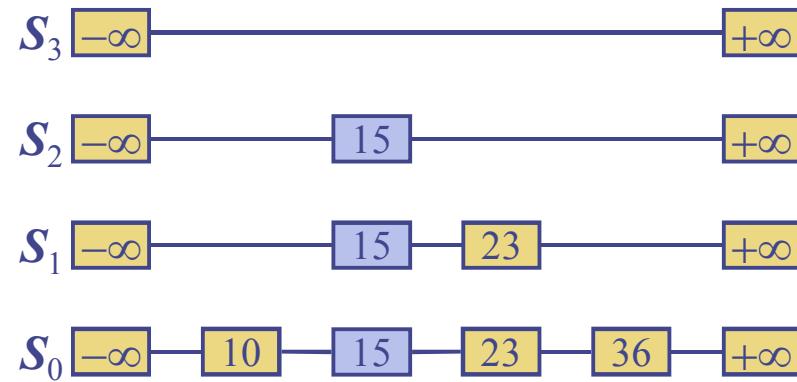


Skip Lists



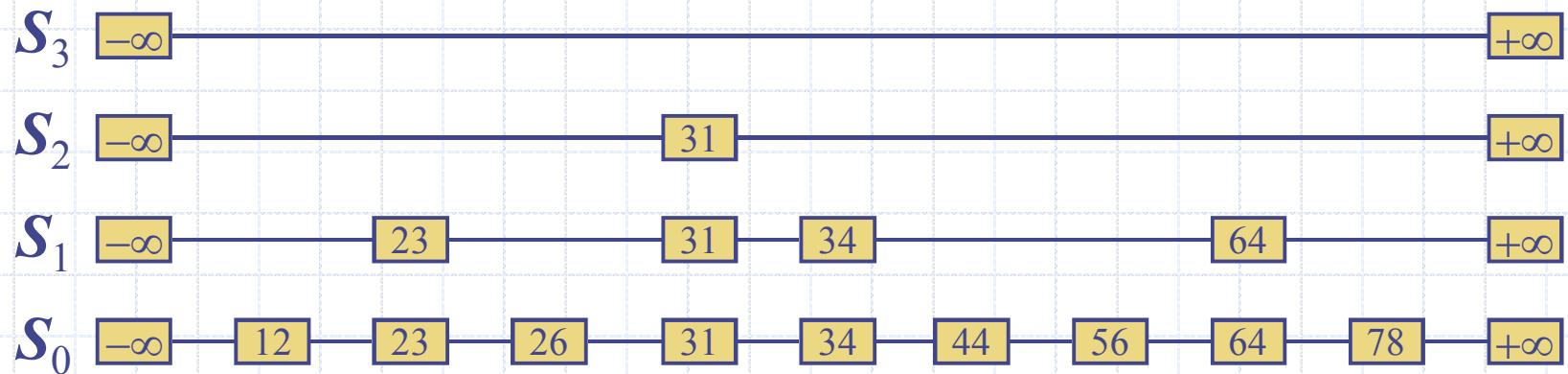
What is a Skip List

A skip list for a set S of distinct (key, element) 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 the keys of S in nondecreasing order
- Each list is a subsequence of the previous one, i.e.,

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

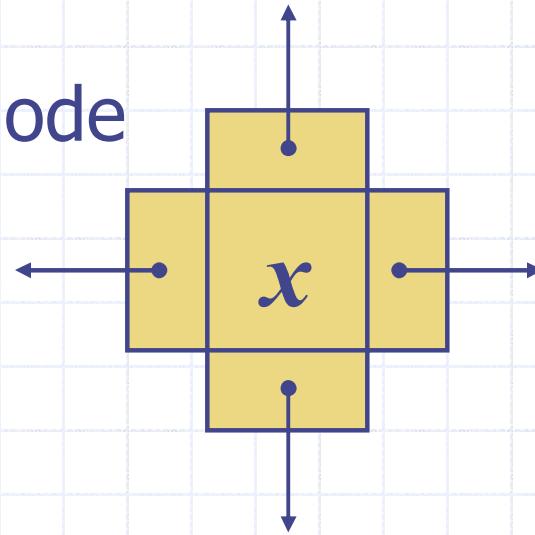
- List S_h contains only the two special keys



