CSE221

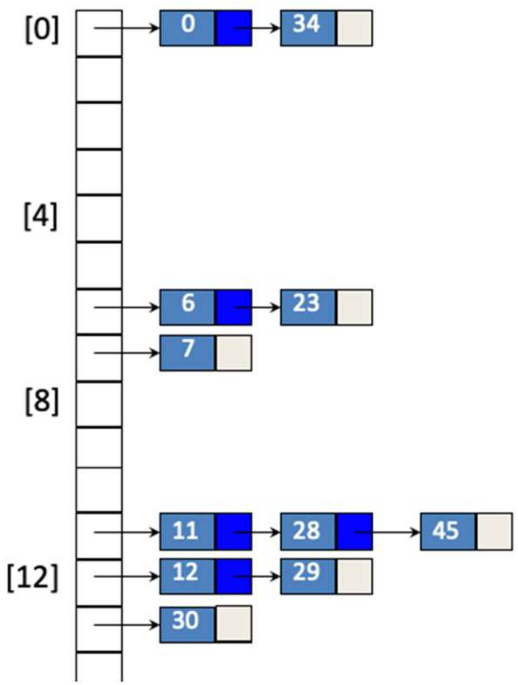
Lecture 13:
Ordered Maps and Skip Lists

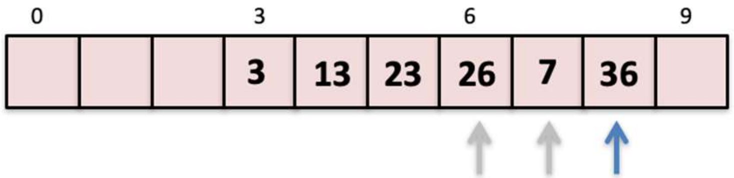
Acknowledgment: The content of this file is based on the slides of the textbook as well as the slides provided by Prof. Won-Ki Jeong and Prof. Tsz-Chiu Au.

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Recap: Hashing

- Hash function
  - Easy to compute
  - Avoid collision (distribute uniformly)
- Overflow handling
  - Open addressing
    - linear probing
    - quadratic probing
  - Chaining





Recap: Hashing

- Clustering
  - Get larger with higher load factor
  - Search / Insert / Delete closer to O(n)
  - Increase both successful & unsuccessful search
  - Cause unnecessary searches (with different bucket keys)
  - Can be relieved with more randomized probing
  - Using tombstone to mark *deleted* has trade-off
- Chaining
  - Less calculation for addressing, but extra memory use

