COMP 446 / 546 ALGORITHM DESIGN AND ANALYSIS

LECTURE 8 SKIP LISTS

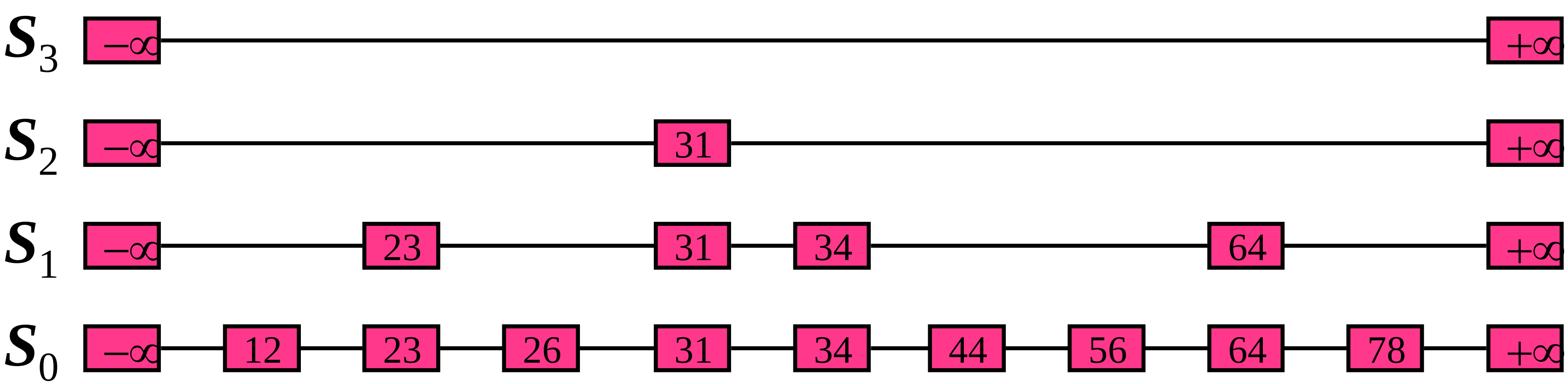
ALPTEKİN KÜPÇÜ

Based on slides of Michael Goodrich, Roberto Tamassia, Erik Demaine, and Shafi Goldwasser

SKIP LIST

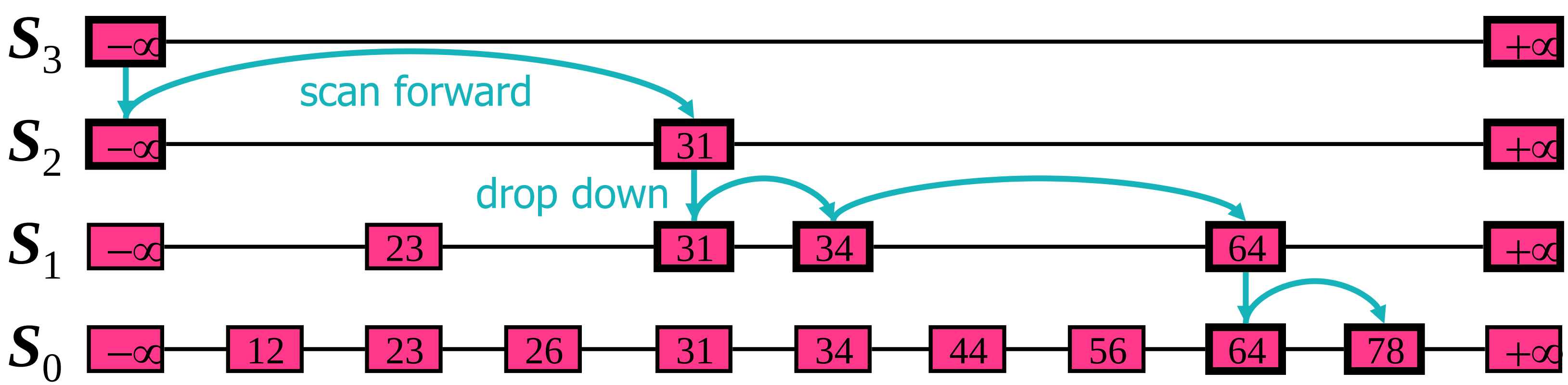
- **A skip list for a set S of distinct (key, element) items is a series of linked lists S_0, S_1, … , S_h such that**
  - Each list S_i contains the special keys $+\infty$ and $-\infty$
  - List S_0 contains the keys of S in non-decreasing order
  - Each list is a subsequence of the previous one, i.e.,

