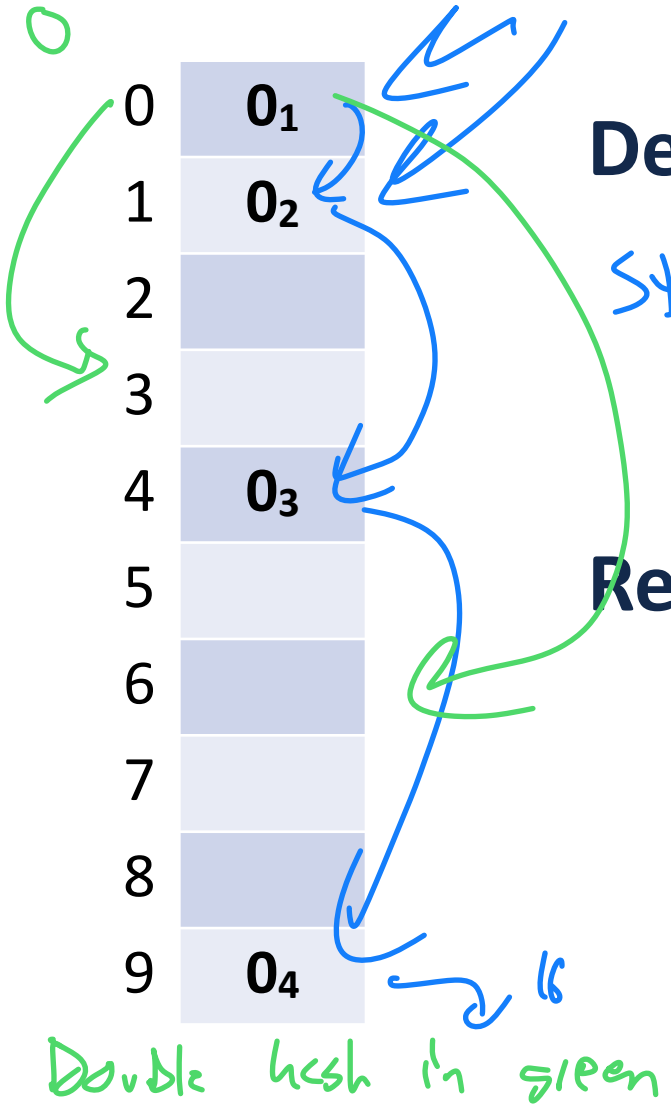






A Problem w/ Quadratic Probing

Secondary clustering: w/ 5



Description: Long runs of initial collisions

Still make long chains

Remedy: Be less consistent but still deterministic

(Example of closed hashing)

