# Skip Lists:
# A Probabilistic Alternative to Balanced Trees

Presented by

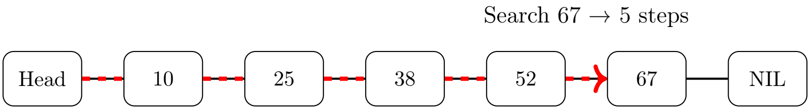
Rishab Biswas - A125017
Siddharth Kumar - A125021

International Institute of Information Technology, Bhubaneswar

December 3, 2025

# Outline

# Revisiting Singly Linked List



# Why Linked Lists Are Not Enough

- Search $= O(n) \rightarrow$ too slow for large data
- No random access
- Cache-unfriendly
- Poor concurrency support

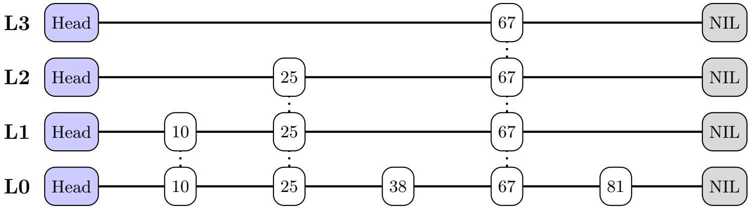
**We need**

Fast search + Simple implementation + Easy updates

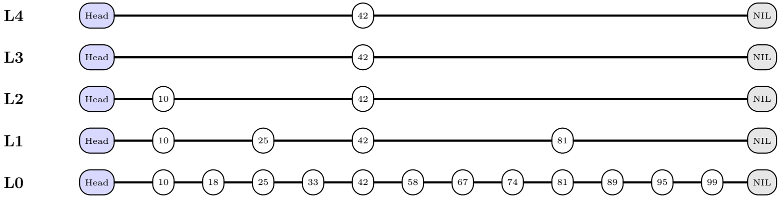
# Skip List: Multi-Level Express Lanes

## Brilliant Idea (William Pugh, 1990)

Randomly promote nodes to higher levels → create express lanes!

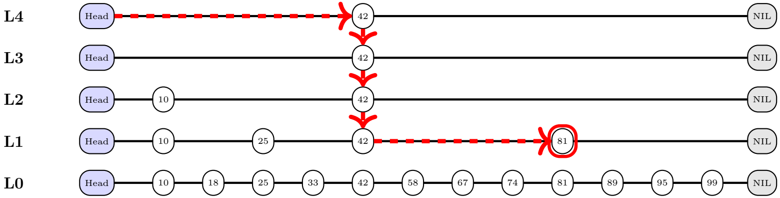


# Realistic Example: Searching in a 12-Node Skip List




**Slide 1:** Full skip-list view before search.

# Realistic Example: Search 81 in a 12-Node Skip List




**Search 81 completed in 5 pointer moves (fast due to higher levels). Linked list needs 9 steps → Skip list wins!**

# Why Skip List Search is $O(\log n)$ (Easy Explanation)

**Key Idea**

Search moves **right** until the next value is too big, then **drops down**. There are $\log n$ levels, and only a constant number of right moves per level.

### 1. Expected Number of Levels

Nodes reach level $i$ with probability $p^i$. Highest level $\approx \log_{1/p} n$. For $p = \frac{1}{2}$: $\log_2 n$ levels.

### 2. Expected Right Moves per Level

- Probability a node appears on level $i$ is $p$.
- Looking backward, number of nodes scanned is geometric$(p)$.

Expected right moves per level: $\frac{1}{p}$.

### 3. Total Expected Search Cost

Total expected steps:

$$\frac{1}{p} \log_{1/p} n + \frac{1}{1-p}$$

For $p = \frac{1}{2}$:

$$2 \log_2 n + 2$$

# Search — Just Two Loops

## SEARCH(key)

1. Start at top-left corner (highest level)
2. While next node exists AND next.key $<$ key $\rightarrow$ move right
3. When stuck $\rightarrow$ drop down one level
4. Repeat until level 0
5. Return node if found

# Insert — Random Height

**INSERT(key, value)**

1. Perform SEARCH and remember one node per level (update[])
2. If key exists → update value
3. Else:
   - Flip coin repeatedly → decide height
   - Create node with that height
   - Link it using update[] pointers

**Randomness does the balancing automatically!**

# Delete — Just Fix Pointers

## DELETE(key)

1. Perform SEARCH and remember update[]
2. If found $\rightarrow$ remove node from all its levels
3. Fix forward pointers (skip over it)