Outline

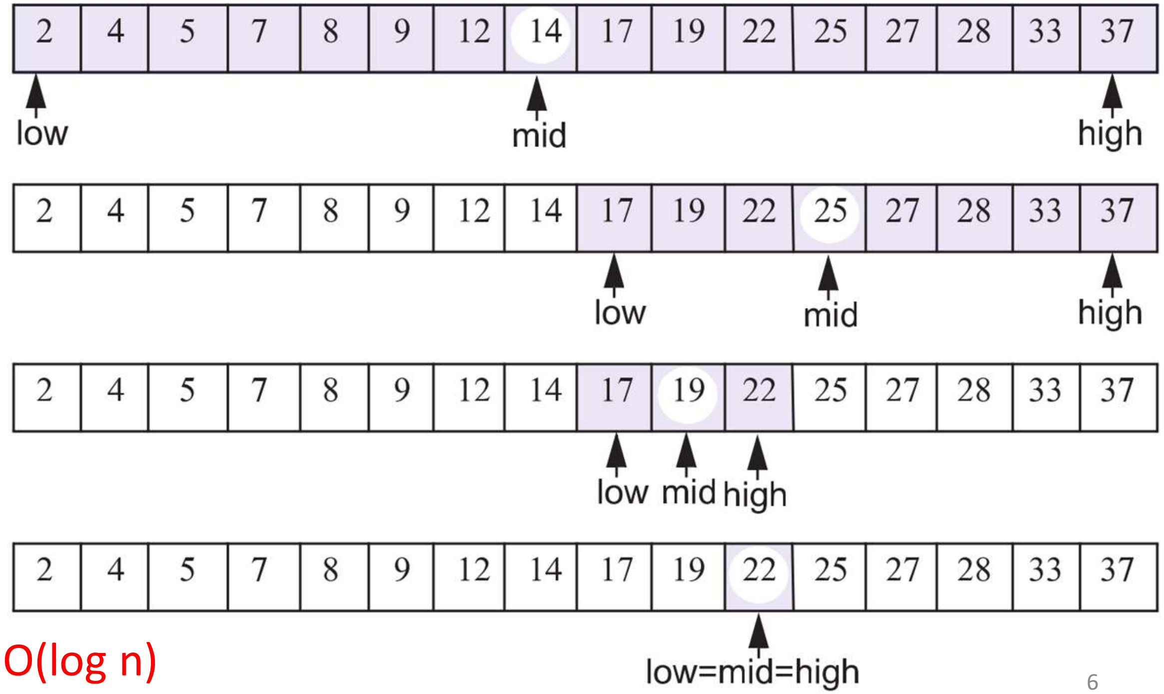
- Ordered Maps
- Skip Lists

Ordered Maps

- Map
  - stores key-value pairs (k,v), called entries
  - quickly locate entries using keys
  - data structures: list, hash table
- Ordered Map
  - Map, with order relation among keys
  - data structures: ordered search tree, skip-list

Ordered Search Table

- Entries are sorted (search table)
  - Arrays allow for easier indexing
  - Binary search
  - High, low, mid (= $\lfloor(\mathit{low}+\mathit{high})/2\rfloor$)



Find(K) : O(log n)

Comparing Map Implementations

Method	*List*	*Hash Table*	*Search Table*
size, empty	$O(1)$	$O(1)$	$O(1)$
find	$O(n)$	$O(1)$ exp., $O(n)$ worst-case	$\boxed{O(\log n)}$
insert	$\boxed{O(1)}$	$\boxed{O(1)}$	$O(n)$
erase	$O(n)$	$\boxed{O(1) \text{ exp., } O(n) \text{ worst-case}}$	$O(n)$