Implementation

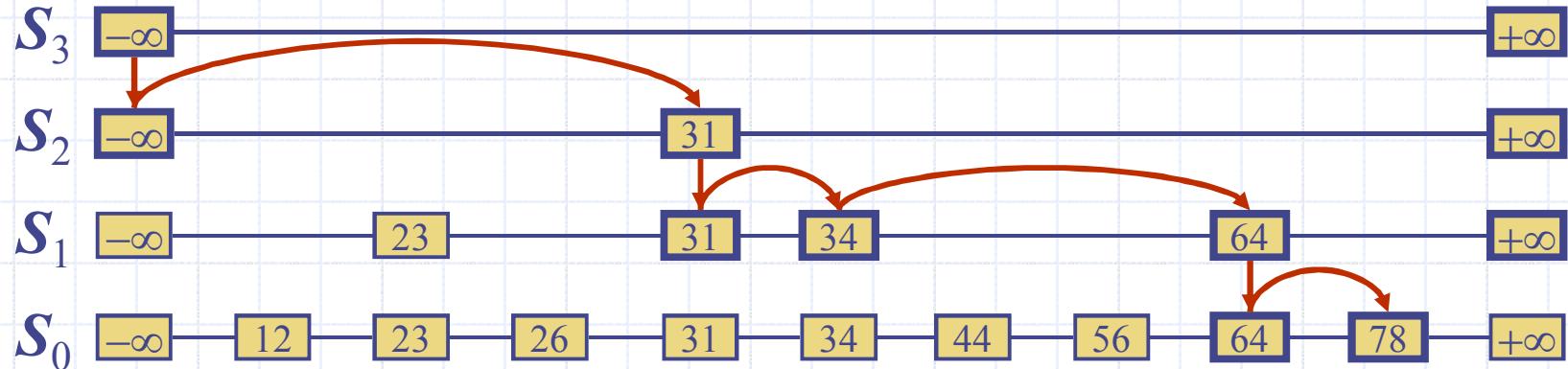
- ◆ We can implement a skip list with quad-nodes
- ◆ A quad-node stores:
 - entry
 - link to the node prev
 - link to the node next
 - link to the node below
 - link to the node above
- ◆ Also, we define special keys PLUS_INF and MINUS_INF, and we modify the key comparator to handle them

quad-node



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 $y \leftarrow \text{key}(\text{after}(p))$
 - $x = y$: we return *element*($\text{after}(p)$)
 - $x > y$: we “scan forward”
 - $x < y$: we “drop down”
 - If we try to drop down past the bottom list, we return *NO SUCH KEY*
- ◆ Example: search for 78



Randomized Algorithms

- ◆ A randomized algorithm performs coin tosses (i.e., uses random bits) to control its execution
- ◆ It contains statements of the type

```
b ← random()
```

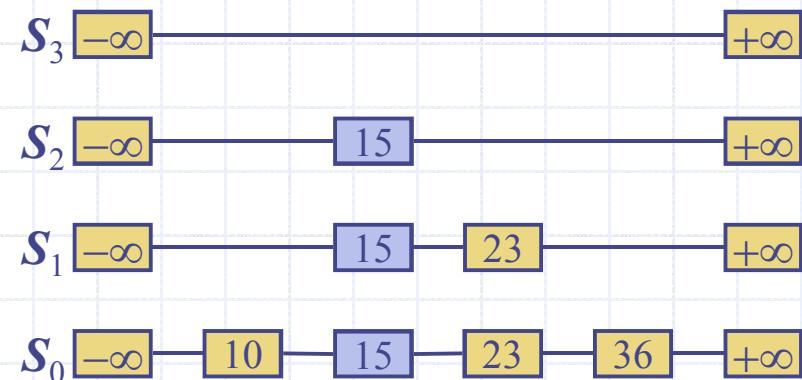
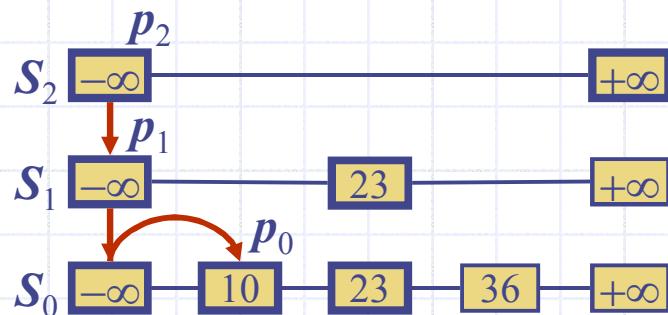
```
if b = 0  
    do A ...  
else { b = 1 }  
    do B ...
```

- ◆ Its running time depends on the outcomes of the coin tosses

- ◆ We analyze the expected running time of a randomized algorithm under the following assumptions
 - the coins are unbiased, and
 - the coin tosses are independent
- ◆ The worst-case running time of a randomized algorithm is often large but has very low probability (e.g., it occurs when all the coin tosses give "heads")
- ◆ We use a randomized algorithm to insert items into a skip list

Insertion

- ◆ The insertion algorithm uses randomization to decide how many references to the new item x should be added to the skip list.
- ◆ To insert an item x 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}, \dots, S_{i+1} , each containing only the two special keys
 - We search for x in the skip list and find the positions p_0, p_1, \dots, p_i of the items with largest key less than x in each list S_0, S_1, \dots, S_i
 - For $j \leftarrow 0, \dots, i$, we insert item x into list S_j after position p_j
- ◆ Example: insert key 15, with $i = 2$

