

Hash Tables

CSE1303 Part A

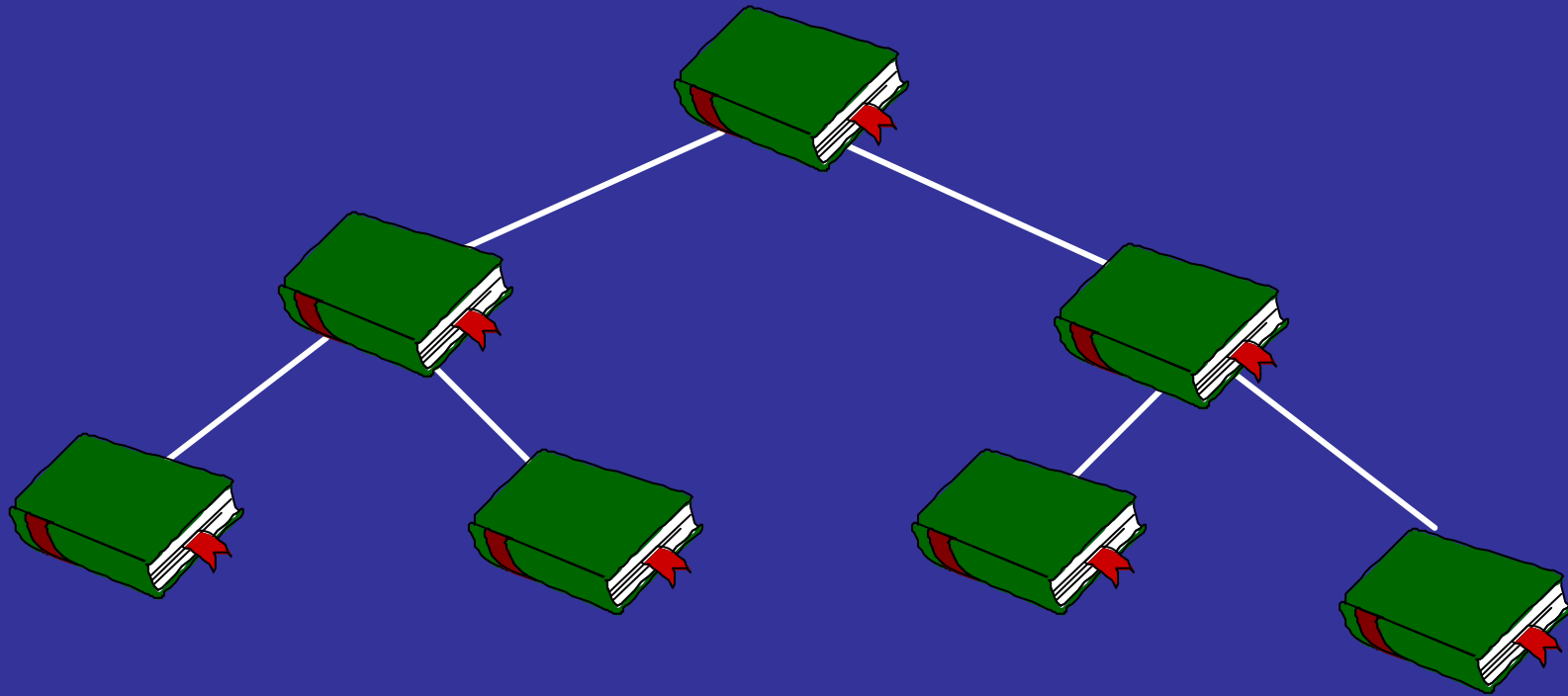
Data Structures and Algorithms

Overview

- Information Retrieval
- Review: Binary Search Trees
- Hashing.
- Applications.
- Example.
- Hash Functions.

Example: Bibliography

- R. **Kruse**, C. Tondo, B. Leung, “*Data Structures and Program Design in C*”, 1991, Prentice Hall.
- E. **Horowitz**, S. Salini, S. Anderson-Freed, “*Fundamentals of Data Structures in C*”, 1993, Computer Science Press.
- R. **Sedgewick**, “*Algorithms in C*”, 1990, Addison-Wesley.
- A. **Aho**, J. Hopcroft, J. Ullman, “*Data Structures and Algorithms*”, 1983, Addison-Wesley.
- T.A. **Standish**, “*Data Structures, Algorithms & Software Principles in C*”, 1995, Addison-Wesley.
- D. **Knuth**, “*The Art of Computer Programming*”, 1975, Addison-Wesley.
- Y. **Langsam**, M. Augenstein, M. Fenenbaum, “*Data Structures using C and C++*”, 1996, Prentice Hall.



