Data Structures and Algorithms
Hashing 3

CS 225
G Carl Evans

April 17, 2024
A Hash Table consists of three things:
1. A hash function
2. A data storage structure
3. A method of addressing *hash collisions*

**Client Code:**

```csharp
Dictionary<KeyType, ValueType> d;
d[k] = v;
```
Resizing a hash table

How do we resize?

\[ h(k, i) = \]
## Running Times

<table>
<thead>
<tr>
<th></th>
<th>Hash Table</th>
<th>AVL</th>
<th>Linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Space</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hash Function

Characteristics of a good hash function:

1. Computation Time:

2. Deterministic:

3. ...
Simple Uniform Hashing Assumption

Given table of size $m$, a simple uniform hash, $h$, implies

$\forall k_1, k_2 \in U$ where $k_1 \neq k_2$, $Pr(h[k_1] = h[k_2]) = \frac{1}{m}$

Uniform:

Independent:
Separate Chaining Under SUHA

Given table of size $m$ and $n$ inserted objects

Claim: Under SUHA, expected length of chain is $\frac{n}{m}$
Running Times  *(Don’t memorize these equations, no need.)*
*(Expectation under SUHA)*

**Open Hashing:**

insert: ___________.

find/ remove: ___________.

**Closed Hashing:**

insert: ___________.

find/ remove: ___________.

(Don’t memorize these equations, no need.)

(Expectation under SUHA)
Running Times  (Don’t memorize these equations, no need.)

The expected number of probes for find(key) under SUHA

**Linear Probing:**
- Successful: $\frac{1}{2}(1 + 1/(1-\alpha))$
- Unsuccessful: $\frac{1}{2}(1 + 1/(1-\alpha))^2$

**Double Hashing:**
- Successful: $1/\alpha \times \ln(1/(1-\alpha))$
- Unsuccessful: $1/(1-\alpha)$

**Separate Chaining:**
- Successful: $1 + \alpha/2$
- Unsuccessful: $1 + \alpha$

Instead, observe:

- As $\alpha$ increases:
- If $\alpha$ is constant:
Running Times

The expected number of probes for find(key) under SUHA

**Linear Probing:**
- Successful: \( \frac{1}{2} (1 + 1/(1-\alpha)) \)
- Unsuccessful: \( \frac{1}{2} (1 + 1/(1-\alpha))^2 \)

**Double Hashing:**
- Successful: \( \frac{1}{\alpha} \cdot \ln(1/(1-\alpha)) \)
- Unsuccessful: \( \frac{1}{1-\alpha} \)

**When do we resize?**
Which collision resolution strategy is better?
- Big Records:
- Structure Speed:

What structure do hash tables implement?

What constraint exists on hashing that doesn’t exist with BSTs?

Why talk about BSTs at all?
Running Times

<table>
<thead>
<tr>
<th></th>
<th>Hash Table</th>
<th>AVL</th>
<th>Linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Find</strong></td>
<td><strong>Expectation</strong>*:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Worst Case:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td><strong>Expectation</strong>*:</td>
<td>Worst Case:</td>
<td></td>
</tr>
<tr>
<td><strong>Storage Space</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
std data structures

std::map
::operator[]
::insert
::erase

::lower_bound(key) → Iterator to first element ≤ key
::upper_bound(key) → Iterator to first element > key
std data structures

std::unordered_map
::operator[]
::insert
::erase

::lower_bound(key) ➔ Iterator to first element ≤ key
::upper_bound(key) ➔ Iterator to first element > key

::load_factor()
::max_load_factor(ml) ➔ Sets the max load factor
Hashing in the real world

Even under SUHA, our estimates are *in expectation*.
Hash Table

Worst-Case behavior is bad — but what about randomness?

1) **Fix** $h$, our hash, and assume it is good for *all keys*:

2) Create a *universal hash function family*:
Hash Function (Division Method or Identity Hash)

Hash of form: $h(k) = k \% m$
Hash Function (Mid-Square Method)

Hash of form: $h(k) = (k \times k)$ and take $b$ middle bits where $m = 2^b$
Hash Function (Multiplication Method)

Hash of form: \( h(k) = \lfloor m(\text{remain}(kA)) \rfloor, \ 0 \leq A \leq 1 \)
Hash Function (Universal Hash Family)

Pick a random $h \in H$ s.t. $\forall k_1, k_2 \in U$, $Pr(h[k_1] = h[k_2]) \leq \frac{1}{m}$
Hash Function (Universal Hash Family)

Hash of form: \( h_{ab}(k) = ((ak + b) \mod p) \mod m, \ a, b \in \mathbb{Z}_p^*, \mathbb{Z}_p \)

\( \forall k_1 \neq k_2, \ Pr_{a,b}(h_{ab}[k_1] = h_{ab}[k_2]) \leq \frac{1}{m} \)