Hashing Analysis

Some slides and materials by Uri Zwick (Tel Aviv University)
Hashing with open addressing

“Uniform probing”

Hash table of size $m$

Assume that $h : U \times [m] \rightarrow [m]$

Insert key $k$ in the first free position among $h(k,0), h(k,1), h(k,2), \ldots, h(k,m-1)$

(Sometimes) assumed to be a permutation

Table is not full $\Rightarrow$ Insertion succeeds

To search, follow the same order
Linear probing

“The most important hashing technique”

\[ h(k, i) = (h(k) + i) \mod m \]

More *probes* than uniform probing due to *clustering*: long runs tend to get longer and merge with other runs

But, many fewer *cache misses*

*Extremely efficient in practice*

How do we analyze it?

Which hash functions should we use?
Order of insertions

Theorem: The set of occupied cell and the total number of probes done while inserting a set of items into a hash table using linear probing does not depend on the order in which the items are inserted.

On-Your-Own Exercise: Prove the theorem.

Is the same true for uniform probing?
Probabilistic analysis of uniform probing

[Petersen (1957)]

\( n \) – number of elements in table
\( m \) – size of hash table
\( \alpha = n/m \) – load factor (Note: \( \alpha \leq 1 \))

**Uniform probing:** for every \( k \in U \),
\( h(k, 0), \ldots, h(k, m - 1) \) is random permutation, independent of all other permutations

Expected no. of probes in an **unsuccessful** search of a *random* item is at most
\[
\frac{1}{1 - \alpha}
\]

Expected no. of probes in a **successful** search is at most
\[
\frac{1}{\alpha} \ln \frac{1}{1 - \alpha}
\]
Claim: Expected no. of probes in an unsuccessful search is at most: \[ \frac{1}{1-\alpha} \]

The probability that a random cell is occupied is \( \alpha \)

The probability that the first \( i \) cells probed are all occupied is at most \( \alpha^i \)

\[ 1 + \alpha + \alpha^2 + \ldots = \frac{1}{1-\alpha} \]

Exercise: Do the calculation more carefully and show that the expected no. of probes in an unsuccessful search is exactly \( (m + 1)/(m - n + 1) \)
Probabilistic analysis of linear probing

[Knuth (1962)]

\( \alpha = n/m \) – load factor \((\alpha \leq 1)\)

Random hash function:
for every \( k \in U, h(k) \) is uniformly distributed,
independent of all other \( h(k'), \) for \( k \neq k' \)

Expected no. of probes in an unsuccessful search is at most
\[ \frac{1}{2} \left( 1 + \left( \frac{1}{1 - \alpha} \right)^2 \right) \]

Expected no. of probes in a successful search of a random item is at most
\[ \frac{1}{2} \left( 1 + \frac{1}{1 - \alpha} \right) \]
### Expected number of probes

**Assuming random hash functions**

<table>
<thead>
<tr>
<th></th>
<th>Unsuccessful Search</th>
<th>Successful Search</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniform Probing</strong></td>
<td>$\frac{1}{1-\alpha}$</td>
<td>$\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$</td>
</tr>
<tr>
<td><strong>Linear Probing</strong></td>
<td>$\frac{1}{2} \left( 1 + \left( \frac{1}{1-\alpha} \right)^2 \right)$</td>
<td>$\frac{1}{2} \left( 1 + \frac{1}{1-\alpha} \right)$</td>
</tr>
</tbody>
</table>

When, say, $\alpha \leq 0.6$, all small constants.
Expected number of probes

\[ \frac{1}{2} \left( 1 + \left( \frac{1}{1 - \alpha} \right)^2 \right) \]

\[ \frac{1}{2} \left( 1 + \frac{1}{1 - \alpha} \right) \]

\[ \frac{1}{1 - \alpha} \]

\[ \frac{1}{\alpha} \ln \frac{1}{1 - \alpha} \]
Double Hashing

• Let $f(i)\ use\ another\ hash\ function$

\[ f(i) = i \times h_2(k) \]

Then $h(k, i) = (h'(k) + i \times h_2(k)) \mod m$

And probes are performed at distances of $h_2(k), 2 \times h_2(k), 3 \times h_2(k), 4 \times h_2(k)$, etc

• Choosing $h_2(k)$
  – Don’t allow $h_2(k) = 0$ for any $k$.
  – A good choice:
    $h_2(k) = R - (k \mod R)$ with $R$ a prime smaller than $m$

• Characteristics
  – No clustering problem
  – Requires a second hash function
## Expected number of probes

Assuming *random* hash functions

<table>
<thead>
<tr>
<th></th>
<th>Unsuccessful Search</th>
<th>Successful Search</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniform Probing</strong></td>
<td>( \frac{1}{1-\alpha} )</td>
<td>( \frac{1}{\alpha} \ln \frac{1}{1-\alpha} )</td>
</tr>
<tr>
<td>&amp; <strong>Double Hashing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(where clustering doesn’t occur)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Linear Probing</strong></td>
<td>( \frac{1}{2} \left( 1 + \left( \frac{1}{1-\alpha} \right)^2 \right) )</td>
<td>( \frac{1}{2} \left( 1 + \frac{1}{1-\alpha} \right) )</td>
</tr>
</tbody>
</table>

When, say, \( \alpha \leq 0.6 \), all small constants
Revisiting Fibonacci Hashing

(For an intuitive more-detailed explanation of why it works see https://probablydance.com/2018/06/16/fibonacci-hashing-the-optimization-that-the-world-forgot-or-a-better-alternative-to-integer-modulo/)
Multiplication Method

- The hash function:
  \[ h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor \]
  where A is some real positive constant.

- A very good choice of A is the inverse of the “golden ratio.”

- Given two positive numbers x and y, the ratio x/y is the “golden ratio” if \( \phi = x/y = (x+y)/x \)

- The golden ratio:
  \[ x^2 - xy - y^2 = 0 \quad \Rightarrow \quad \phi^2 - \phi - 1 = 0 \]
  \[ \phi = (1 + \sqrt{5})/2 = 1.618033989... \]
  \[ \sim= \text{Fib}_i/\text{Fib}_{i-1} \]
Fibonacci Hashing

h(k) vs. k
Phyllotactic Ratios: Subdividing the Circle

$360^\circ / \phi \approx 222.5^\circ$, or, equivalently, $360^\circ - 360^\circ / \phi \approx 137.5^\circ$, the most common leaf angle observed
Multiplication Method (cont.)

- Because of the relationship of the golden ratio to Fibonacci numbers, this particular value of A in the multiplication method is called “Fibonacci hashing.”

- Some values of

\[ h(k) = \lfloor m(k \phi^{-1} - \lfloor k \phi^{-1} \rfloor) \rfloor \]

- \[= 0 \quad \text{for } k = 0 \]
- \[= 0.618m \text{ for } k = 1 \quad (\phi^{-1} = 1/1.618... = 0.618...) \]
- \[= 0.236m \text{ for } k = 2 \]
- \[= 0.854m \text{ for } k = 3 \]
- \[= 0.472m \text{ for } k = 4 \]
- \[= 0.090m \text{ for } k = 5 \]
- \[= 0.708m \text{ for } k = 6 \]
- \[= 0.326m \text{ for } k = 7 \]
- \[= ... \]
- \[= 0.777m \text{ for } k = 32 \]