Introduction to Double Hash
Recitation #3

Zhexiong Liu
zhexiong@cs.pitt.edu

Department of Computer Science
University of Pittsburgh

September 18, 2020
Slides adapted from Shun Yan Cheung
Introduction of hashing

- A hash function maps keys (arbitrary values) to integers in (0, N-1), e.g. hash function \( h(k) = k \% 10 \)
- A data structure that can map keys to these integers called a hash table
- Use of a hash function to index a hash table is called hashing
- Hashing provides \( O(1) \) time for search, insert, delete, and update
Structure of hashing

A hash function usually consists of two parts

- A **hash code** that maps a key (arbitrary value, such as integer, float, or string) to an (unbounded) integer
- A **compression function** that maps an arbitrary value to an integer in \((0, N-1)\)
Desired property for hash codes

- Different keys should map to different values
- The same keys should map to the same value
- Avoid collisions that occur when two different keys map to the same hash value

Key1 -> Value

Key2 -> Value

Collision!
Handling collisions in hashing

- Closed Addressing
  - Separate Chaining
- Open Addressing
  - Linear Probing
  - Quadratic Probing
  - Double Hashing (Rehashing)
The separate chaining

Each hash bucket contains a linked list of entries.

Note that a linked list can be arbitrarily long, so one bucket can contain an arbitrary large number of entries.

- hash function $h(k) = k \% 5$ with 5 buckets

(1, A) (2, B) (17, C) (23, D) (28, E) (9, F)
The separate chaining

Each hash bucket contains a linked list of entries.

Note that a linked list can be arbitrarily long, so one bucket can contain an arbitrary large number of entries.

- hash function \( h(k) = k \% 5 \) with 5 buckets

\[
(1, A) \quad (2, B) \quad (17, C) \quad (23, D) \quad (28, E) \quad (9, F)
\]
The separate chaining

Each hash bucket contains a linked list of entries

Note that a linked list can be arbitrarily long, so one bucket can contain an arbitrary large number of entries

- hash function $h(k) = k \% 5$ with 5 buckets

(1, A) (2, B) (17, C) (23, D) (28, E) (9, F)
The separate chaining

Each hash bucket contains a linked list of entries

Note that a linked list can be arbitrarily long, so one bucket can contain an arbitrary large number of entries

- hash function \( h(k) = k \% 5 \) with 5 buckets

<table>
<thead>
<tr>
<th>Bucket 0</th>
<th>Bucket 1</th>
<th>Bucket 2</th>
<th>Bucket 3</th>
<th>Bucket 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, A)</td>
<td>(2, B)</td>
<td>(17, C)</td>
<td>(23, D)</td>
<td>(28, E)</td>
</tr>
<tr>
<td>(1, A)</td>
<td>(2, B)</td>
<td>(17, C)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The separate chaining

Each hash bucket contains a linked list of entries

Note that a linked list can be arbitrarily long, so one bucket can contain an arbitrary large number of entries

- hash function $h(k) = k \% 5$ with 5 buckets

(1, A)  (2, B)  (17, C)  (23, D)  (28, E)  (9, F)

Bucket 0  Bucket 1  Bucket 2  Bucket 3  Bucket 4

(1, A)  (2, B)  (23, D)  (28, E)

(17, C)
The separate chaining

Each hash bucket contains a linked list of entries

Note that a linked list can be arbitrarily long, so one bucket can contain an arbitrary large number of entries

- hash function $h(k) = k \mod 5$ with 5 buckets

$(1, A) \quad (2, B) \quad (17, C) \quad (23, D) \quad (28, E) \quad (9, F)$
Separate chaining

- Chaining is simple to implement
- Hash table never fills up so that the linked list can be arbitrarily long
- Wastage of space as some parts of hash table in chaining are never used or used only once
Linear probing

The location for the i-th probe is \( h(k) + i \)

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function \( h(k) = k \% 5 \)

\[
\begin{align*}
(1, A) & \quad (2, B) & \quad (17, C) & \quad (7, D) & \quad (23, E)
\end{align*}
\]

Bucket 0 \quad Bucket 1 \quad Bucket 2 \quad Bucket 3 \quad Bucket 4 \quad Bucket 5 \quad Bucket 6

(1, A)
Linear probing

The location for the i-th probe is $h(k) + i$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

<table>
<thead>
<tr>
<th></th>
<th>Bucket 0</th>
<th>Bucket 1</th>
<th>Bucket 2</th>
<th>Bucket 3</th>
<th>Bucket 4</th>
<th>Bucket 5</th>
<th>Bucket 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, A)</td>
<td>(2, B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Linear probing

The location for the i-th probe is $h(k) + i$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

$(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)$

Collision!