Insert the information into a Binary Search Tree,
using the first author's surname as the key

Kruse

Horowitz

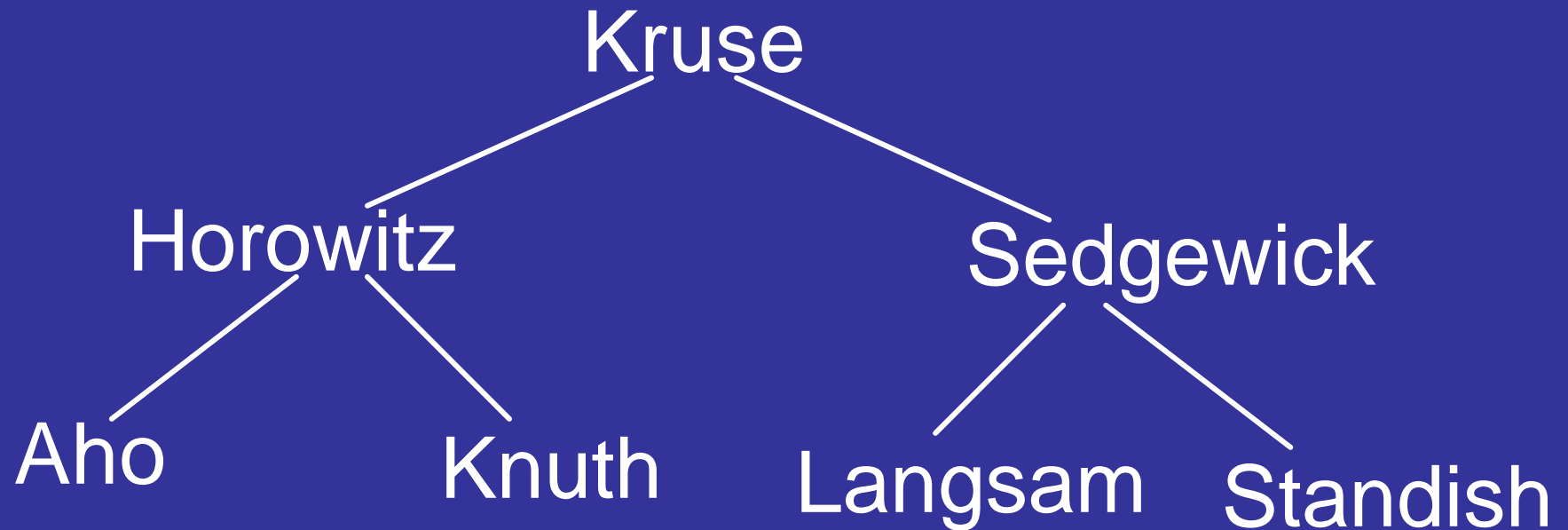
Sedgewick

Aho

Knuth

Langsam

Standish

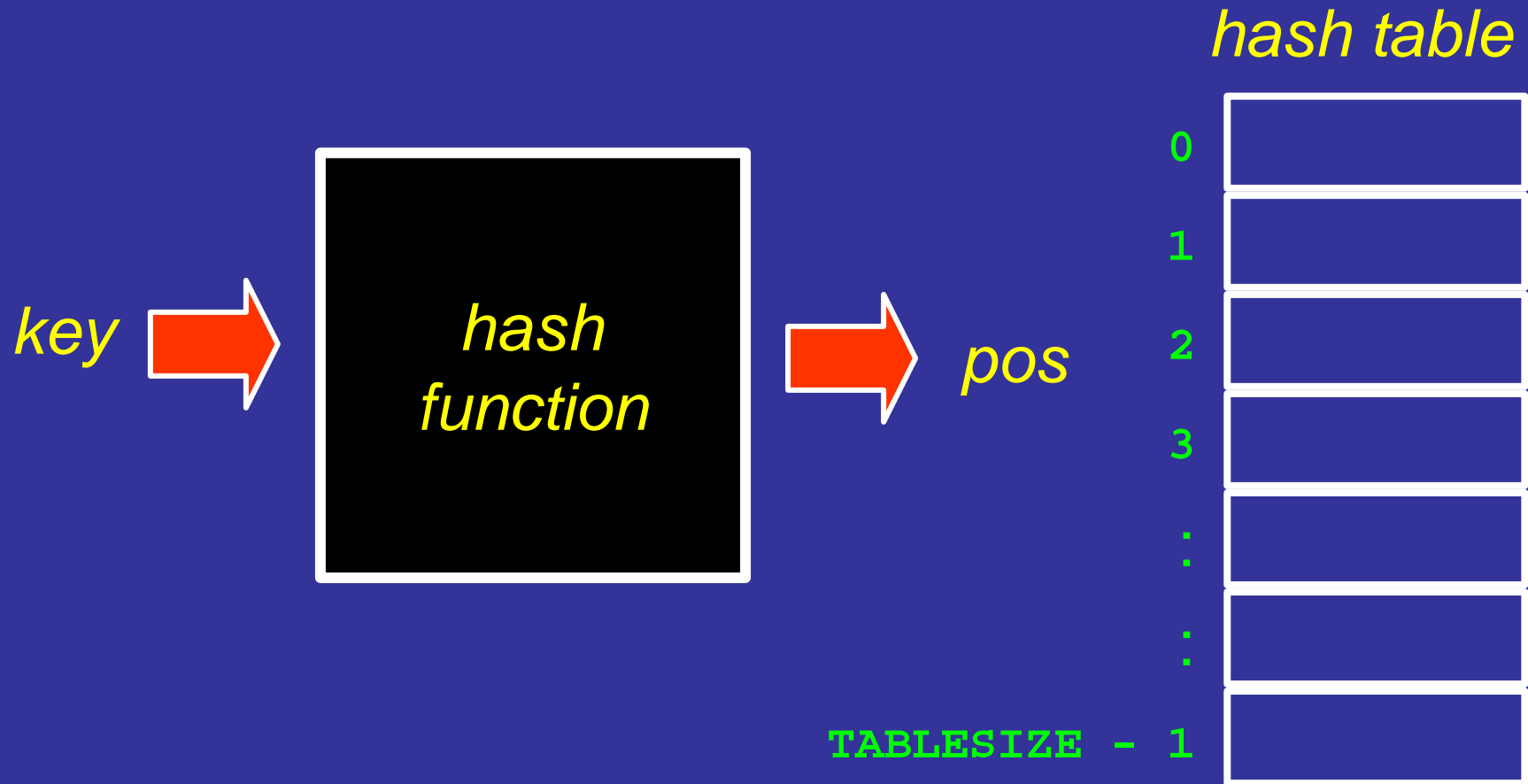


Insert the information into a Binary Search Tree,
using the first author's surname as the key

Complexity

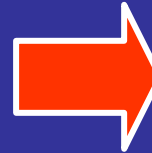
