1 Skip Lists

GOAL: Implement a simple data structure that supports search, insert, and delete in $O(\log n)$ time each.

1.1 Initial Idea: ”Deterministic” Skip Lists

The data structure contains some number of levels ($L_0, L_1, \ldots, L_N$). Each level is an ordered list of elements. $L_0$ contains all elements. The set of elements in $L_{i+1} \subseteq$ set of elements in $L_i$.

![Figure 1: Example Skip List](image)

We used the deterministic rule ”keep every alternate element” in going from $L_i$ to $L_{i+1}$. Search 16 is represented by the red arrows in the above picture. Elements in the search path are $-\infty, -\infty, 4, 4, 15, 15$. Search returns the largest element smaller than the searched number (15 in this case).

-Deterministic rule is hard to maintain for insert and delete so we will try to use randomization.

1.2 Probabilistic Rule

Each element in $L_i$ is chosen independently with probability $\frac{1}{2}$ to be copied to $L_{i+1}$ Search works as before.

1.2.1 Insert

Insert $k$ works as follows:

- First search for $k$ and find search path $e_t, e_{t-1}, \ldots, e_0$
- Insert $k$ immediately after $e_0$
- We toss a fair coin to determine if $k$ should be inserted into $L_1$
• If we determine k should be inserted, we walk back the search path to the first element $e_i$ in Level $L_1$
• We insert k immediately after $e_i$ in $L_1$
• We continue coin tossing until tails

1.2.2 Delete
Delete k also involves doing search on k first and then removing k from all levels it occurs in

1.2.3 Runtime
What is the running time of search, insert, and delete? Search can be viewed as a sequence of down and right arrow traversals.

Any elements visited when moving right are visited because they are smaller than k. If the element was on previous levels, we would have visited it.
How many down traversals can we make? Let $H_e$ denote the height of element $e$. $H_e$ is a geometrically distributed random variable.

$$P(H_e \geq k) = \frac{1}{2^{k-1}}$$

\[ \therefore \text{Setting } k = 3\log_2 n \text{ gives us } P(H_e \geq 3\log_2 n) = \frac{2}{n^3}. \]
Let $H$ denote $\max_e H_e$.

$$P(H \geq 3 \log_2 n) = P(\exists e : H_e \geq 3 \log_2 n) \tag{1}$$

$$\leq \sum_e e P(H_e \geq 3 \log_2 n) \tag{2}$$

$$= \frac{2}{n^2} \tag{3}$$

(2) is established by union bound. \therefore With probability $\geq 1 - \frac{2}{n^2}$, $H < 3 \log_2 n$.

Let $L$ be the length of the search path. We will upper bound $P(L \geq \log_2 n)$ for appropriately chosen $c$.

$$P(L \geq \log_2 n) = P(L \geq \log_2 n | H \geq \log_2 n) P(H \geq \log_2 n) \tag{4}$$

$$+ P(L \leq \log_2 n | H < \log_2 n) P(H < \log_2 n) \tag{5}$$

$$\leq \frac{2}{n^2} + P(L \geq \log_2 n H < \log_2 n) \tag{6}$$

We want to focus on upper bounding $P(L \geq \log_2 n H < \log_2 n)$.

Now view $L$ as being obtained via a sequence of coin tosses. We are given that the number of heads $< \log_2 n$.

\therefore If $L = \log_2 n$, the number of tails $\geq (c - 3) \log_2 n$. The expected number of tails is $\frac{c}{2} \log n$.

Let $T = \text{number of tails}$. We want to use Chernoff bounds to upper bound the $P(T \geq (c - 3) \log n)$.

$$(1 + \delta)\mu = (1 + \delta)\frac{c}{2} \log n = (c - 3) \log$$

$$\delta = 1 - \frac{6}{c}$$

If we pick $c > 6$, $\delta$ will satisfy $0 < \delta < 1$. Using form (b) of the Chernoff bounds gives us:

$$P(t \geq (c - 3) \log n) \leq e^{-(c - 3) \log n} \tag{7}$$

$$= e^{-\frac{(c - 3)^2}{6c}} \tag{8}$$

We can pick $c$ large enough to make $\frac{(c - 3)^2}{6c}$ as large as we want to ensure $P(T \geq (c - 3) \log n) \leq \frac{1}{n^2}$. Using this, we can plug into equation (6) to get the following:

$$P(L \geq c \log n) \leq \frac{2}{n^2} + \frac{1}{n^2} \tag{9}$$

$$= \frac{3}{n^2} \tag{10}$$

Insert time = time to search + height of element being inserted. \therefore Insert time is $O(\log n)$ with high probability. The time analysis of delete is similar.

## 2 Extensions to Chernoff Bounds

### 2.1 Negative Dependence

Let $X_1, X_2, ..., X_n$ be a random variable such that for any $I, J \subseteq 1, 2, ..., n$ with $I \cap J = \emptyset$ given that $X_i's$ where $i \in I$ take on "high values", the probability that $X_j's$ where $j \in J$ take on "low values" increases.
2.1.1 Example: Balls in Bins

For \( i = 1, 2, \ldots, n \) let
\[
B_i = \begin{cases} 
1 & \text{if bin is empty} \\
0 & \text{otherwise}
\end{cases}
\]

(11)

\( B_1, B_2, \ldots B_n \) exhibit negative dependence.

