1 Balls and Bins (Cont.)

- For $k \ll \sqrt{n}$, $P[\text{bin } i \text{ has } k \text{ balls}] \leq \frac{1}{e k!}$
- $P[\text{bin } i \text{ has } > k \text{ balls}] \leq \frac{C}{e k!}$, $C$ is constant
- $P[\exists \text{ a bin has } \geq k \text{ balls}] \leq \frac{n}{e k!}$

\[ - P[\exists \text{ a bin has } \geq k \text{ balls}] = P[\text{bin } 0 \geq k \text{ balls} \cup \cdots \cup \text{bin } n \geq k \text{ balls}] \]
\[ \leq P[\text{bin } 0 \geq k \text{ balls}] + \cdots + P[\text{bin } n \geq k \text{ balls}] \leq \frac{n}{e k!} \]

**Definition 1.** An event $E(n)$ occurs “with high probability” (w.h.p.) if $\forall \alpha > 0$, \( \exists C, \text{ s.t. } P[E(n)] \geq 1 - \frac{C}{n^\alpha} \).

**Theorem 2.** After inserting $n$ items into a hash table with $n$ buckets, the fullest bucket will contain $O\left(\frac{\ln n}{\ln \ln n}\right)$ elements w.h.p.

Proof:
Suppose $\alpha > 0$,

\[ P[\exists \text{ bin has } > k \text{ balls}] \leq \frac{n}{e k!} \leq \frac{C}{n^\alpha} \]
\[ \ln n - 1 - k \ln k \leq \ln C - \alpha \ln n \]
\[ k \ln k \geq (\alpha + 1) \ln n, \text{ for some constant } C \]

Assume \( k = (\alpha + 1) \frac{\ln n}{\ln \ln n} \), then

\[ k \ln k = (\alpha + 2) \frac{\ln n}{\ln \ln n} (\ln (\alpha + 2) + \ln \ln n - \ln \ln \ln n) \]

Since \( \ln \ln n \) dominates \( \ln (\alpha + 2) \) and \( \ln \ln \ln n \)

\[ k \ln k = (\alpha + 2) \ln n \geq (\alpha + 1) \ln n \]

The conclusion is proved. □

**Theorem 3.** Lookups in a hash table with \( n \) buckets and \( n \) items (and a uniform hash function) are \( O\left(\frac{\ln n}{\ln \ln n}\right) \) w.h.p.

**Proof:**

The worst case is searching the bucket with longest chain. Thus, the conclusion can be proved by Theorem 2. □

## 2 Bloom Filter

**Definition 4.** A Bloom filter is a space-efficient probabilistic data structure used to test whether an element is a member of a set. It has several obvious properties:

- It is a dynamic set.
- Elements can be added to the set.
- Elements can not be removed from the set.
- When a query returns “inside set”, the element may be in the set or not; when a query returns “not in set”, the element is definitely not in the set.
An empty Bloom filter is a bit array vector $A$ of $n$ bits and there are $k$ different hash functions $\{h_i : U \rightarrow \{0, 1, \cdots, n-1\}, i = 1, \cdots, k\}$ each maps element to one of $n$ array positions uniformly.

The following graph implicates how to implement insert and lookup in Bloom filter.

The corresponding algorithm can be summarized as:

\[
\begin{align*}
\text{insert} (x) \\
\text{for } i = 1 \text{ to } k \\
\quad A[h_i(x)] \leftarrow 1 \\
\end{align*}
\]

\[
\begin{align*}
\text{lookup} (x) \\
\text{for } i = 1 \text{ to } k \\
\quad \text{if } A[h_i(x)] \neq 1 \\
\quad \quad \text{return } \text{No} \\
\quad \text{return } \text{Yes} \\
\end{align*}
\]

After inserting $m$ items into $n$ slots with $k$ hash functions, then

\[
P[A[i] = 0] = \left(1 - \frac{1}{n}\right)^{km} \approx e^{-\frac{km}{n}} = e^{-\frac{s}{n}}
\]

suppose $s = \frac{m}{n}$. 

3