7.1 Skip Lists (cont.)

Expected Search time = \( E \left[ \sum_{j=r}^{1} 1 + C(I_j(y)) \right] \), where \( C(I_j(y)) \) is the number of children of that node.

Example:
Interval \( I \xrightarrow{\text{siblings}} [x_1, x_2] [x_2, x_3] \ldots [x_S, x_{S+1}] \)

Expected number of children for any node in the tree = \( O(1) \).

Expected Search time \( \leq \left( 1 - n^{-\alpha} \right) C_1 \log n + n^{-\alpha} C_2 n = O(\log n) \)

Insert(\( x \)):
- Create a new node & put \( x \) in it.
- Create a level for this node randomly.
- Search from root to leaf.
- Split & insert.

Expected time for insertion & deletion is \( O(\log n) \).

7.2 Hashing:

Hashing can be used to implement a dictionary in a randomized sense. Assume that the dictionary keys are integers from \( M = \{0, 1 \ldots m-1\} \). If we have \( m \) space, each dictionary operation can be done in \( O(1) \) time. If we only have \( n \) space, we can use a function \( h: M \rightarrow N = \{0, 1 \ldots n-1\} \)

Search (\( x \)):
Compute \( h(x) \) and look for \( x \) in a list \( a[h(x)] \), where \( a \) is an array of \( n \) lists.

**Definition 7.2.1**: if \( x \neq y \) and \( h(x) = h(y) \) then we say there is a collision.
**Definition 7.2.2:** \( h \) is perfect for \( S \subseteq M \) if \( h(x) \neq h(y) \) for any two \( x, y \in S \) where \( x \neq y \).

There are \( n^m \) functions from \( M \to N \).

**Fact 7.2.1:** For a given \( S \), we can always find a hash function that is perfect for \( S \).

**Fact 7.2.2:** There is no hash function that is perfect for every \( S \subseteq M \), if \( m > n \).

If one assumes that the input keys are uniformly distributed in \( M \), and if there are \( n' \) keys, then the expected size of any list in \( a[] \) is \( O \left( \frac{n'}{n} \right) \). This is the expected time for any dictionary operation.

**Idea:**
1. Settle for near-perfect hash functions
2. Use a collection \( H \) of hash functions for any \( S \subseteq M \), a good fraction of \( H \) will be near-perfect for \( S \).
3. For any given \( S \), we pick a random \( h \in H \) and use it for \( S \).

**Definition 7.2.3:** A class \( H \) of hash function is 2-universal if for a random \( h \in H \)

\[
\Pr[h(x) = h(y) \mid x \neq y] \leq \frac{1}{n} \text{ for any two } x, y \in M.
\]

The class of all functions from \( M \) to \( N \) is 2-universal \(|H| = n^m\). We need \( m \log n \) bits to specify a member of \( H \).

**Definition 7.2.4:**
\[
\delta(x, y, h) = \begin{cases} 
1, & \text{if } h(x) = h(y) \text{ and } x \neq y \\
0, & \text{otherwise}
\end{cases}
\]

\[
\delta(x, y, h) = \sum_{h \in H} \delta(x, y, h) \quad \delta(x, y, H) = \sum_{h \in H} \delta(x, y, h)
\]

\[
\delta(x, y, h) = \sum_{x \in X} \delta(x, y, h) \quad \delta(X, y, H) = \sum_{x \in X} \delta(x, y, H)
\]
\[ \delta(x, Y, h) = \sum_{y \in Y} \delta(x, y, h) \quad \text{and} \quad \delta(X, Y, H) = \sum_{x \in X} \delta(x, Y, H) \]

**Lemma 7.2.1:** Let \( H \) be any class of hash functions then \( \delta(x, y, H) > \frac{|H|}{n} - \frac{|H|}{m} \).

\( \delta(x, y, H) \) is the number of hash functions that will force \( x \) & \( y \) to collide.

**Proof:**

For any \( h \in H \), let \( A_z = \{ x \in M : h(x) = z \} \) where \( z = \{ 0, 1, ..., n-1 \} \)

\[ \delta(A_w, A_z, h) = 0 \quad \text{if} \quad w \neq z \]

\[ \delta(A_w, A_z, h) = |A_z| \cdot (|A_z| - 1) \quad \text{if} \quad w = z \quad (\text{Elements of the same part}) \]

\[ \delta(M, M, h) = \sum_{z=0}^{n-1} \delta(A_z, A_z, h) \]

\[ \Rightarrow \delta(M, M, h) \leq n \cdot \frac{m}{n} \left( \frac{m}{n} - 1 \right) = m \left( \frac{1}{n} - \frac{1}{m} \right) \]

\[ \exists x, y \text{ s.t. } \delta(x, y, h) \geq \frac{m \left( \frac{1}{n} - \frac{1}{m} \right)}{m^2} = \frac{1}{n} - \frac{1}{m} \quad \text{(a random } h) \]

\[ \Rightarrow \delta(x, y, h) \geq \frac{|H|}{n} - \frac{|H|}{m} \quad \text{(all possible } h's) \]

**Lemma 7.2.2:** Suppose \( H \) is 2-universal, \( S \subseteq M \) & \( x \in M \) then \( E[\delta(x, S, h)] = \frac{|S|}{n} \)