**No rotations • No rebalancing • Super simple!**

# Space Complexity of Skip Lists (p = 1/2)

**Expected number of nodes per level:**
At level $i$, a node is promoted with probability $p^i$.

$$\text{Expected nodes at level } i = n \cdot p^i$$

For $p = \frac{1}{2}$:

$$n\left(\frac{1}{2}\right)^i$$

**Total expected pointers in the entire structure:**

$$\sum_{i=0}^{\infty} n\left(\frac{1}{2}\right)^i = n \cdot \sum_{i=0}^{\infty}\left(\frac{1}{2}\right)^i = n \cdot 2 = 2n$$

## Final Result

$$\boxed{\text{Expected space complexity } = O(2n)}$$

# Tradeoff: Time Complexity vs Space (Role of p)

Skip list performance depends on the promotion probability $p$.

| **Probability $p$** | **Height (Levels)** | **Space Used** |
|---|---|---|
| High ($p \to 1$) | Tall structure | Many nodes per level (high space) |
| Low ($p \to 0$) | Short structure | Few levels (low space) |

**Search complexity:**

$$O\left(\frac{1}{p}\log n\right)$$

**Space complexity:**

$$O\left(\frac{n}{1-p}\right)$$

## Key Tradeoff

- Lower $p \to$ fewer levels $\to$ **less space** but **slower searches**
- Higher $p \to$ more levels $\to$ **faster searches** but **more space**
- $p = \frac{1}{2}$ is the sweet spot $\to$ balanced time and space

# Skip List Family

**Deterministic** $\rightarrow$ no randomness, guaranteed $O(\log n)$

**Concurrent/Lock-Free** $\rightarrow$ Java's `ConcurrentSkipListMap`

**Finger Skip List** $\rightarrow$ cached pointer (used in Redis)

**Different** $p \rightarrow p = 1/e$ optimal, but $p = 0.5$ most common

**Most Popular**

Probabilistic + Concurrent

# Skip List vs Other Structures

| Structure | Search | Insert/Delete | Code Complexity |
|---|---|---|---|
| Linked List | $O(n)$ | $O(1)$–$O(n)$ | Very Simple |
| BST (unbalanced) | $O(n)$ worst | $O(n)$ worst | Simple |
| AVL / Red-Black | $O(\log n)$ | $O(\log n)$ | Complex |
| **Skip List** | $O(\log n)$ **exp.** | $O(\log n)$ **exp.** | **Very Simple** |

# Used in Production Everywhere

- **Redis** → Sorted Sets (ZSET)
- **Java** → `ConcurrentSkipListMap/Set`
- **LevelDB** / **RocksDB** → MemTable
- **Lucene** / **Solr** → Term dictionary
- **MongoDB WiredTiger, Cassandra, Intel TBB**

**Why Industry Loves It Because**

Simple + Fast + Excellent concurrency

# Real-World Applications (Detailed)

## 1. Redis Sorted Sets (ZSET)

Used for:

- Leaderboards
- Time-series indexing
- Score-based range queries

Expected search time: $O(\log n)$

## 2. LevelDB / RocksDB MemTable

Skip lists provide:

- Fast in-memory writes
- Lock-free reads
- Efficient range scans

## 3. Java Concurrency

`ConcurrentSkipListMap/Set` offers:

- Non-blocking reads
- Sorted ordering
- Scalable multi-threaded performance

## 4. Search Engines (Lucene/Solr)

Used for:

- Term dictionary indexing
- Fast prefix/range lookups
- Real-time text search

# Key Takeaways

## Skip Lists = Best of Both Worlds

- Simplicity of Linked Lists
- Performance of Balanced Trees
- No complex rotations or rebalancing
- Expected $O(\log n)$ using only randomness

## William Pugh's Legacy

"Replace complicated deterministic code with simple code + a little randomness"

# References


- Pugh, William.
  Skip Lists: A Probabilistic Alternative to Balanced Trees.
  *Communications of the ACM*, 33(6), 1990.
- Herlihy, Maurice, and Shavit, Nir.
  *The Art of Multiprocessor Programming.*
  Morgan Kaufmann, 2008.
- Redis Developers.
  Sorted Sets (ZSET): Internal Implementation.
  Available at: `https://redis.io/docs`
- Google Developers.
  LevelDB Architecture and MemTable Implementation.
  Available at: `https://github.com/google/leveldb`
- Apache Lucene.
  Term Dictionary and Index Structures.

# Thank You!

Questions?

Rishab Biswas Siddharth Kumar
IIIT Bhubaneswar December 3, 2025