Data Structures and Algorithms
Hashing and Probability

CS 225
Brad Solomon

November 18, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Reminder: Schedule Exam 5

Exam Dates: 11/28 — 11/30

Learning Objectives

Determine when and how to resize a hash table

Justify when to use different hashing approaches

Discuss common ways of creating a hash function

A return to skiplist theory

A Hash Table based Dictionary

Client Code:

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

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

1. A hash function

2. A data storage structure

3. A method of addressing *hash collisions*

Running Times

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

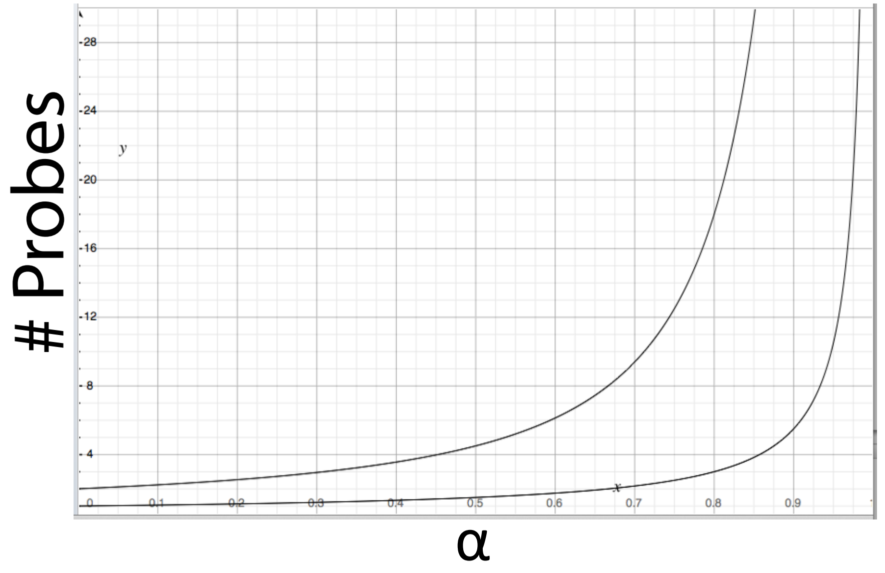
Linear Probing:

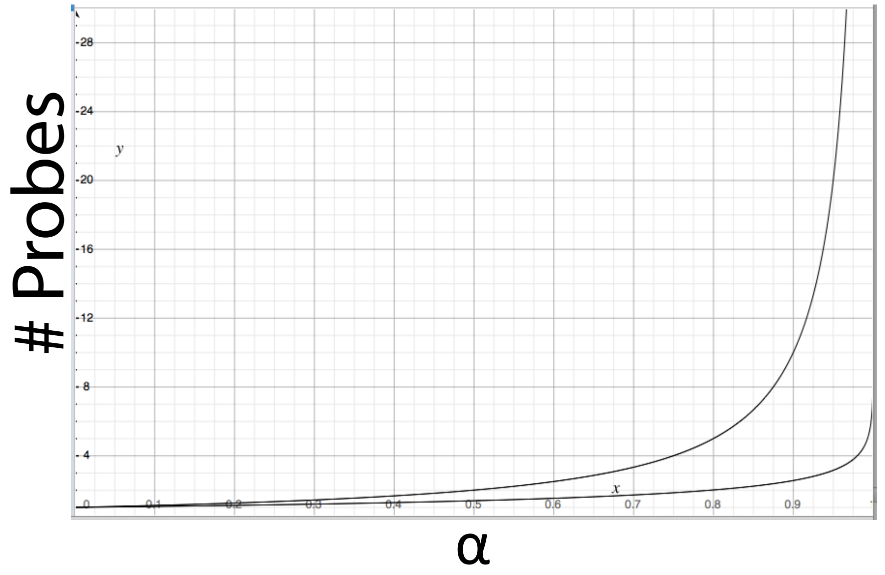
- Successful: **$\frac{1}{2}(1 + 1/(1-\alpha))$**
- Unsuccessful: **$\frac{1}{2}(1 + 1/(1-\alpha))^2$**