- **Inserting**
 - Balanced Trees **$O(\log(n))$**
 - Unbalanced Trees **$O(n)$**
- **Searching**
 - Balanced Trees **$O(\log(n))$**
 - Unbalanced Trees **$O(n)$**

Hashing



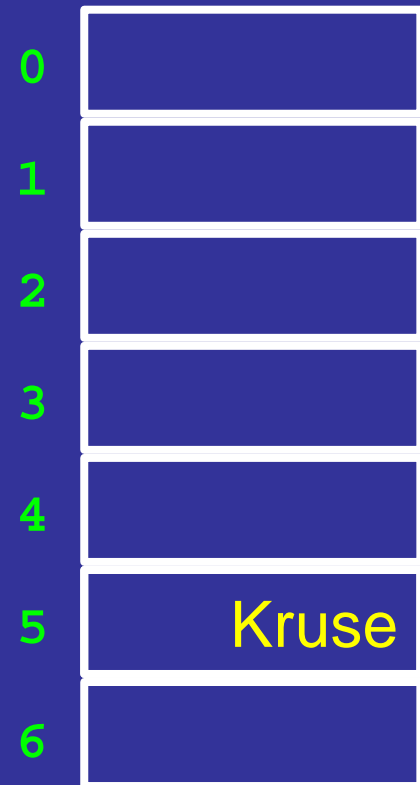
Example:

“Kruse”



5

hash table



Hashing

- Each item has a **unique key**.
- Use a large **array** called a **Hash Table**.
- Use a **Hash Function**.