Outline

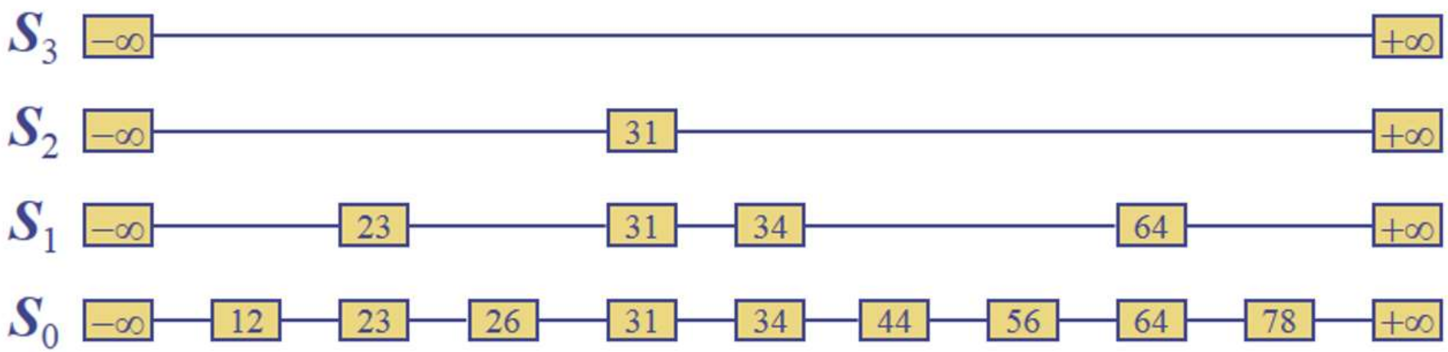
- Ordered Maps
- Skip Lists

Skip List

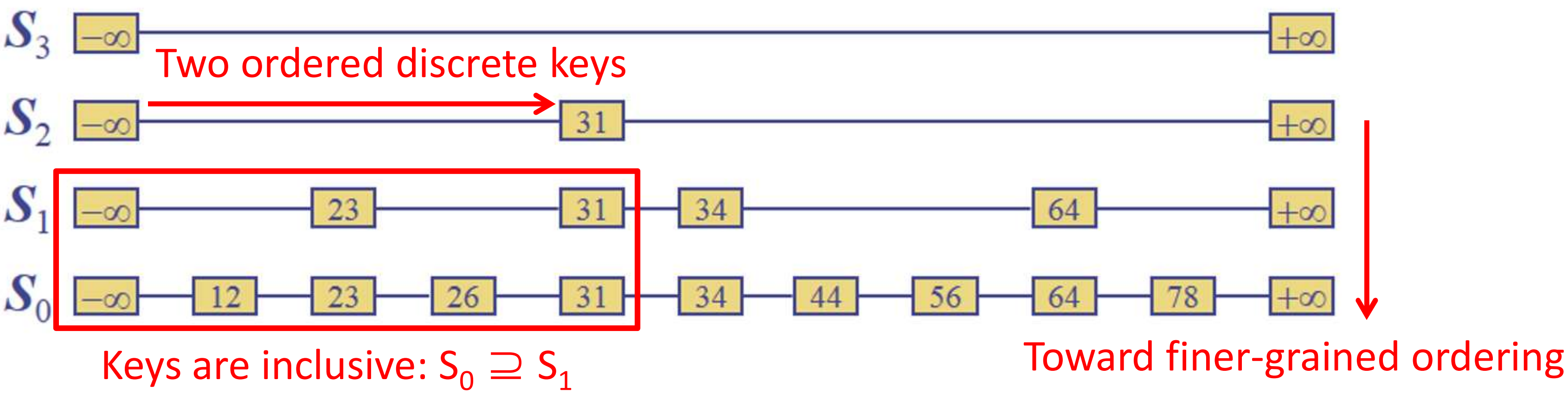
- Skip list: probabilistic data structure
  - William Pugh. Communications of the ACM. 33 (6):668-676, 1990.
- A skip list for a set **S** of distinct (key, value) items is a series of lists $S_0, S_1, \dots, S_h$ such that
  - Each list S_i contains the special keys $+\infty$ and $-\infty$
  - List S_0 contains all the keys of **S** in nondecreasing order
  - Each list is a subsequence of the previous one

