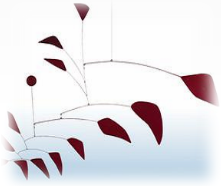


EECS 4101-5101
Advanced Data Structures

Shahin Kamali

Topic 6: Randomized Data Structures

York University

Picture is from the cover of the textbook CLRS.

Objectives

- By the end of this module, you will be able to:
  - Describe advantages of randomization in designing data structures with improved **expected** running time.

Objectives

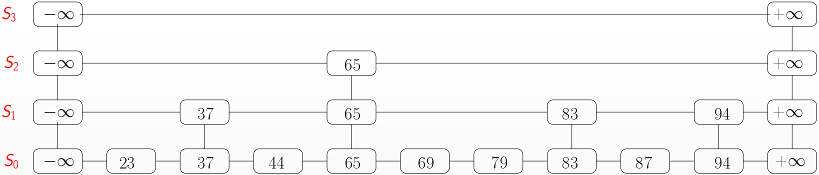
- By the end of this module, you will be able to:
  - Describe advantages of randomization in designing data structures with improved **expected** running time.
  - Compare and contrast randomized and deterministic data structures for implementing an abstract data type.

Objectives

- By the end of this module, you will be able to:
  - Describe advantages of randomization in designing data structures with improved **expected** running time.
  - Compare and contrast randomized and deterministic data structures for implementing an abstract data type.
  - Describe randomized data structures for Dictionaries and analyze their space and time complexity.

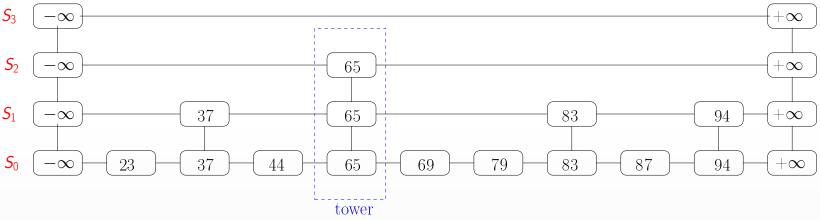
Skip Lists

- **Randomized** data structure for dictionary ADT
- A hierarchy of ordered linked lists
- A **skip list** for a set S of items is a series of lists $S_0, S_1, \cdots, S_h$ 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., $S_0 \supseteq S_1 \supseteq \cdots \supseteq S_h$
  - List S_h contains only the two special keys



Skip Lists

- A skip list for a set S of items is a series of lists $S_0, S_1, \cdots, S_h$
- A two-dimensional collection of positions: levels and towers
- Traversing the skip list: after(p), below(p)



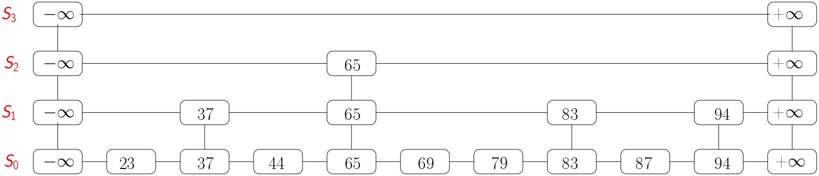
Search in Skip Lists

```
skip-search(L, k)
L: A skip list, k: a key
    p ← topmost left position of L
    S ← stack of positions, initially containing p
    while below(p) ≠ null do
        p ← below(p)
        while key(after(p)) < k do
            p ← after(p)
        push p onto S
    return S
```

- S contains positions of the largest key **less than** k at each level.
- $after(top(S))$ will have key k, iff k is in L.
- drop down: $p \leftarrow below(p)$
- scan forward: $p \leftarrow after(p)$