    $$S_0 \supseteq S_1 \supseteq \ \dots \supseteq S_h$$
  - List S_h contains only the two special keys $+\infty$ and $-\infty$



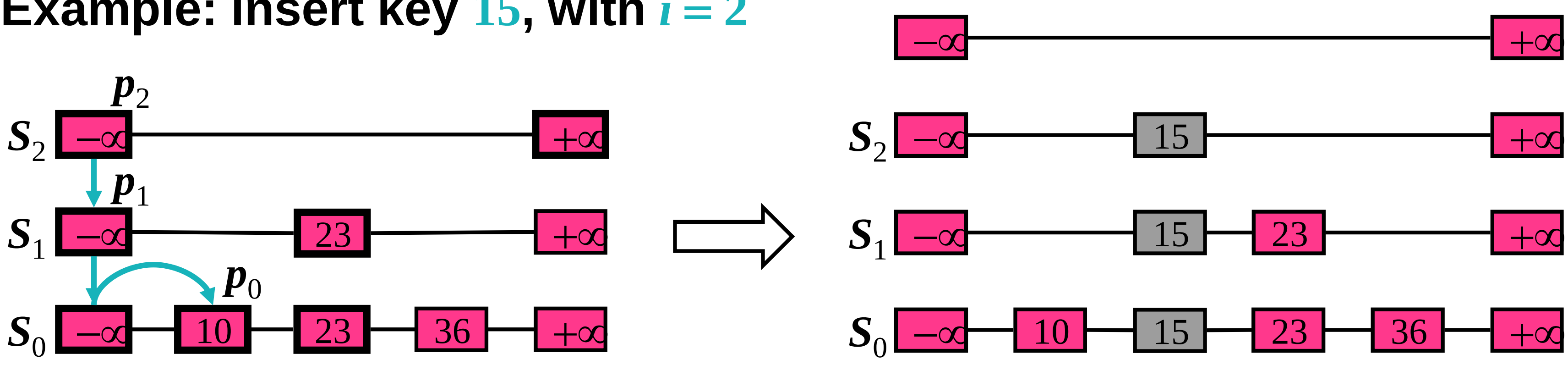
SKIP LIST ALGORITHMS: SEARCH

- **We search for a key x in a a skip list as follows:**
  - We start at the first position of the top list (the root)
  - At the current position p, we compare x with $y \leftarrow key(next(p))$
    - $x = y$: we return $element(next(p))$
    - $x > y$: 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**



