Lecture 5: Hashing

David Woodruff
Hashing

• Universal hashing

• Perfect hashing
Maintaining a Dictionary

• Let U be a universe of “keys”
  • U could be all strings of ASCII characters of length at most 80

• Let S be a subset of U, which is a small “dictionary”
  • S could be all English words

• Support operations to maintain the dictionary
  • Insert(x): add the key x to S
  • Query(x): is the key x in S?
  • Delete(x): remove the key x from S
Dictionary Models

• **Static**: don’t support insert and delete operations, just optimize for fast query operations
  - For example, the English dictionary does not change much
  - Could use a sorted array with binary search

• **Insertion-only**: just support insert and query operations

• **Dynamic**: support insert, delete, and query operations
  - Could use a balanced search tree (AVL trees) to get $O(\log |S|)$ time per operation

• Hashing is an alternative approach, often the fastest and most convenient
Formal Hashing Setup

• Universe U is very large
  • E.g., set of ASCII strings of length 80 is $128^{80}$

• Care about a small subset $S \subseteq U$. Let $N = |S|$.
  • $S$ could be the names of all students in this class

• Our data structure is an array $A$ of size $M$ and a “hash function” $h : U \rightarrow \{0, 1, ..., M-1\}$.
  • Typically $M \ll U$, so can’t just store each key $x$ in $A[x]$
  • $\text{Insert}(x)$ will try to place key $x$ in $A[h(x)]$

• But what if $h(x) = h(y)$ for $x \neq y$? We let each entry of $A$ be a linked list.
  • To insert an element $x$ into $A[h(x)]$, insert it at the top of the list
  • Hope linked lists are small
How to Choose the Hash Function h?

• Want it to be unlikely that \( h(x) = h(y) \) for different keys \( x \) and \( y \)
• Want our array size \( M \) to be \( O(N) \), where \( N \) is number of keys
• Want to quickly compute \( h(x) \) given \( x \)
  • We will treat this computation as \( O(1) \) time

• How long do Query(\( x \)) and Delete(\( x \)) take?
  • \( O(\text{length of list } A[h(x)]) \) time

• How long does Insert(\( x \)) take?
  • \( O(1) \) time no matter what

• How long can the lists \( A[h(x)] \) be?
Bad Sets Exist for any Hash Function

• **Claim:** For any hash function $h: U \rightarrow \{0, 1, 2, ..., M-1\}$, if $|U| \geq (N - 1)M + 1$, there is a set $S$ of $N$ elements of $U$ that all hash to the same location.

• **Proof:** If every location had at most $N-1$ elements of $U$ hashing to it, we would have $|U| \leq (N - 1)M$.

• There’s no good hash function $h$ that works for every $S$. Thoughts?

• **Universal Hashing:** *Randomly choose $h$!*
  - Show for *any* sequence of insert, query, and delete operations, the expected number of operations, over a random $h$, is small.
Universal Hashing

- **Definition:** A set $H$ of hash functions $h$, where each $h$ in $H$ maps $U \rightarrow \{0, 1, 2, ..., M-1\}$ is universal if for all $x \neq y$,

\[
\Pr_{h \leftarrow H} [h(x) = h(y)] \leq \frac{1}{M}
\]

- The condition holds for every $x \neq y$, and the randomness is only over the choice of $h$ from $H$

- Equivalently, for every $x \neq y$, we have:

\[
\frac{|\{h \in H | h(x) = h(y)\}|}{|H|} \leq \frac{1}{M}
\]
**Universal Hashing Examples**

**Example 1:** The following three hash families with hash functions mapping the set \( \{a, b\} \) to \( \{0, 1\} \) are universal, because at most \( 1/M \) of the hash functions in them cause \( a \) and \( b \) to collide, were \( M = |\{0, 1\}| \).

<table>
<thead>
<tr>
<th></th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( h_2 )</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( h_2 )</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( h_2 )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( h_3 )</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Examples that are Not Universal

- Note that a and b collide with probability more than $1/M = 1/2$
Universal Hashing Example

- The following hash function is universal with $M = |\{0,1,2\}|$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$h_2$</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$h_3$</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Using Universal Hashing

• **Theorem:** If $H$ is universal, then for any set $S \subseteq U$ with $|S| = N$, for any $x \in S$, if we choose $h$ at random from $H$, the **expected** number of collisions between $x$ and other elements in $S$ is less than $N/M$.

• **Proof:** For $y \in S$ with $y \neq x$, let $C_{xy} = 1$ if $h(x) = h(y)$, otherwise $C_{xy} = 0$.

Let $C_x = \sum_{y \neq x} C_{xy}$ be the total number of collisions with $x$.

$E[C_{xy}] = \Pr[h(x) = h(y)] \leq \frac{1}{M}$

By linearity of expectation, $E[C_x] = \sum_{y \neq x} E[C_{xy}] \leq \frac{N-1}{M}$
Using Universal Hashing

• **Corollary:** If $H$ is universal, for any sequence of $L$ insert, query, and delete operations in which there are at most $M$ keys in the data structure at any time, the expected cost of the $L$ operations for a random $h \in H$ is $O(L)$
  • Assumes the time to compute $h$ is $O(1)$

• **Proof:** For any operation in the sequence, its expected cost is $O(1)$ by the last theorem, so the expected total cost is $O(L)$ by linearity of expectation
But how to Construct a Universal Hash Family?

• Suppose $|U| = 2^u$ and $M = 2^m$
• Let $A$ be a random $m \times u$ binary matrix, and $h(x) = Ax \mod 2$

• Claim: for $x \neq y$, $\Pr[h(x) = h(y)] = \frac{1}{M} = \frac{1}{2^m}$
But how to Construct a Universal Hash Family?

• **Claim:** For \( x \neq y \), \( \Pr_{h}[h(x) = h(y)] = \frac{1}{M} = \frac{1}{2^m} \)

• **Proof:** \( A \cdot x \mod 2 = \sum_i A_i x_i \mod 2 \), where \( A_i \) is the i-th column of \( A \)
  
  If \( h(x) = h(y) \), then \( Ax = Ay \mod 2 \), so \( A(x-y) = 0 \mod 2 \)
  
  If \( x \neq y \), there exists an \( i^* \) for which \( x_{i^*} \neq y_{i^*} \)
  
  Fix \( A_j \) for all \( j \neq i^* \), which fixes \( b = \sum_{j\neq i^*} A_j (x_j - y_j) \mod 2 \)
  
  \( A(x-y) = 0 \mod 2 \) if and only if \( A_{i^*} = b \)

  \[
  \Pr_{A_{i^*}}[A_{i^*} = b] = \frac{1}{2^m} = \frac{1}{M}
  \]

  So \( h(x) = Ax \mod 2 \) is universal