Applications

- Databases.
- Spell checkers.
- Computer chess games.
- Compilers.

Operations

- Initialize
 - all locations in Hash Table are **empty**.
- Insert
- Search
- Delete

Hash Function

- **Maps** keys to positions in the Hash Table.
- Be **easy** to calculate.
- Use **all** of the key.
- Spread the keys **uniformly**.

Example: Hash Function #1

```
unsigned
hash(char* s)
{
    int i = 0;
    unsigned value = 0;

    while (s[i] != '\0')
    {
        value = (s[i] + 31*value) % 101;
        i++;
    }
    return value;
}
```

Example: Hash Function #1

$\text{value} = (\text{s}[i] + 31 * \text{value}) \% 101;$

- A. **Aho**, J. Hopcroft, J. Ullman, “*Data Structures and Algorithms*”, 1983, Addison-Wesley.

'A' = 65

'h' = 104

'o' = 111

$\text{value} = (65 + 31 * 0) \% 101 = 65$

$\text{value} = (104 + 31 * 65) \% 101 = 99$

$\text{value} = (111 + 31 * 99) \% 101 = 49$

Example: Hash Function #1

$value = (s[i] + 31 * value) \% 101;$

<u>Key</u>	<u>Hash Value</u>
Aho	49
Kruse	95
Standish	60
Horowitz	28
Langsam	21
Sedgewick	24
Knuth	44

*resulting
table is
“sparse”*

Example: Hash Function #2

$\text{value} = (\text{s}[i] + 1024 * \text{value}) \% 128;$

Aho	111
Kruse	101
Standish	104
Horowitz	122
Langsam	109
Sedgewick	107
Knuth	104

*likely to
result in
“clustering”*

Example: Hash Function #3

$$\text{value} = (\text{s}[i] + 3 * \text{value}) \% 7;$$

Aho	0
Kruse	5
Standish	1
Horowitz	5
Langsam	5
Sedgewick	2
Knuth	1

“collisions”

Insert

- Apply hash function to get a position.
- Try to insert key at this position.
- Deal with collision.