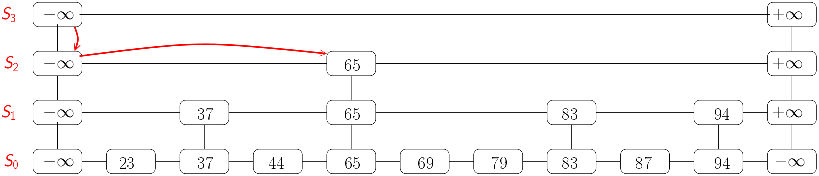
Search in Skip Lists

Example: Skip-Search$(S, 87)$



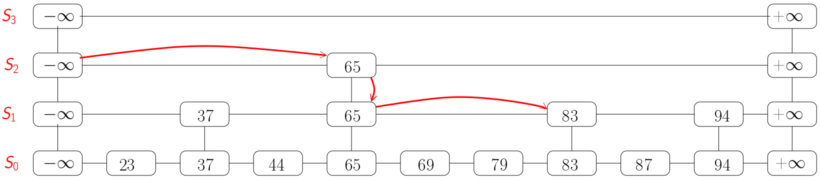
Search in Skip Lists

Example: Skip-Search$(S, 87)$



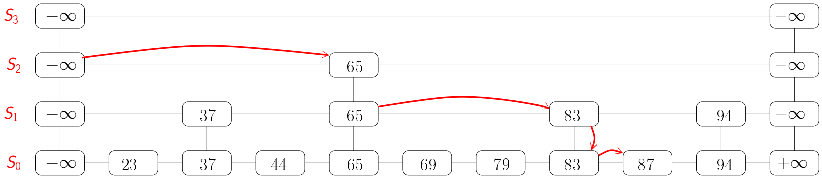
Search in Skip Lists

Example: Skip-Search$(S, 87)$



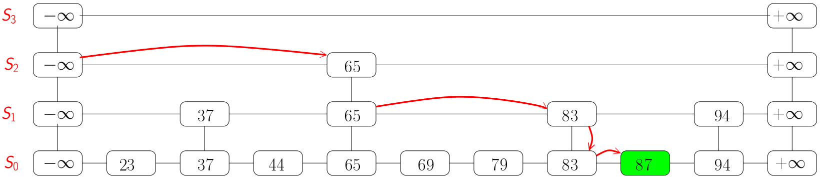
Search in Skip Lists

Example: Skip-Search$(S, 87)$



Search in Skip Lists

Example: Skip-Search$(S, 87)$

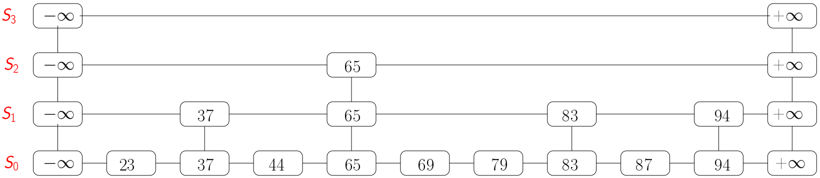


Insert in Skip Lists

- *Skip-Insert*(S, k, v)
  - Randomly compute the height of new item: repeatedly toss a coin until you get tails, let i the number of times the coin came up heads
  - Search for k in the skip list and find the positions $p_0, p_1, \cdots, p_i$ of the items with largest key less than k in each list $S_0, S_1, \cdots, S_i$ (by performing *Skip-Search*(S, k))
  - Insert item (k, v) into list S_j after position p_j for $0 \leq j \leq i$ (a tower of height i)

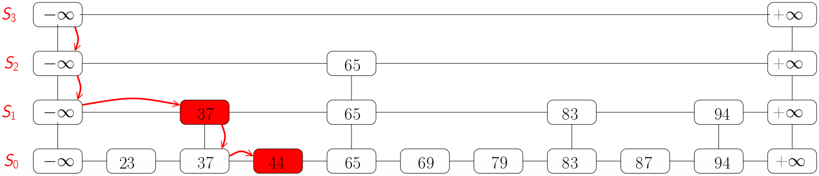
Insert in Skip Lists

Example: Skip-Insert($S, 52, v$)
Coin tosses: H,T $\Rightarrow i = 1$



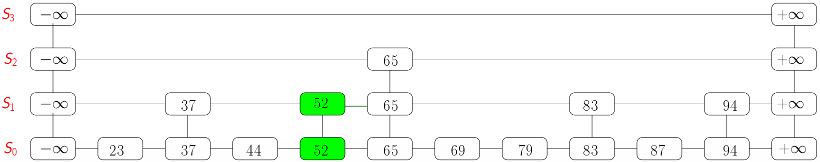
Insert in Skip Lists

Example: Skip-Insert$(S, 52, v)$
Coin tosses: H,T $\Rightarrow i = 1$
Skip-Search$(S, 52)$



