

**CSE1303 Part A**  
**Data Structures and Algorithms**  
**Summer Semester 2003**

**Lecture A14 – Hash Tables**

Kymerly Fergusson

**Overview**

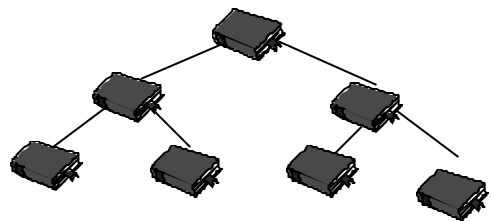
- Information Retrieval
- Review: Binary Search Trees
- Hashing.
- Applications.
- Example.
- Hash Functions.

2

**Example: Bibliography**

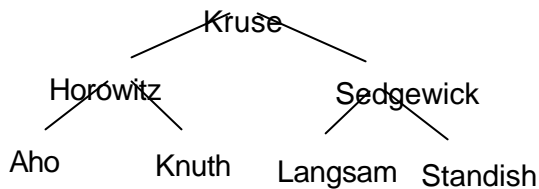
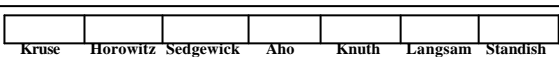
- R. **Kruse**, C. Tondo, B. Leung, "Data Structures and Program Design in C", 1991, Prentice Hall.
- E. **Horowitz**, S. Salini, S. Anderson-Freed, "Fundamentals of Data Structures in C", 1993, Computer Science Press.
- R. **Sedgewick** "Algorithms in C", 1990, Addison-Wesley.
- A. **Aho**, J. Hopcroft, 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. Fenenaub, "Data Structures using C and C++", 1996, Prentice Hall.

3



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

4



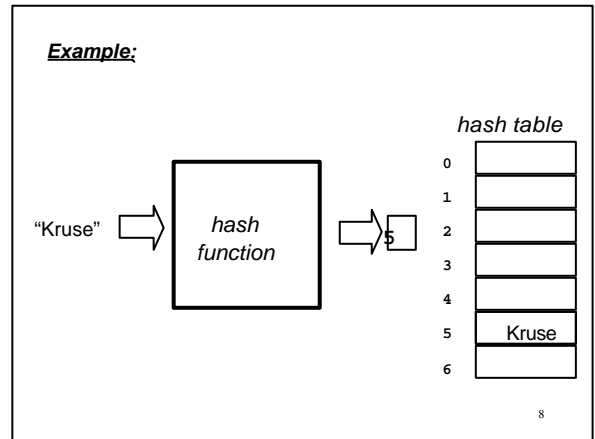
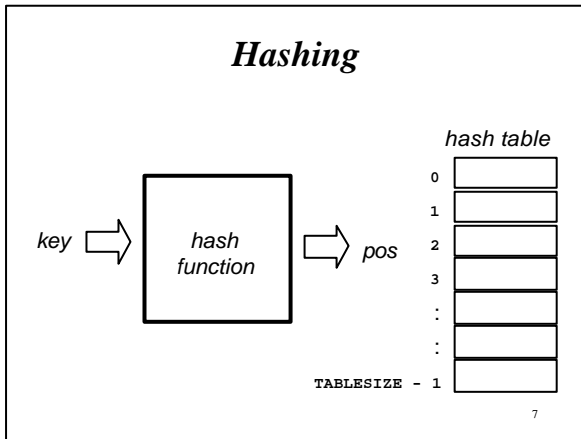
Insert the information into a Binary Search Tree, using the first author's surname as the key

5

**Complexity**

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

6



- ### *Hashing*
- Each item has a unique key.
  - Use a large array called a Hash Table.
  - Use a Hash Function.
- 9

- ### *Applications*
- Databases.
  - Spell checkers.
  - Computer chess games.
  - Compilers.
- 10

- ### *Operations*
- Initialize
    - all locations in Hash Table are empty.
  - Insert
  - Search
  - Delete
- 11

- ### *Hash Function*
- Maps keys to positions in the Hash Table.
  - Be easy to calculate.
  - Use all of the key.
  - Spread the keys uniformly.
- 12

**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;
}
    
```

13

**Example: Hash Function #1**

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

- A. Aho, J. Hopcroft, J. Ullman, "Data Structures and Algorithms", 1983, Addison-Wesley.

'A' = 65      'h' = 104      'o' = 111

```

value = (65 + 31 * 0) % 101 = 65
value = (104 + 31 * 65) % 101 = 99
value = (111 + 31 * 99) % 101 = 49
    
