Today’s Outline

• Announcements
  – Project 2B due Wednesday, 2/10 at 11pm
  – Midterms returned and discussed in section Thurs
  – Written Homework #4 due Friday 2/12

• Today’s Topics:
  – Hash Tables

Hash Tables

Chapter 5 in Weiss

CSE 326
Data Structures
Ruth Anderson

Today’s Outline

• Announcements
  – Project 2B due Wednesday, 2/10 at 11pm
  – Midterms returned and discussed in section Thurs
  – Written Homework #4 due Friday 2/12

• Today’s Topics:
  – Hash Tables

Hash Tables

• Constant time accesses!

• A hash table is an array of some fixed size, usually a prime number.

• General idea:

  hash function: \( h(K) \)

  key space (e.g., integers, strings) \( \rightarrow \) TableSize – 1

Information Retrieval

Implementations So Far

(Only showing keys, but leaves also have data!)

B-Tree with \( M = 4 \)
and \( L = 4 \)

Perform Insert(34)

Student Activity
Example

- key space = integers
- TableSize = 10
- \( h(K) = K \mod 10 \)
- **Insert**: 7, 18, 41, 94

Another Example

- key space = integers
- TableSize = 6
- \( h(K) = K \mod 6 \)
- **Insert**: 7, 18, 41, 34, 32

Hash Functions

1. **simple/fast** to compute,
2. Avoid collisions
3. have keys distributed *evenly* among cells

Perfect Hash function:

Sample Hash Functions:

- key space = strings
- \( s = s_0 \ s_1 \ s_2 \ \ldots \ s_{k-1} \)

1. \( h(s) = s_0 \mod \text{TableSize} \)
2. \( h(s) = \left( \sum_{i=0}^{k-1} s_i \right) \mod \text{TableSize} \)
3. \( h(s) = \left( \sum_{i=0}^{k-1} s_i \cdot 37^i \right) \mod \text{TableSize} \)

Designing a Hash Function for web URLs

\( s = s_0 \ s_1 \ s_2 \ \ldots \ s_{k-1} \)

Issues to take into account:

\[ h(s) = \]

Collision Resolution

**Collision**: when two keys map to the same location in the hash table.

Two ways to resolve collisions:
1. Separate Chaining
2. Open Addressing (linear probing, quadratic probing, double hashing)
Separate Chaining

- **Separate chaining:**
  All keys that map to the same hash value are kept in a list (or "bucket").

Insert:
- 10
- 22
- 107
- 12
- 42

Analysis of find

- **Defn:** The load factor, $\lambda$, of a hash table is the ratio:
  $$\frac{N}{M} \leftarrow \text{no. of elements}$$
  $$\text{table size}$$
  For separate chaining, $\lambda = \text{average # of elements in a bucket}$

How big should the hash table be?

- For Separate Chaining:

Insert:
- 38
- 19
- 8
- 109
- 10

Open Addressing

- **Linear Probing:**
  after checking spot $h(k)$, try spot $h(k) + 1$, if that is full, try $h(k) + 2$, then $h(k) + 3$, etc.

tableSize: Why Prime?

- **Suppose**
  - data stored in hash table: 7160, 493, 60, 55, 321, 900, 810
  - tableSize = 10
    data hashes to 0, 3, 0, 5, 1, 0, 0
  - tableSize = 11
    data hashes to 10, 9, 5, 2, 9, 7

Real-life data tends to have a pattern

Being a multiple of 11 is usually not the pattern :)

Terminology Alert!

- **Open Hashing**
  equals
  “Open Addressing”

- **Closed Hashing**
  equals
  “Separate Chaining”

Weiss
Linear Probing

\[ f(i) = i \]

- Probe sequence:
  
  0\textsuperscript{th} probe = \( h(k) \mod \text{TableSize} \)
  
  1\textsuperscript{st} probe = \( (h(k) + 1) \mod \text{TableSize} \)
  
  2\textsuperscript{nd} probe = \( (h(k) + 2) \mod \text{TableSize} \)
  
  \ldots
  
  \iota\textsuperscript{th} probe = \( (h(k) + \iota) \mod \text{TableSize} \)

Load Factor in Linear Probing

- For any \( \lambda < 1 \), linear probing will find an empty slot
- Expected # of probes (for large table sizes)
  
  - successful search:
    \[
    \frac{1}{2} \left( 1 + \frac{1}{(1 - \lambda)} \right)
    \]
  
  - unsuccessful search:
    \[
    \frac{1}{2} \left( 1 + \frac{1}{(1 - \lambda)^2} \right)
    \]

- Linear probing suffers from \textit{primary clustering}
- Performance quickly degrades for \( \lambda > 1/2 \)

Quadratic Probing

\[ f(i) = i^2 \]

- Probe sequence:
  
  0\textsuperscript{th} probe = \( h(k) \mod \text{TableSize} \)
  
  1\textsuperscript{st} probe = \( (h(k) + 1) \mod \text{TableSize} \)
  
  2\textsuperscript{nd} probe = \( (h(k) + 4) \mod \text{TableSize} \)
  
  3\textsuperscript{rd} probe = \( (h(k) + 9) \mod \text{TableSize} \)
  
  \ldots
  
  \iota\textsuperscript{th} probe = \( (h(k) + \iota^2) \mod \text{TableSize} \)

Quadratic Probing:

- \( h(k) = k \mod 7 \)
- Perform these inserts:
  
  - Insert(65)
  
  - Insert(10)
  
  - Insert(47)

Quadratic Probing:

- \( h(k) = k \mod 7 \)
- Perform these inserts:
  
  - Insert(65)
  
  - Insert(10)
  
  - Insert(47)