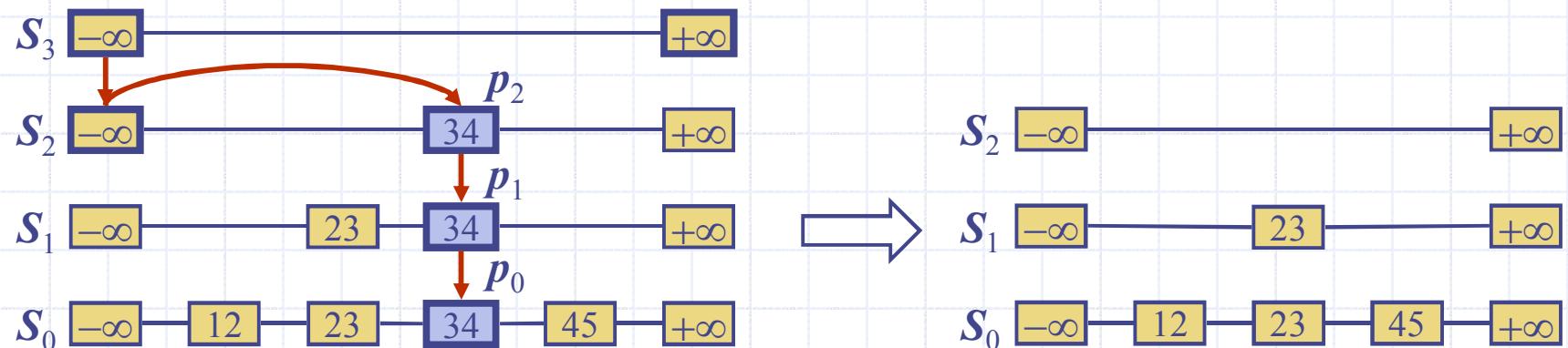


Deletion

◆ To remove an item with key x from a skip list, we proceed as follows:

- We search for x in the skip list and find the positions p_0, p_1, \dots, p_i of the items with key x , where position p_j is in list S_j
- We remove positions p_0, p_1, \dots, p_i from the lists S_0, S_1, \dots, S_i
- We remove all but one list containing only the two special keys

◆ Example: remove key 34



Space Usage

- ◆ The space used by a skip list depends on the random bits used by each invocation of the insertion algorithm
- ◆ We use the following two basic probabilistic facts:

Fact 1: The probability of getting i consecutive heads when flipping a coin is $1/2^i$

Fact 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 Fact 1, we insert an entry in list S_i with probability $1/2^i$
 - By Fact 2, the expected size of list S_i is $n/2^i$

- ◆ The expected number of nodes used by the skip list is

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

- ◆ Thus, the expected space usage of a skip list with n items is $O(n)$

Height

- ◆ The running time of the search and insertion algorithms is affected by the height h of the skip list
- ◆ We show that with high probability, a skip list with n items has height $O(\log n)$
- ◆ We use the following additional probabilistic fact:

Fact 3: If each of n events has probability p , the probability that at least one event occurs is at most np
- ◆ Consider a skip list with n entries
 - By Fact 1, we insert an entry in list S_i with probability $1/2^i$
 - By Fact 3, the probability that list S_i has at least one item is at most $n/2^i$
- ◆ By picking $i = 3\log n$, we have that the probability that $S_{3\log n}$ has at least one entry is at most
$$n/2^{3\log n} = n/n^3 = 1/n^2$$
- ◆ Thus a skip list with n entries has height at most $3\log n$ with probability at least $1 - 1/n^2$

Search and Update Times

- ◆ 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
- ◆ To analyze the scan-forward steps, we use yet another probabilistic fact:

Fact 4: The expected number of coin tosses required in order to get tails is 2
- ◆ When we scan forward in a list, the destination key does not belong to a higher list
 - A scan-forward step is associated with a former coin toss that gave tails
- ◆ By Fact 4, in each list the expected number of scan-forward steps is 2
- ◆ Thus, the expected number of scan-forward steps is $O(\log n)$
- ◆ We conclude that a search in a skip list takes $O(\log n)$ expected time
- ◆ The analysis of insertion and deletion gives similar results

Summary

- ◆ A skip list is a data structure for dictionaries that uses a randomized insertion algorithm
- ◆ In a skip list with n entries
 - The expected space used is $O(n)$
 - The expected search, insertion and deletion time is $O(\log n)$
- ◆ Using a more complex probabilistic analysis, one can show that these performance bounds also hold with high probability
- ◆ Skip lists are fast and simple to implement in practice

Draw a Skip List

Draw the skip list that results from inserting 28, deleting 34, and then inserting 54 in the following skip list.

Assume the coin flips for the first insertion yield two heads followed by a tail, and those for the second insertion yield three heads followed by a tail.