Example: Insert

Aho, Kruse, Standish, Horowitz, Langsam, Sedgewick, Knuth

hash table

Aho



0

0	Aho
1	
2	
3	
4	
5	
6	

Example: Insert

Aho, Kruse, Standish, Horowitz, Langsam, Sedgewick, Knuth

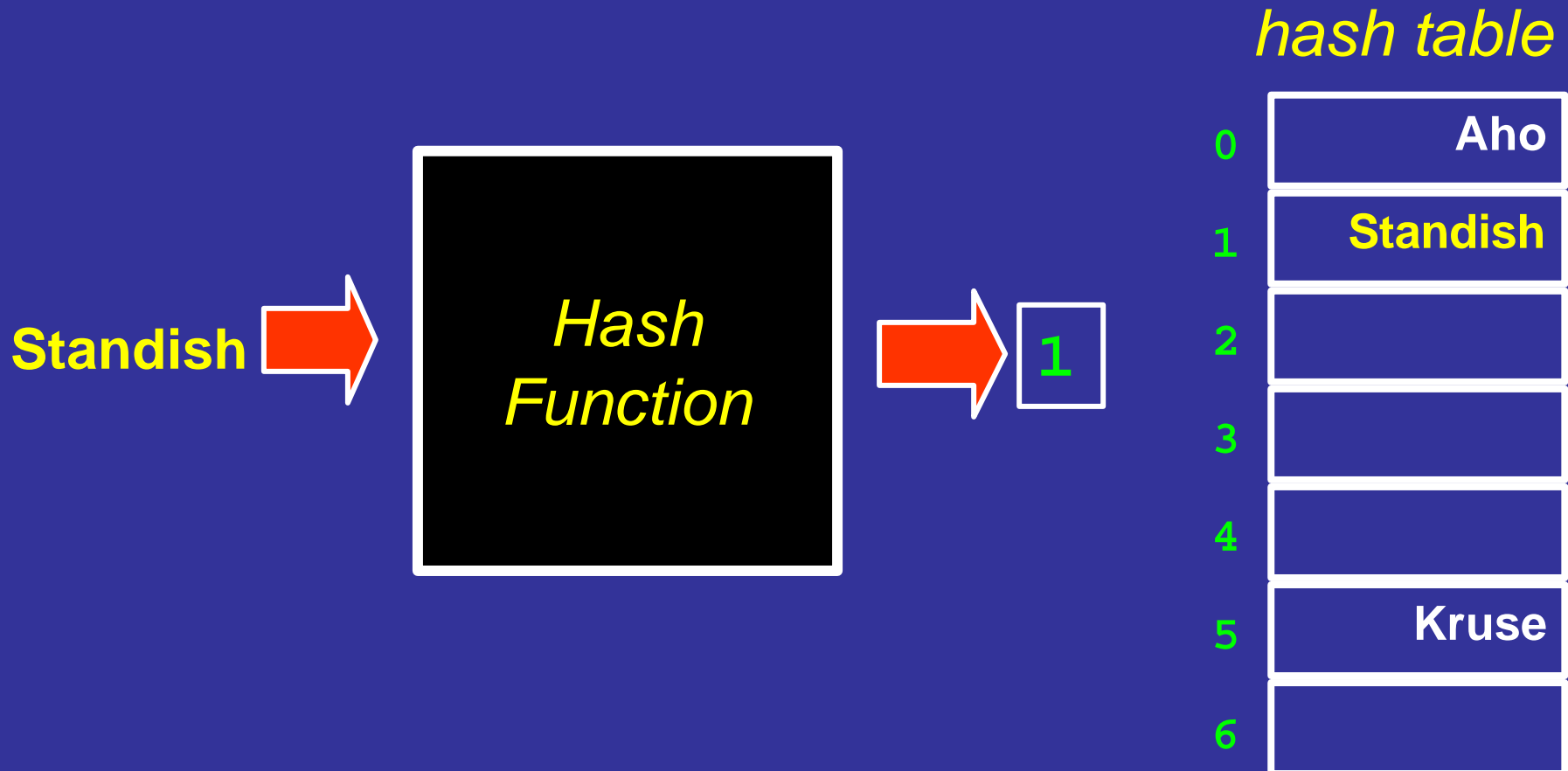
hash table



0	Aho
1	
2	
3	
4	
5	Kruse
6	

Example: Insert

Aho, Kruse, Standish, Horowitz, Langsam, Sedgewick, Knuth



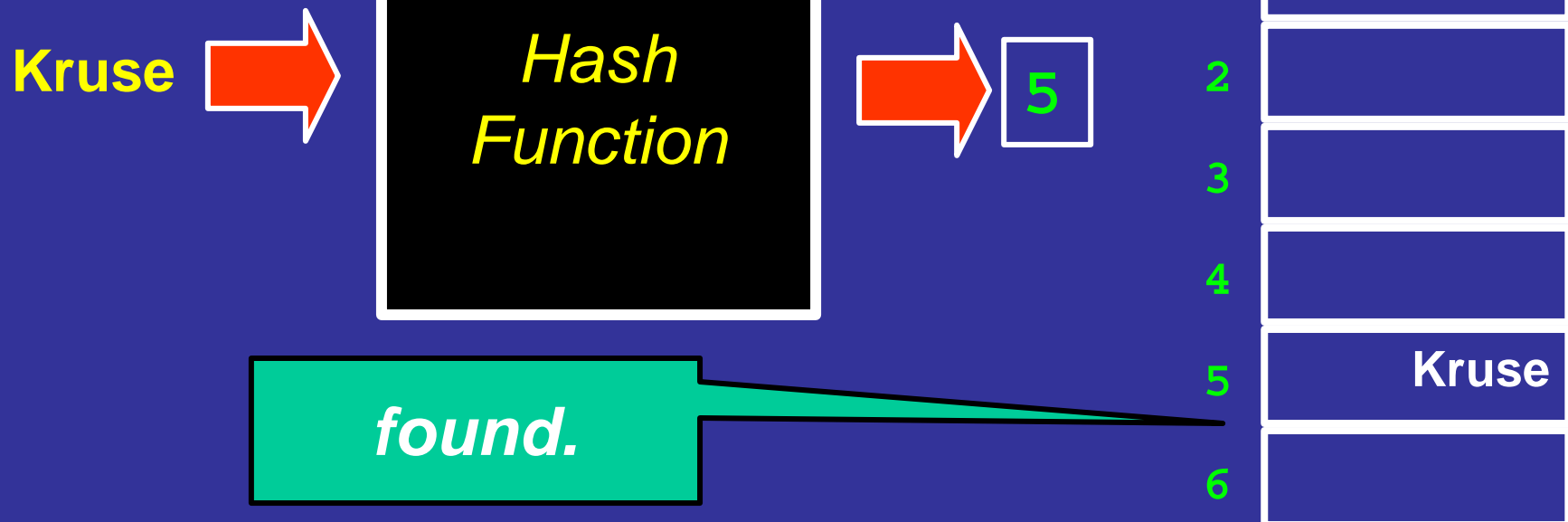
Search