Double Hashing:

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

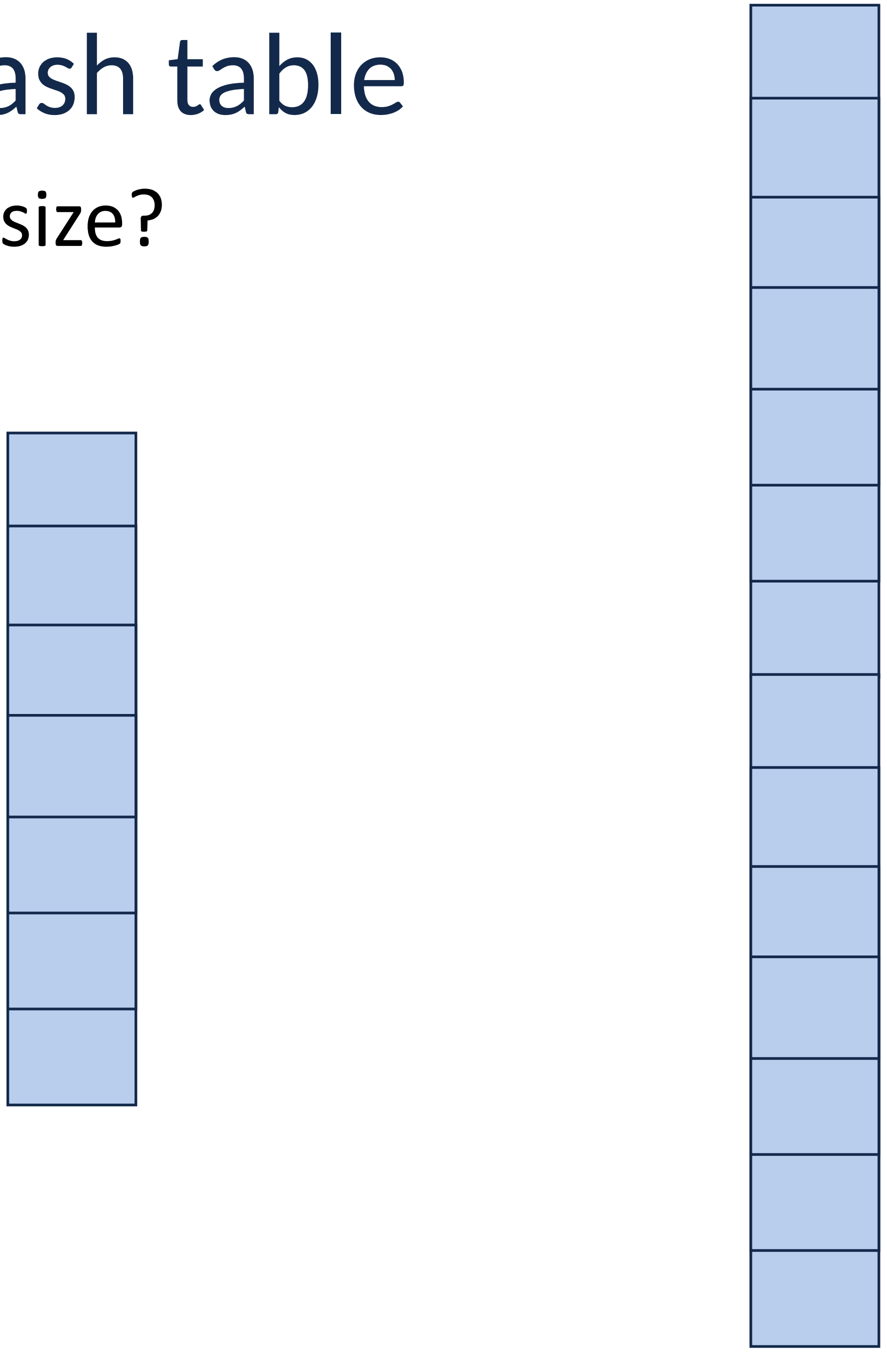
When do we resize?





