Hash Tables

CLRS Reading: Sections 11.1, 11.2, 11.3, 11.4, pages 253 -- 277

Hashtable Example

```
"Brian" hash("Brian") 12
"Stella" hash("Stella") 5
"Ellen" hash("Ellen") 7
"Lyn" hash("Lyn") 1
"Randy" hash("Randy") 8
"Sohie" hash("Sohie") 4

Pros and cons?
```

T. - 1

T. - 2
Pros and Cons

**Pros**
- Fast searching
- Fast inserting
- Fast deleting

**Cons**
- Table size is fixed (like an array)
- Collisions

Hashing

- To search for the record associated with a given key, compute the value of a *hash function* on the key.
- Use the value of the hash function as an *index* into a table of addresses.
- Two or more keys may *hash* to the same address. In such an instance, some method of *collision resolution* is employed.
Load Factor

• The load factor, $\alpha$, of a hashtable is the number of entries $n$ in the table divided by the table capacity $m$.

Collisions

• Separate chaining (linked lists):

  - “Brian” $\text{hash(“Brian”)}$ 6
  - “Stella”
  - “Ellen”
  - “Sohie”
  - “Lyn”
  - “Randy”
Collisions

• Separate chaining (linked lists):

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“Brian”</td>
<td></td>
</tr>
</tbody>
</table>

“Brian” $\text{hash(“Brian”)}$ 6

“Stella”

“Ellen”

“Lyn”

“Randy”

“Sohie”
Collisions

- **Separate chaining (linked lists):**

  - "Brian" \( \text{hash("Brian")} \rightarrow 6 \\
  - "Stella" \( \text{hash("Stella")} \rightarrow 6 \\
  - "Ellen" \\
  - "Lyn" \\
  - "Randy" \\
  - "Sohie"

  ![Diagram of separate chaining](image)

Collisions

- **Separate chaining (linked lists):**

  - "Brian" \( \text{hash("Brian")} \rightarrow 6 \\
  - "Stella" \( \text{hash("Stella")} \rightarrow 6 \\
  - "Ellen" \( \text{hash("Ellen")} \rightarrow 1 \\
  - "Lyn" \\
  - "Randy" \\
  - "Sohie"

  ![Diagram of separate chaining](image)
Collisions

• Separate chaining (linked lists):

- "Ellen" hash("Ellen") = 1
- "Stella" hash("Stella") = 6
- "Brian" hash("Brian") = 6
- "Lyn" hash("Lyn") = 4
- "Randy"
- "Sohie"
Collisions

• Separate chaining (linked lists):

- Load factor?

1. Randy
2. Ellen
3. Lyn
4. Stella
5. Brian
6. Randy
7. Sohie

- Load factor?

1. Randy
2. Ellen
3. Lyn
4. Stella
5. Brian
6. Randy
7. Sohie
Separate Chaining -- an example

<table>
<thead>
<tr>
<th>H</th>
<th>A</th>
<th>S</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>G</th>
<th>I</th>
<th>S</th>
<th>F</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>19</td>
<td>8</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>9</td>
<td>19</td>
<td>6</td>
<td>21</td>
<td>14</td>
</tr>
</tbody>
</table>

Load factor?

A Question of Efficiency

• In terms of $n$ and $m$, what is the average size of a linked list?

• How long do searches take?

• If $n \ll m$, accesses are (often) made in a single step, but there is a great deal of wasted space.
Open Addressing

• Open addressing (if \( m \) is much larger than \( n \)):

<table>
<thead>
<tr>
<th>R</th>
<th>M</th>
<th>B</th>
<th>...</th>
<th>Y</th>
<th>R</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>997</td>
<td>998</td>
</tr>
</tbody>
</table>

• How are collisions handled in this case?

Linear Probing

• When the index hashed to is occupied by a stranger, we probe the next position.

• If that position is empty, we insert the entry, otherwise, we probe the next position and repeat.

<table>
<thead>
<tr>
<th>H</th>
<th>A</th>
<th>S</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>G</th>
<th>I</th>
<th>S</th>
<th>F</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>19</td>
<td>8</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>9</td>
<td>19</td>
<td>6</td>
<td>21</td>
<td>14</td>
</tr>
</tbody>
</table>

1 2 3 4 5 6 7 8 9 10 11
A H
There is a problem though: Clustering

- As the table begins to fill up, more and more entries must be examined before the desired entry is found.

- Insertion of one entry may greatly increase the search time for others. For example, consider H, S, H, I in the example below:

<table>
<thead>
<tr>
<th>H</th>
<th>A</th>
<th>S</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>G</th>
<th>I</th>
<th>S</th>
<th>F</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>19</td>
<td>8</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>9</td>
<td>19</td>
<td>6</td>
<td>21</td>
<td>14</td>
</tr>
</tbody>
</table>

  - This problem is known as clustering.

Double Hashing

- When a collision occurs, we use a second hash function to generate a fixed increment for the probe.

- A common function is $hash_2(k) = (m - 2 - k) \mod (m - 2)$.

<table>
<thead>
<tr>
<th>H</th>
<th>A</th>
<th>S</th>
<th>H</th>
<th>I</th>
<th>N</th>
<th>G</th>
<th>I</th>
<th>S</th>
<th>F</th>
<th>U</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
<td>19</td>
<td>8</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>9</td>
<td>19</td>
<td>6</td>
<td>21</td>
<td>14</td>
</tr>
</tbody>
</table>

  - T - 19

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  - T - 20
## Runtime?

<table>
<thead>
<tr>
<th>Operation</th>
<th>Worst Case</th>
<th>Average Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Successor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predecessor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>