$$S_0 \supseteq S_1 \supseteq \dots \supseteq S_h$$

  - List S_h contains only two special keys

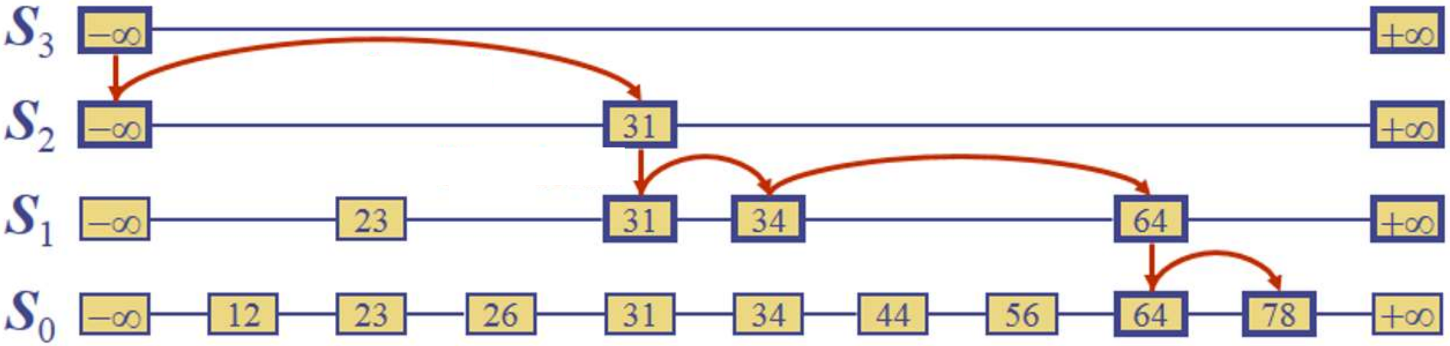


Properties



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

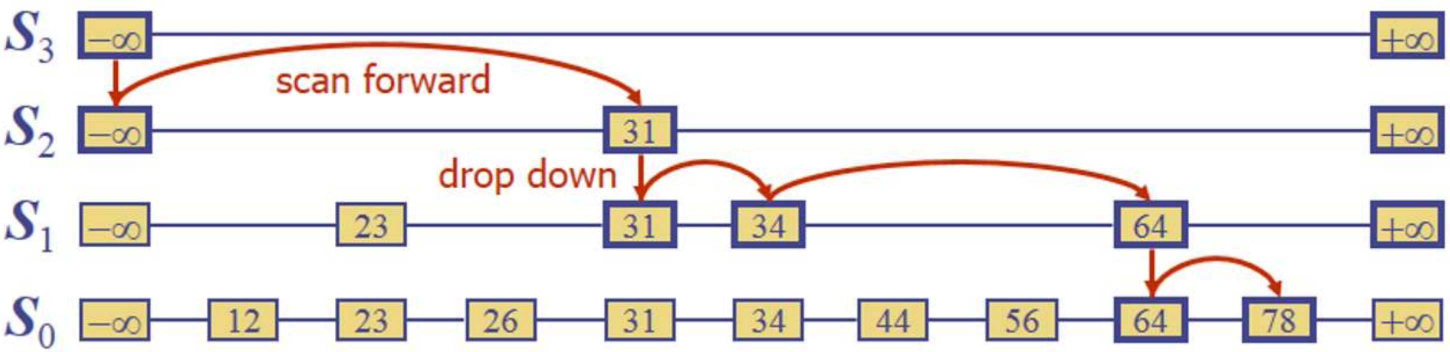
Search



Example: search for 78

Search

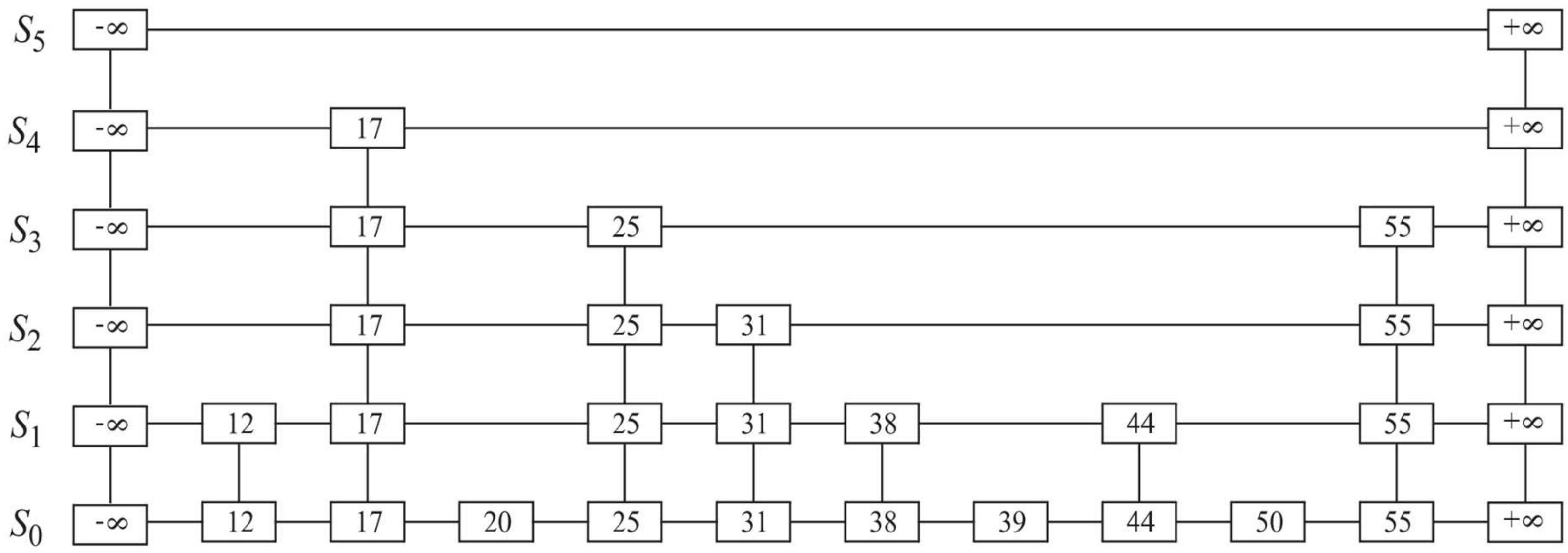
- We search for a key *x* in a skip list as follows:
  - We start at the first position of the top list
  - At the current position *p*, we compare *x* with $y \leftarrow$ *key(next(p))*
    - $x = y$: we return *value(next(p))*
    - $x > y$: we "*scan forward*"
    - $x < y$: we "drop down"
  - If we try to drop down past the bottom list, we return *null*