Resizing a hash table

How do we 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			

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~~

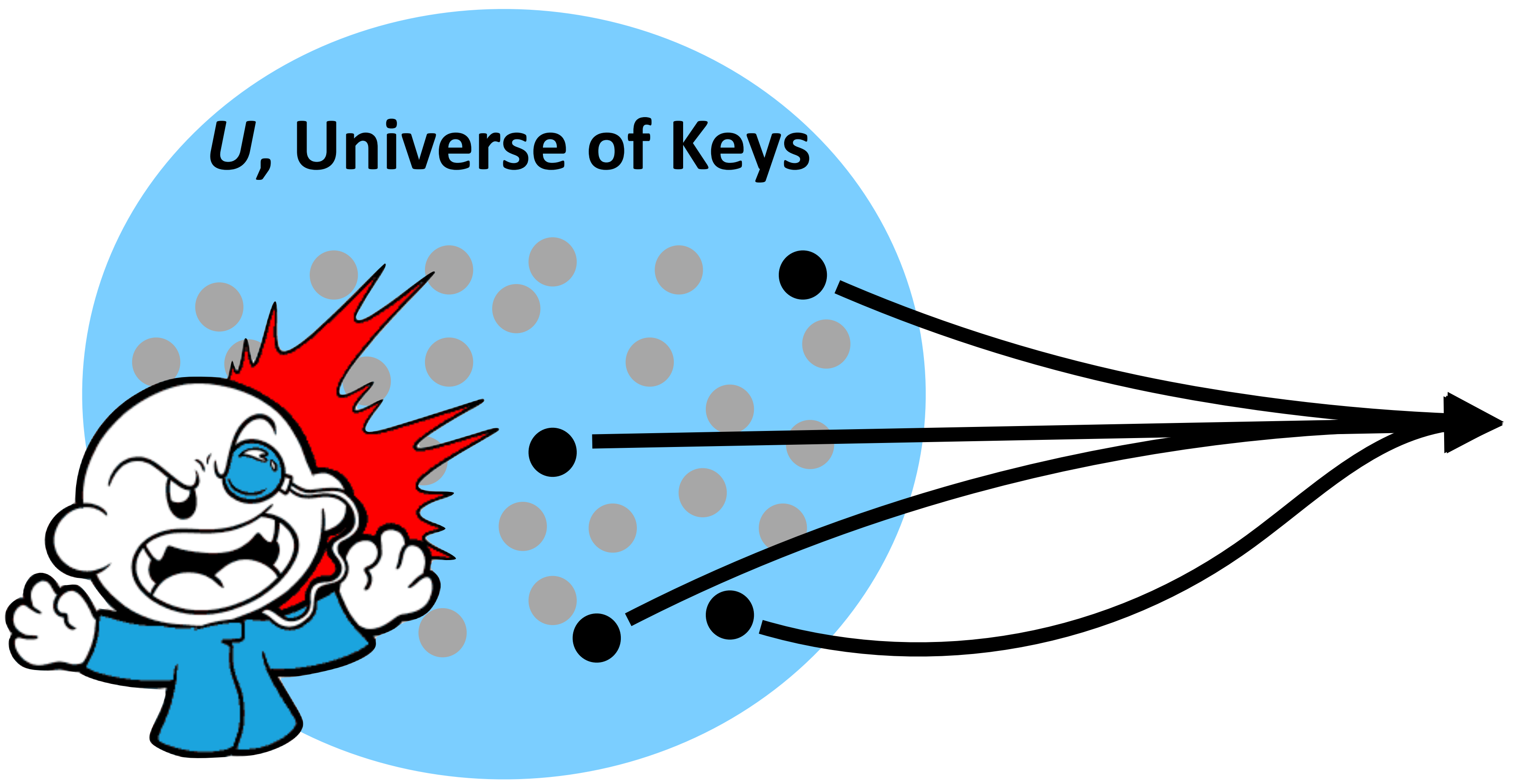
~~::upper_bound(key) ➔ Iterator to first element > key~~

::load_factor()

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

Hashing in the real world

Even under SUHA, our estimates are *in expectation*.



m **elements**

Key	Value

Hash Function (Division Method)

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

Hash Function (Mid-Square Method)