```

14

**Example: Hash Function #1**

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

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

resulting table is "sparse"

15

**Example: Hash Function #2**

value = (s[i] + 1024\*value) % 128;

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

likely to result in "clustering"

16

**Example: Hash Function #3**

value = (s[i] + 3\*value) % 7;

Key	Hash Value
Aho	0
Kruse	5
Standish	1
Horowitz	5
Langsam	5
Sedgewick	2
Knuth	1

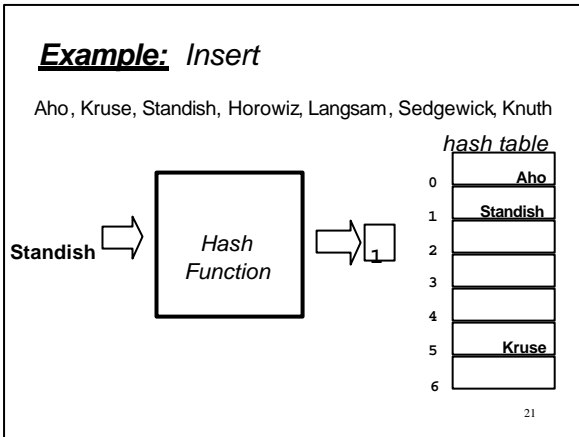
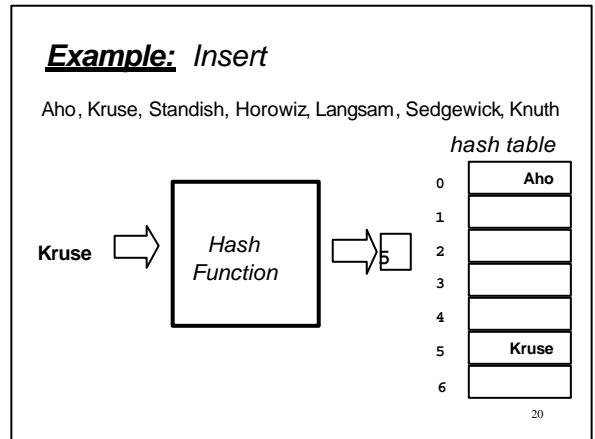
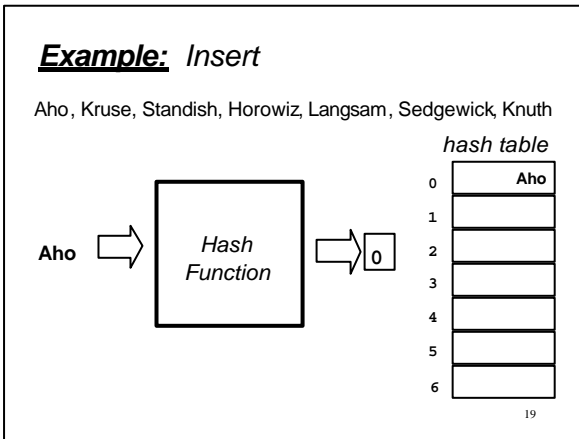
"collisions"

17

**Insert**

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

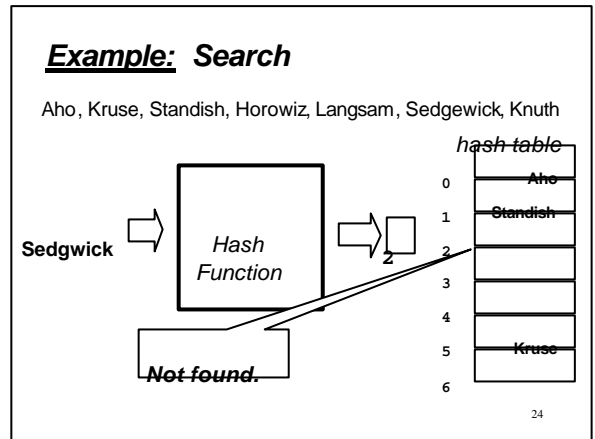
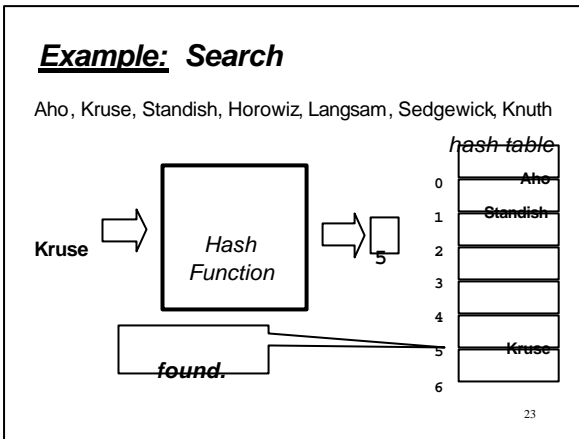
18



**Search**

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

22



### ***Revision***

- Hash Tables
  - Hash Functions
  - Insert, Search

### ***Revision: Reading***

- Kruse 8
- Standish 11
- Langsam 7.4

### ***Preparation***

*Next lecture: Hash Tables: Collision Detection*

- Read Chapter 8 in Kruse et al.

25