Bucket 0   Bucket 1   Bucket 2   Bucket 3   Bucket 4   Bucket 5   Bucket 6

(1, A)    (2, B)    (17, C)    (7, D)    (23, E)
Linear probing

The location for the i-th probe is $h(k) + i$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

(1, A) (2, B) (17, C) (7, D) (23, E)

Bucket 0 Bucket 1 Bucket 2 Bucket 3 Bucket 4 Bucket 5 Bucket 6

(1, A) (2, B) (17, C)
Linear probing

The location for the i-th probe is $h(k) + i$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

(1, A) (2, B) (17, C) (7, D) (23, E)

Collision!

Bucket 0 Bucket 1 Bucket 2 Bucket 3 Bucket 4 Bucket 5 Bucket 6

(1, A) (2, B) (17, C)
Linear probing

The location for the i-th probe is \( h(k) + i \)

- Insert \((1, A), (2, B), (17, C), (7, D), (23, E)\)
- Hash function \( h(k) = k \% 5 \)

\[
\begin{array}{cccccccc}
(1, A) & (2, B) & (17, C) & (7, D) & (23, E) \\
\hline
\end{array}
\]
Linear probing

The location for the i-th probe is $h(k) + i$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

$(1, A) \ (2, B) \ (17, C) \ (7, D) \ (23, E)$
Linear probing

The location for the i-th probe is $h(k) + i$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

$(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)$

Bucket 0   Bucket 1   Bucket 2   Bucket 3   Bucket 4   Bucket 5   Bucket 6

(1, A)   (2, B)   (17, C)   (7, D)   (23, E)

Linearly increase i
Quadratic probing

The location for the i-th probe is $h(k) + i^2$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

$(1, A) \ (2, B) \ (17, C) \ (7, D) \ (23, E)$

<table>
<thead>
<tr>
<th>Bucket 0</th>
<th>Bucket 1</th>
<th>Bucket 2</th>
<th>Bucket 3</th>
<th>Bucket 4</th>
<th>Bucket 5</th>
<th>Bucket 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, A)</td>
<td>(2, B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Quadratic probing

The location for the i-th probe is \( h(k) + i^2 \)

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function \( h(k) = k \% 5 \)

\[
\begin{array}{ccccccc}
(1, A) & (2, B) & (17, C) & (7, D) & (23, E) \\
\end{array}
\]

Collision!
Quadratic probing

The location for the i-th probe is $h(k) + i^2$

- Insert $(1, A), (2, B), (17, C), (7, D), (23, E)$
- Hash function $h(k) = k \% 5$

$(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)$
Quadratic probing

The location for the i-th probe is \( h(k) + i^2 \)

- Insert \((1, A), (2, B), (17, C), (7, D), (23, E)\)
- Hash function \( h(k) = k \% 5 \)

\[
\begin{align*}
(1, A) & (2, B) & (17, C) & (7, D) & (23, E) \\
\end{align*}
\]

Collision!
Quadratic probing

The location for the i-th probe is $h(k) + i^2$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

(1, A) (2, B) (17, C) (7, D) (23, E)
**Quadratic probing**

The location for the $i$-th probe is $h(k) + i^2$

- Insert $(1, A), (2, B), (17, C), (7, D), (23, E)$
- Hash function $h(k) = k \% 5$

$(1, A) (2, B) (17, C) (7, D) (23, E)$
Quadratic probing

The location for the i-th probe is $h(k) + i^2$

- Insert (1, A), (2, B), (17, C), (7, D), (23, E)
- Hash function $h(k) = k \% 5$

$(1, A) (2, B) (17, C) (7, D) (23, E)$

Collision!
Quadratic probing

The location for the i-th probe is $h(k) + i^2$

- Insert $(1, A), (2, B), (17, C), (7, D), (23, E)$
- Hash function $h(k) = k \% 5$

$(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)$
Linear and quadratic probing issue

- Waste of time for searching
- May cause clustering issue, say it starts taking time to visit many consecutive elements in order to find a free bucket

\[(1, A) \ (2, B) \ (17, C) \ (7, D) \ (23, E)\]