Hash of form: $h(k) = (k * k)$ and take b middle bits where $m = 2^b$

Hash Function (Multiplication Method)

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

Hash Function (Universal Hash Family)

Pick a random $h \in H$ s.t. $\forall k_1, k_2 \in U,\ \ Pr(h[k_1] = h[k_2]) \leq \frac{1}{m}$

Hash Function (Universal Hash Family)

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

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

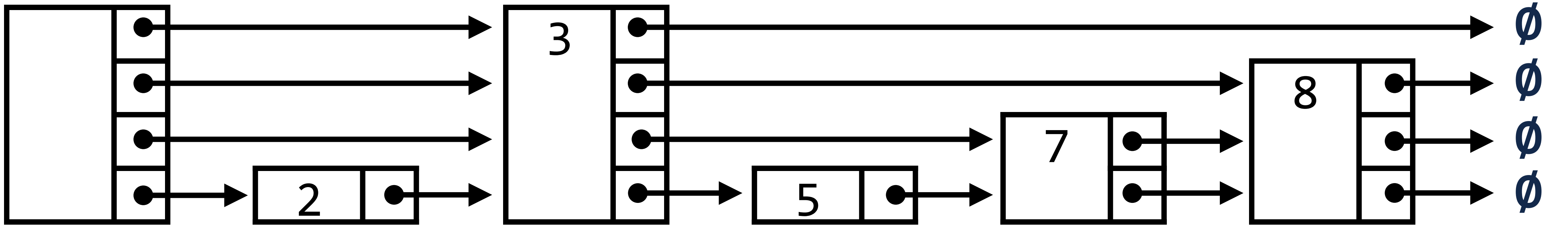
The Skip List

An ordered linked list where each node has variable size

Each node has at most one key but an arbitrary number of pointers

The decision for height is **randomized**

Claim: The **expected** time to insert, search, or delete is $O(log\ n)$



Skip List Expectation

Lets assume our skip list uses a coin flip for randomness `(c=0.5)`

Claim: Expected size of a node is 2.

Skip List Expectation

Lets assume our skip list uses a coin flip for randomness `(c=0.5)`

Claim: Expected size of skip list is 2n.

Skip List Expectation

Claim: Expected height of skip list is $O(log\ n)$

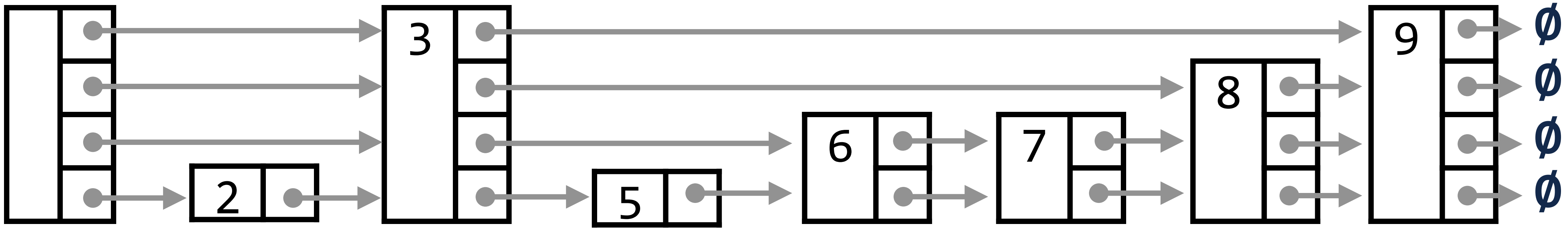
Skip List Expectation

Claim: Expected height of skip list is $O(log\ n)$

$$E[h] = \sum_{l=0}^{\lceil log\ n \rceil} E[I_l] + \sum_{l=\lceil log\ n \rceil+1}^{\infty} E[I_l] \qquad I_l = \begin{cases} 1 & l\text{th level contains a node} \\ 0 & l\text{th level empty} \end{cases}$$

Skip List Expectation

Claim: Expected length of search of skip list is $O(log\ n)$



2
3
5
6
7
8
9
Ø
Ø
Ø
Ø