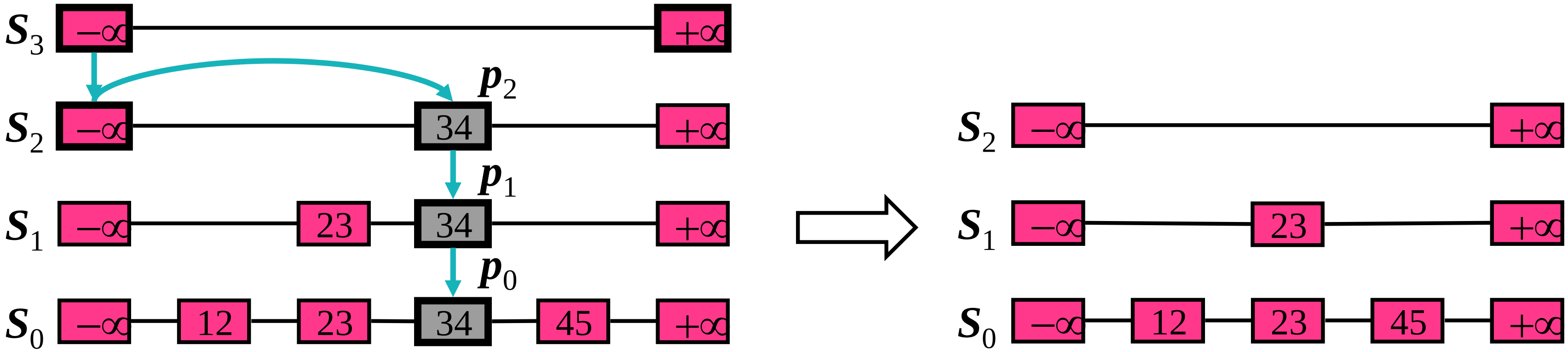
SKIP LIST ALGORITHMS: INSERT

- **To insert an entry (x, o) into a skip list, we use a randomized algorithm:**
  - We repeatedly toss a coin until we get tails, and we denote with i the number of times the coin came up heads
  - If $i \geq h$, we add to the skip list new lists $S_{h+1}, \ldots, S_{i+1}$, each containing only the two special keys $+\infty$ and $-\infty$
  - We search for x in the skip list and 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 $j \leftarrow 0..i$, we insert item (x, o) into list S_j after position p_j
- **Example: insert key 15, with $i = 2$**



SKIP LIST ALGORITHMS: DELETE

- **To delete an entry with key x from a skip list:**
  - We search for x in the skip list and 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 the two special keys
- **Example: delete key 34**



SPACE COMPLEXITY

- Remember: We repeatedly toss a coin until we get tails, and we denote with i the number of times the coin came up heads. The element goes up i levels. Thus, on average:
  - 1/2 the elements go up to level 1
  - 1/4 the elements go up to level 2
  - 1/8 the elements go up to level 3, etc.
- We insert each entry in list S_i with probability $1/2^i$
- The expected size of list S_i is $n/2^i$
- The expected total number of nodes in a skip list with n elements is

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

- The expected space complexity of a skip list with n elements is $O(n)$

HEIGHT OF A SKIP LIST

- **Theorem: With high probability, a skip list with n elements has O(log n) levels**
- **An event occurs E with high probability if, for any $k \geq 1$, there is an appropriate choice of constants for which *E* occurs with probability at least $1 - O(1/n^k)$**
- **Proof:**
  - Remember, we insert an entry in list S_i with probability $1/2^i$
  - Thus, Pr[element x is in level $i = c \log n$] = $½^{c \log n} = 1/n^c$
  - Recall Boole's inequality / union bound:
    - $Pr[E_1 \cup E_2 \cup \ldots \cup E_n] \leq Pr[E_1] + Pr[E_2] + \ldots + Pr[E_n]$
    - Hence, the probability that list S_i has at least one item is at most $n/2^i$
  - Pr[any element is in more than $c \log n$ levels] $\leq n/n^c = 1/n^{c-1}$
  - Let $c \geq 2$ be a constant. A skip list with n entries has height at most $O(c \log n)$ with probability at least $1 - 1/n^{c-1}$

RUNNING TIME ANALYSIS: SEARCH, INSERT, DELETE

- **The search time in a skip list is proportional to**
  - the number of drop-down steps, plus
  - the number of scan-forward steps
- **The drop-down steps are bounded by the height of the skip list and thus are $O(\log n)$ with high probability**
- **Each level contains half of the elements of the level below, in expectation.**
  - Hence, a scan-forward at level i skips roughly $n/2^{h-i}$ of the elements.
  - Thus, the expected number of scan-forward steps is $O(\log n)$
- **Therefore a search in a skip list takes $O(\log n)$ expected time.**
- **The analysis of insertion and deletion are very similar, and both take $O(\log n)$ expected time.**

CONCLUSIONS

- **In a skip list with n entries**
  - The expected space used is $O(n)$
  - The expected height is $O(\log n)$
  - The expected search, insertion and deletion time is $O(\log n)$
  - All these bounds hold with high probability
- **Skip lists are fast in practice and simple to implement.**
- **They are nice alternatives to balanced trees.**