Dictionary: search, insert, delete

Balanced BST an efficient dictionary: \( O(\log n) \) for each operation

Set of elements \( S \), Dictionary \( (S) \)

\[ |S| = n \quad S \text{ a totally ordered} \]

Question: Can we do better?

Define a universal set \( U \) of keys

\[ U = \{0, 1, 2, \ldots, n-1\} \]

Hash function \( h: U \rightarrow T: \{0, 1, \ldots, m-1\} \)

\[ |U| \gg |T| \]

Any instance of building a dictionary is

Given a subset \( S \subseteq U \),

\[ h: S \rightarrow T \quad |S| \leq |T| \]

\[ |S| = n \quad \text{otherwise collision} \]
For any \( x, y \in U \), \( x \neq y \) calls if:
\[
h(x) = h(y)
\]

A hash function is "good" if there are no collisions within \( S \).

A good hashing scheme should work for all subsets \( S \):
\[
h(x) = x \mod m
\]

What happens for a random subset \( S \):

- Each of the \( n \) elements of \( S \) is chosen uniformly at random from \( U \).
- (Strictly speaking, \( \binom{N}{n} \) of them is chosen at random.)
What is performance of the hash function mod m? 

What is the expected # of elements that will fall into location 0?

Recall that when elements collide in the hash table, build a linked list.

By using the linked list, the performance of hashing is proportional to the length of the list.

Let \( Z_i \) be a r.v.: 
\[
Z_i = \begin{cases} 
1 & \text{if } x_i \in T(0) \\
0 & \text{otherwise}
\end{cases}
\]

\[
\text{Prob } [Z_i = 1] = \frac{1}{m}
\]

# elements in \( T(0) \) = \( Z_1 + Z_2 + \cdots + Z_n = Z \)

\[
E[Z] = E\left[ \sum_i Z_i \right] \leq \sum_i E[Z_i] = n \cdot \frac{1}{m}
\]
\[
E[Z_i] = \frac{1}{m} \quad (\text{Read Ch 2, note 0.1})
\]

If \( n = m \), \( E[Z] = 1 \)

\[
\Rightarrow \quad \text{Prob} \left( Z \geq 2 \right) \leq \frac{1}{2} \quad \text{from Markov}
\]

We would like to analyze the performance over a sequence of operations

\[
O_1, O_2, O_3, \ldots, O_t \quad O_i \in \{ \text{search, insert, delete} \}
\]

\[
E[O_i] = \frac{n}{m}
\]

\[
E[O_1 + O_2 + O_3 + \ldots + O_t] = \sum E[O_i] = t \cdot \frac{n}{m}
\]

What if \( S \) is worst-case arbitrary?

Assume that we have a family of hash functions \( H \)

**Basic Hope:** Given an arbitrary \( S \) there is at least one good \( h \in H \) for \( S \)
Idea: Pick a random $h \in H$

A family of hash functions $H$ is called \textit{universal} if
\[
\forall x, y \in U \quad \sum_{h \in H} \delta_h(x, y) \leq c \frac{|H|}{m}
\]
where $c$ is some constant
\[
\delta_h(x, y) = \begin{cases} 1 & \text{if } h(x) = h(y) \\ 0 & \text{otherwise} \end{cases}
\]
collision function for $h$

Therefore, if we pick a hash function at random, the probability that any given pair $x, y$ will collide is $\frac{c}{m}$

What is the expected number of collisions in location 0 of the table for a random hash function?

(For any arbitrary $S$)

Consider a generic element $x \in S$
\[
\frac{1}{|H|} \sum_{y \in H \atop y \neq x} \sum_{h \in H} \delta_h(z, y) = \text{expected collision for a fixed element } x \in S
\]

\[
\sum_{y \in S} \frac{1}{|H|} \sum_{h \in H} \delta_h(x, y)
\]

\[
\sum_{y \in S} \frac{1}{m} \leq \frac{C \cdot n}{m}
\]

Existence of Universal family?