Example: search for 78

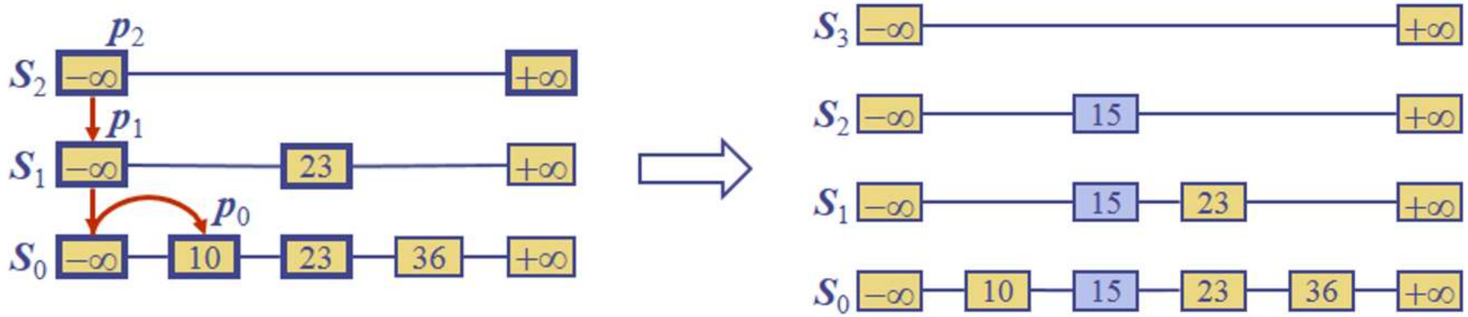
Insertion: What We Want

- About a half of items are promoted to next level
  - Purely driven by random events (e.g., coin tossing)
    - Probabilistic: uses probability
  - Height is approximately *log n*



Insertion

- To insert an entry *(x, o)* into a skip list:
  - We pick ***i*** through "randomized algorithm" such that its probability becomes $1/2^i$
    - *E.g.,* getting *i* consecutive heads when tossing a coin
  - If $i \geq h$, we add new lists $S_{h+1}, \ldots, S_{i+1}$, each containing special keys
  - We find the positions $p_0, p_1, \ldots, p_i$ of the items with largest key less than *x* in each list $S_0, S_1, \ldots, S_i$
  - For $j \leftarrow 0, \ldots, i$, we insert item *(x, o)* into list S_j after position p_j