Bucket 0  Bucket 1  Bucket 2  Bucket 3  Bucket 4  Bucket 5  Bucket 6

---

Time consuming!
Double hashing

Uses 2 different hash functions: $h_1(k)$ and $h_2(k)$

- $h_1(k)$ is used to find the hash location
- $h_2(k)$ is used to probe
  - $h_2(k)$ cannot produce zero!
  - Commonly used $h_2(k) = q - (k \% q)$ where $q$ is a prime number less than $N$
Double hashing

Uses 2 different hash functions: \( h_1(k) \) and \( h_2(k) \)

- Insert \((1, A), (2, B), (17, C), (7, D), (23, E)\)
- Suppose \( h_1(k) = k \mod 5, h_2(k) = 3 - k \mod 3 \)

\[
\begin{align*}
(1, A) & \quad (2, B) & \quad (17, C) & \quad (7, D) & \quad (23, E)
\end{align*}
\]

Collision!
Double hashing

Uses 2 different hash functions: $h_1(k)$ and $h_2(k)$

- Insert (1, A), (2, B), (17, C), (7, D), (22, E)
- Suppose $h_1(k) = k \% 5$, $h_2(k) = 3 - k \% 3$

<table>
<thead>
<tr>
<th>(1, A)</th>
<th>(2, B)</th>
<th>(17, C)</th>
<th>(7, D)</th>
<th>(23, E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket 0</td>
<td>Bucket 1</td>
<td>Bucket 2</td>
<td>Bucket 3</td>
<td>Bucket 4</td>
</tr>
</tbody>
</table>

Probe $h_2(k) = 3 - 17 \% 3 = 1$
Double hashing

Uses 2 different hash functions: \( h_1(k) \) and \( h_2(k) \)

- Insert \((1, A), (2, B), (17, C), (7, D), (22, E)\)
- Suppose \( h_1(k) = k \% 5, h_2(k) = 3 - k \% 3 \)

\((1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)\)

Collision!
### Double hashing

Uses 2 different hash functions: \( h_1(k) \) and \( h_2(k) \)

- Insert \((1, A), (2, B), (17, C), (7, D), (22, E)\)
- Suppose \( h_1(k) = k \mod 5 \), \( h_2(k) = 3 - k \mod 3 \)

\[
(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)
\]

<table>
<thead>
<tr>
<th>Bucket 0</th>
<th>Bucket 1</th>
<th>Bucket 2</th>
<th>Bucket 3</th>
<th>Bucket 4</th>
<th>Bucket 5</th>
<th>Bucket 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, A)</td>
<td>(2, B)</td>
<td>(17, C)</td>
<td>(7, D)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Probe \( h_2(k) = 3 - 7 \mod 3 = 2 \)
Double hashing

Uses 2 different hash functions: $h_1(k)$ and $h_2(k)$

- Insert $(1, A), (2, B), (17, C), (7, D), (22, E)$
- Suppose $h_1(k) = k \% 5$, $h_2(k) = 3 - k \% 3$

$(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)$

Collision!
Double hashing

Uses 2 different hash functions: $h_1(k)$ and $h_2(k)$

- Insert (1, A), (2, B), (17, C), (7, D), (22, E)
- Suppose $h_1(k) = k \% 5$, $h_2(k) = 3 - k \% 3$

(1, A) (2, B) (17, C) (7, D) (23, E)

Probe $h_2(k) = 3 - 23 \% 3 = 1$
Double hashing

Uses 2 different hash functions: $h_1(k)$ and $h_2(k)$

- Insert $(1, A), (2, B), (17, C), (7, D), (22, E)$
- Suppose $h_1(k) = k \% 5$, $h_2(k) = 3 - k \% 3$

$(1, A) \quad (2, B) \quad (17, C) \quad (7, D) \quad (23, E)$

Probe $h_2(k) = 3 - 23 \% 3 = 1$
Double hashing

Uses 2 different hash functions: \( h_1(k) \) and \( h_2(k) \)

- Insert \((1, A), (2, B), (17, C), (7, D), (22, E)\)
- Suppose \( h_1(k) = k \mod 5, h_2(k) = 3 - k \mod 3 \)

\[
\begin{align*}
(1, A) & \quad (2, B) & \quad (17, C) & \quad (7, D) & \quad (23, E) \\
\end{align*}
\]

Probe \( h_2(k) = 3 - 23 \mod 3 = 1 \)
Double hashing

- Use two hash functions
- Try the first one, if collide, increment by the probe calculated using the second hash function
- A nice demo here
  https://www.cs.usfca.edu/~galles/visualization/CloseHash.html