**Implications of negative dependence**

1. If \( X_1 \) and \( X_2 \) exhibit negative dependence then
\[
E[X_1 X_2] \leq E[X_1]E[X_2]
\]

This makes sense because
\[
P(X_1 = \text{high} \cap X_2 = \text{high}) \leq P(X_1 = \text{high})P(X_2 = \text{high})
\]

2. If \( X_1, X_2, \ldots, X_n \) exhibit negative dependence then
\[
E\left[\prod_{i=1}^{n} f_i(X_i)\right] \leq \prod_{i=1}^{n} E[f_i(X_i)]
\]

for non-decreasing function \( f_i \). A special case of 2. is
\[
E\left[\prod_{i=1}^{n} e^{tx_i}\right] \leq \prod_{i=1}^{n} E[e^{tx_i}]
\]

Mutual independence gave us = in place of the \( \leq \).

Let \( B = \sum_{i=1}^{n} B_i \) so \( B \) denotes the number of empty bins.

\[
E[B] = \sum_{i=1}^{n} E[B_i]
\]

(12)

\[
= \sum_{i=1}^{n} P(B_i = 1)
\]

(13)

\[
= \sum_{i=1}^{n} (1 - \frac{1}{n})^n
\]

(14)

\[
= n \frac{1}{n} e
\]

(15)

So let \( \mu \) denote \( \frac{n}{e} \). Since \( B_i \)'s exhibit negative dependence, we can use Chernoff Bounds on \( B \).

\[
\therefore P(B \geq (1 + \delta) \frac{n}{e}) \leq e^{-\frac{\delta^2 \mu}{3}}
\]

(16)

\[
\leq \frac{1}{\exp(n)}
\]

(17)

for any small constant \( \delta \) where \( 0 < \delta < 1 \).
2.2 K-Wise Independence

Recall that $X_1, X_2, ..., X_n$ are k-wise independent if for any $I \subseteq 1, 2, ..., n$, $|I| \leq k$ and any $i \in I$

\[ P(X_i = x_i | \land_{j \in I} X_j = x_j) = P(X_i = x_i) \]

The setting $k = 2$ is called pair-wise independence. There are Chernoff bounds for k-wise independent random variables (Srinivasan, Schmidt, Siegel).

2.3 Bounded Dependence

Each random variable is independent with respect to all other random variables except for at most $d$. This is usually defined with a dependency graph.

![Figure 3: Example of bounded dependency graph. X1 is independent of X2 and X5. The outdegree of each node $\leq d$](image)

Chernoff Bounds worsen an exponential “$\frac{1}{d}$” factor

2.3.1 Example

\[ P(X \geq (1 + \delta)\mu) \leq (e^{-\delta^2 \mu / 3})^{\frac{1}{d}} \]

2.4 Chernoff-Hoeffding bounds

There are extensions to Chernoff bounds that allow $X'_i$’s to take on values other than 0 and 1. These are called Chernoff-Hoeffding bounds.

3 Probabilistic Method

Refers to technique for proving that an object with certain properties exist. Sometimes probabilistic method proofs can be turned into algorithms. These algorithms may be slightly weaker and not hold all the properties that the proof does.

3.1 Example: MaxCut

**INPUT:** A graph $G = (V, E)$
**OUTPUT:** A partition of $V$ into $(V_1, V_2)$ such that the number of edges between $V_1$ and $V_2$ is maximized.
This is NP-hard. Could we get an efficient approximation algorithm? i.e. could we design an algorithm such that the number of edges crossing the partition it produces \( \geq 90\% \) of the size of the optimal cut.

**Theorem 1.** Let \( G = (V, E) \) be a graph with \( m \) edges. Then there exists a partition \((V_1, V_2)\) of \( V \) such that \( \geq \frac{m}{2} \) edges cross that \((V_1, V_2)\) partition.

**Proof.** Take each vertex \( v \in V \) and place it in \( A \) or in \( B \) independently with probability = \( \frac{1}{2} \). Let the edges be \( e_1, e_2, ..., e_m \). For \( i = 1, 2, ..., m \) let

\[
X_i = \begin{cases} 
1 & \text{if } e_i \text{ crosses the } (A, B) \text{ partition} \\
0 & \text{otherwise} 
\end{cases}
\]

(18)

\[
P(X_i = 1) = \frac{1}{2} \\
E[X_i] = \frac{1}{2} \\
E[X] = \frac{m}{2}
\]

where

\[
X = \sum_{i=1}^{m} X_i
\]

\( X \) = number of edges crossing \((A,B)\) partition. Since \( \frac{m}{2} \) is the average, there must be some cut that is at least \( \frac{m}{2} \). \qed