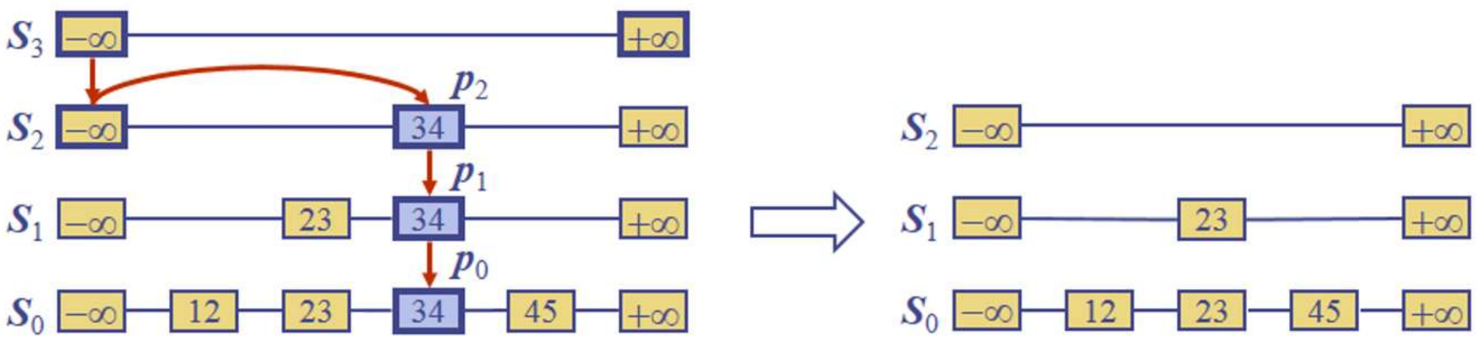


Example: insert key 15, with *i* = *2*

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Deletion

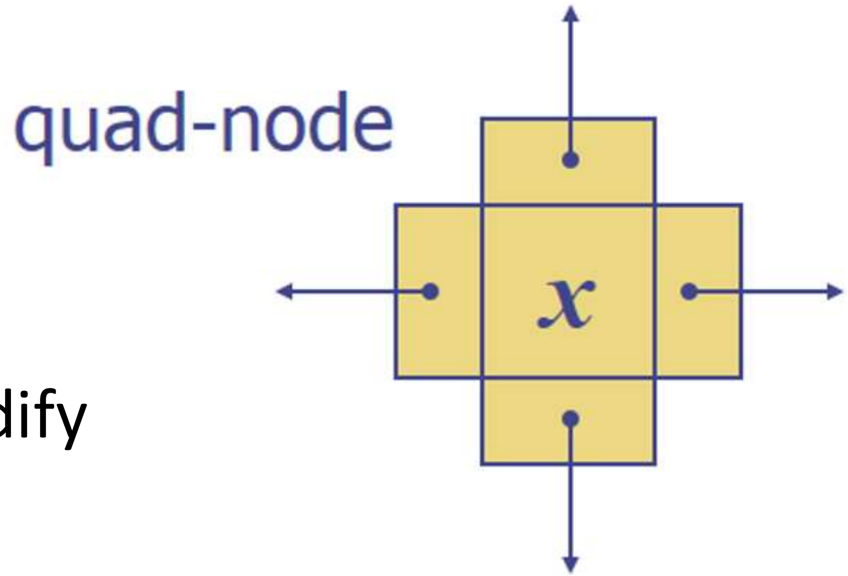
- To remove an entry with key x from a skip list:
  - We find the positions $p_0, p_1, \ldots, p_i$ of the items with key x, where position p_j is in list S_j
  - We remove positions $p_0, p_1, \ldots, p_i$ from the lists $S_0, S_1, \ldots, S_i$
  - We remove all but one list containing only special keys



Example: remove key 34

Implementation

- Use quad-node
  - entry
  - Links to four nodes
    - (prev, next, below, above)
- Also, we define special keys (+∞ & -∞) and modify the key comparator to handle them



Space Usage

- Two probability facts:
  1) Getting *i* consecutive heads when tossing a coin: $1/2^i$
  2) If each of *n* entries is present in a set with probability *p*, the expected size of the set is *np*
- Consider a skip list with *n* entries
  - By 1) and 2), the expected size of list S_i is $n/2^i$
- The number of nodes used by the skip list is

$$\sum_{i=0}^{h} \frac{n}{2^i} = n \sum_{i=0}^{h} \frac{1}{2^i} < 2n$$

Expected space usage: O(n)

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Comparison

Method	List	Hash Table	Ordered List	Skip List
size, empty	O(1)	O(1)	O(1)	O(1)
find	O(n)	O(1)/O(n)	O(log n)	O(log n)/O(n)
insert	O(1)	O(1)	O(n)	O(log n)/O(n)
erase	O(n)	O(1)/O(n)	O(n)	O(log n)/O(n)

Expected/Worst

Questions?