- Apply hash function to get a position.
- Look in that position.
- Deal with collision.

Example: Search

Aho, Kruse, Standish, Horowitz, Langsam, Sedgewick, Knuth

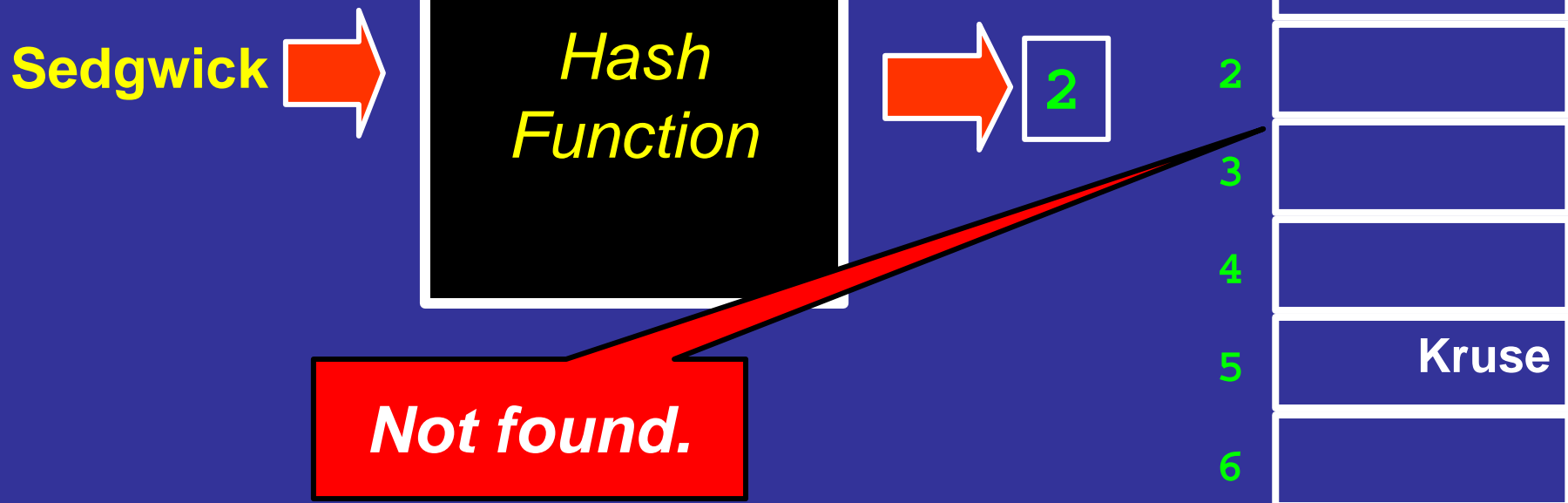
hash table



Example: Search

Aho, Kruse, Standish, Horowitz, Langsam, Sedgewick, Knuth

hash table



Revision

- Hash Tables
 - Hash Functions
 - Insert, Search

Preparation

- Read Chapter 8 in Kruse et al.