Collision Handling: Double Hashing

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

$h_1(k) = k \% 7$

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

Increment

|S| = n

|Array| = m

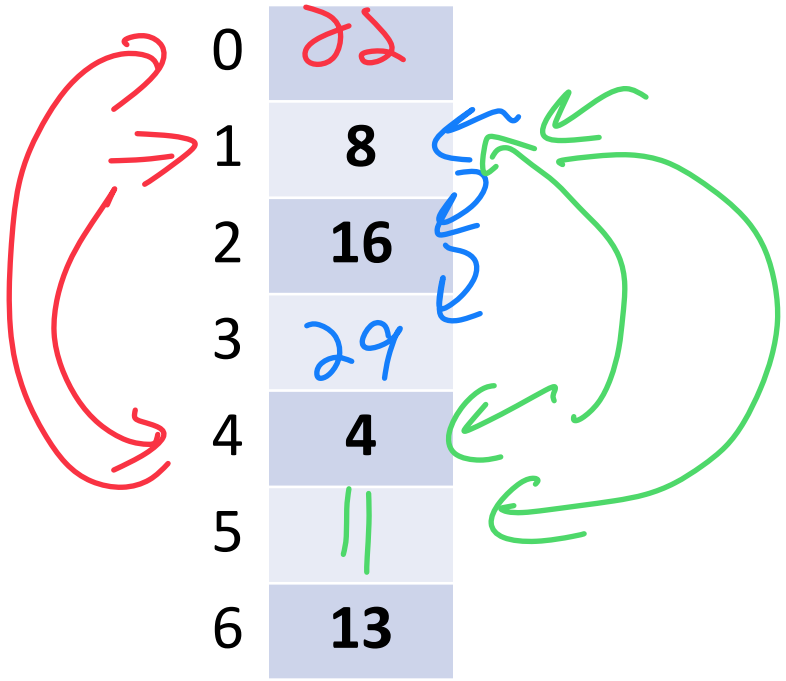
$h(k, i) = (h_1(k) + i*h_2(k)) \% 7$

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

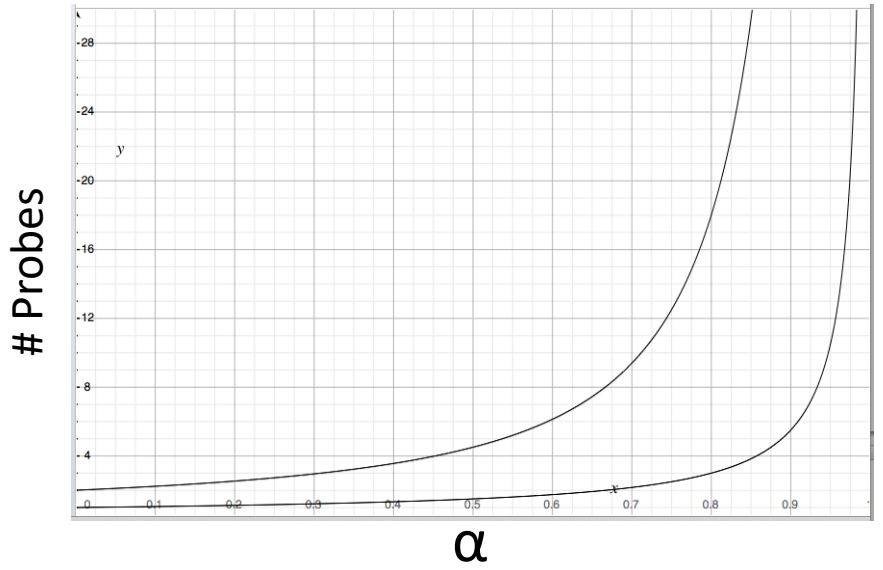
Linear Probing:

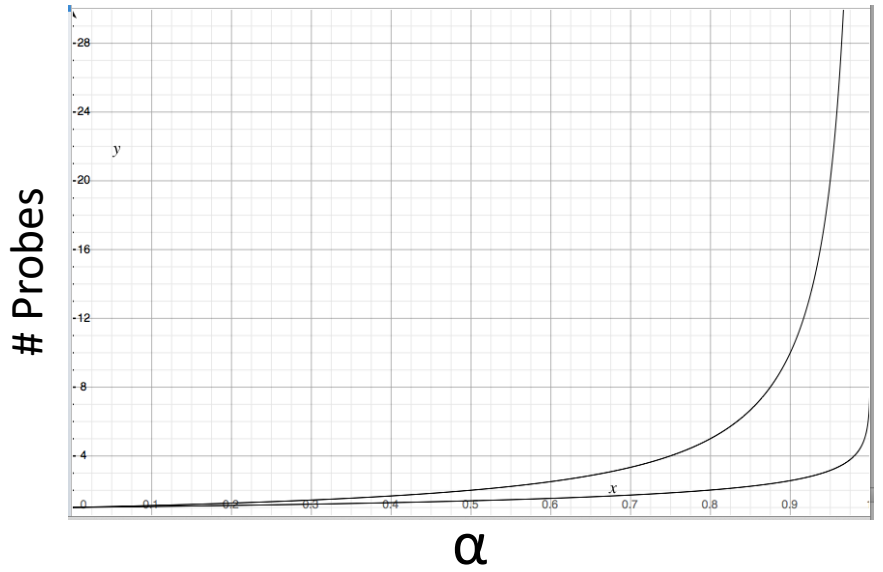
- Successful: **½(1 + 1/(1-α))**
- Unsuccessful: **½(1 + 1/(1-α))²**

Double Hashing:

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

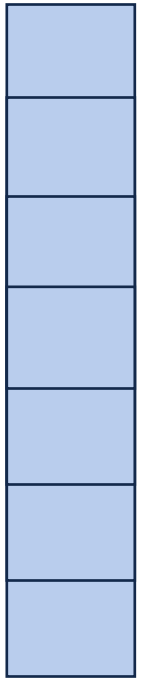
When do we resize?

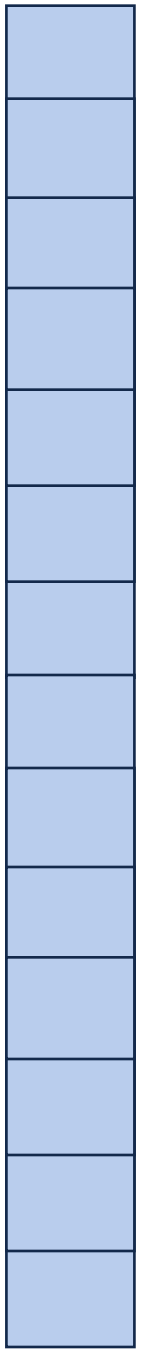




Resizing a hash table

How do you resize?





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	Expectation*: Worst Case:		
Insert	Expectation*: Worst Case:		
Storage Space			