More on Hashing

CSCI2100 Tutorial 7
Shangqi Lu
Review on Hash Table

• Given a set of $n$ integers $S$ in $[1, U]$
• Main idea: divide $S$ into a number $m$ of disjoint subsets
• Guarantees
  • Space consumption: $O(n + m)$
  • Preprocessing cost: $O(n + m)$
  • Query cost: $O(1 + n/m)$ in expectation
Review on Hash Table

• Given a set of $n$ integers $S$ in $[1, U]$
• Main idea: divide $S$ into a number $m$ of disjoint subsets
  • Set $m = \Theta(n)$
• Guarantees
  • Space consumption: $O(n)$
  • Preprocessing cost: $O(n)$
  • Query cost: $O(1)$ in expectation
Review on Hash Table

• Divide $S$ into a number $m$ of disjoint subsets:
  • Choose a function $h$ from $[1, U]$ to $[1, m]$
  • For each $i \in [1, m]$, create an empty linked list $L_i$
  • For each $x \in S$:
    • Compute $h(x)$
    • Insert $x$ into $L_{h(x)}$

• Important:
  • Choose a good hash function $h$
Review on Hash Table

• Construct a **universal family**
  • Pick a prime number $p$ such that $p \geq m$ and $p \geq U$
  • Choose an integer $\alpha$ from $[1, p - 1]$ uniformly at random
  • Choose an integer $\beta$ from $[0, p - 1]$ uniformly at random
  • Define a hash function:
    $$h(k) = 1 + ((\alpha k + \beta) \mod p) \mod m$$
Example

• Let $S = \{19, 36, 63, 53, 14, 9, 70, 26\}$
• We choose $m = 10, p = 71$, suppose that $\alpha$ and $\beta$ are randomly chosen to be 3 and 7, respectively
• $h(k) = 1 + ((3k + 7) \mod 71) \mod 10$
Hash Table

- Let $H$ be the universal family defined in the previous slides
- Given a function $h \in H$ and an integer $q \in [1, U]$:
  - Define $\text{cost}(h, q) = |\{x \in S \mid h(x) = h(q)\}|$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>$U$</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>$\text{cost}(h_1, 1)$</td>
<td>$\text{cost}(h_1, 2)$</td>
<td>...</td>
<td>$\text{cost}(h_1, U)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>$h_2$</td>
<td>$\text{cost}(h_2, 1)$</td>
<td>$\text{cost}(h_2, 2)$</td>
<td>...</td>
<td>$\text{cost}(h_2, U)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>$h_{</td>
<td>H</td>
<td>}$</td>
<td>$\text{cost}(h_{</td>
<td>H</td>
<td>}, 1)$</td>
</tr>
<tr>
<td>Average</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td></td>
</tr>
</tbody>
</table>
Hash Table

- **Worst-case expected query cost:** $O(1)$
  - Pick a hash function from a universal family
- **Worst-case query cost:** $O(n)$
  - All elements are hashed into the same value

**Question:**
- Can we improve the worst-case query cost?
Hash Table

• Replace linked lists with arrays
• Sort the arrays, cost $O(n \log n)$ for preprocessing
Hash Table

• Query: whether 29 exists
• Step 1:
  • Access the hash table to obtain the address of corresponding array
  • $O(1)$ time
Hash Table

• Query: whether 29 exists
• Step 2:
  • Perform binary search on the array to find the target
    • $O(\log n)$ time
• Overall worst-case complexity: $O(\log n)$
Hash Table

• This method retains the $O(1)$ worst-case expected query time.

• Proof:
  • Suppose we look up an integer $q$
  • Define random variable $X_{h(q)}$ to be the length of array that corresponds to the hash value $h(q)$
  • Expected query time:
    • $E[\log_2 X_{h(q)}] = \sum_{l=1}^{n} \log_2 l \Pr(X_{h(q)} = l)$
    • $\leq \sum_{l=1}^{n} l \Pr(X_{h(q)} = l)$
    • $= E[X_{h(q)}]$ 
    • $= O(1)$
The Two-Sum Problem (revisited)

• Problem Input:
  • A set $S$ of \textit{unsorted} $n$ distinct integers
  • The value $n$ has been placed in Register 1
  • A positive integer $\nu$ has been placed in Register 2

• Goal:
  • Determine whether if there exist two different integers $x$ and $y$ in $S$ such that $x + y = \nu$

• For example:
  • Find a pair whose sum is 20

\begin{array}{cccccc}
11 & 3 & 17 & 7 & 2 & 13 \\
\end{array}
Solution 1: Binary Search the Answer

- Goal: Find a pair \((x, y)\) such that \(x + y = v\)
- Observe that given \(x\), \(y = v - x\), is determined

- Solution:
  - Sort \(S\)
  - For each \(x\) in \(S\):
    - set \(y\) as \(v - x\)
    - Use binary search to see if \(y\) exists in the sequence

- Time complexity: \(O(n \log n)\)
Solution 2: Using the Hash Table

• Step 1 and 2:
  • Choose a hash function $h$ and create an empty hash table $H$
  • Insert each $x$ in $S$ into $L_h(x)$

• Step 3:
  • For each $x$ in $S$:
    • Set $y$ as $v - x$
    • Check if $y$ is in the hash table; if it is, return yes
  • Return no
Time Complexity

• Step 1 and 2: $O(n)$

• Step 3:
  • Let $X_i$ be the query time for the $i$-th integer in $S$
  • We know $E[X_i] = O(1)$
  • Define $X = \sum_i X_i$
  • The worst-case expected cost of step 3:
    • $E[X] = \sum_i E[X_i] = O(n)$

• Overall: $O(n)$ in expectation