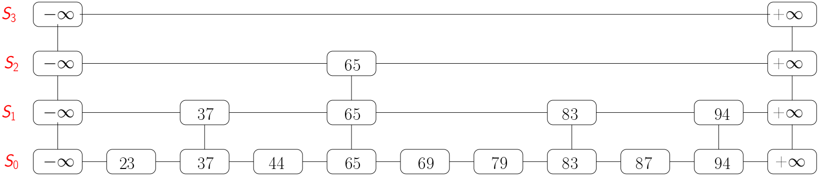
Insert in Skip Lists

Example: Skip-Insert($S, 52, v$)
Coin tosses: H,T $\Rightarrow i = 1$



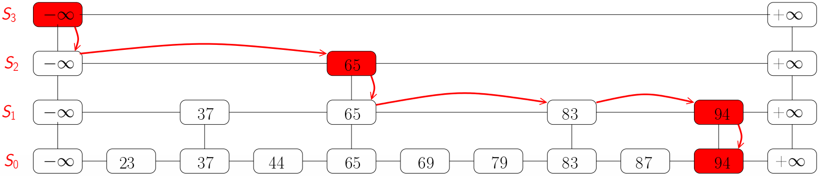
Insert in Skip Lists

Example: Skip-Insert($S, 100, v$)
Coin tosses: H,H,H,T $\Rightarrow i = 3$



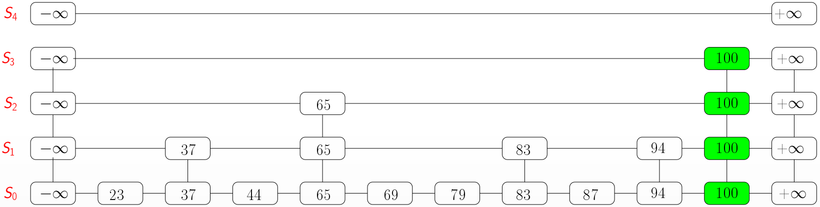
Insert in Skip Lists

Example: Skip-Insert(S, 100, v)
Coin tosses: H,H,H,T $\Rightarrow i = 3$
Skip-Search(S, 100)



Insert in Skip Lists

Example: Skip-Insert($S, 100, v$)
Coin tosses: H,H,H,T $\Rightarrow i = 3$
Height increase

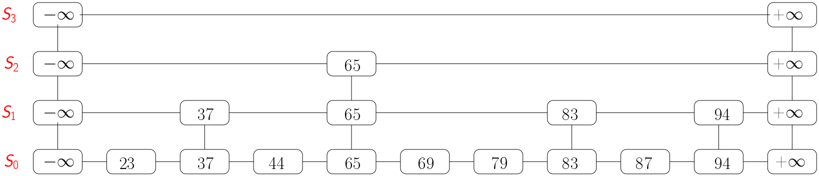


Delete in Skip Lists

- **Skip-Delete** (S, k)
  - Search for k in the skip list and find all the positions $p_0, p_1, \ldots, p_i$ of the items with the largest key smaller than k, where p_j is in list S_j. (this is the same as Skip-Search)
  - For each i, if $key(after(p_i)) == k$, then remove $after(p_i)$ from list S_i
  - Remove all but one of the lists S_i that contain only the two special keys

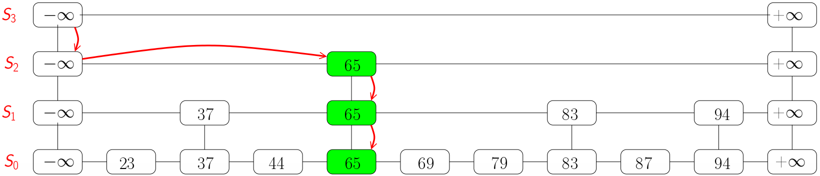
Delete in Skip Lists

Example: Skip-Delete(S, 65)



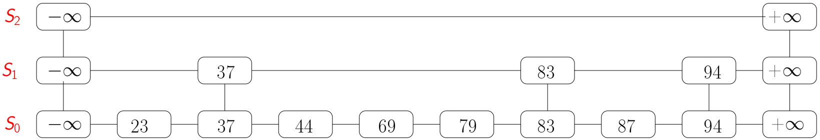
Delete in Skip Lists

Example: Skip-Delete(S, 65)
Skip-Search(S, 65)



Delete in Skip Lists

Example: Skip-Delete(S, 65)



Skip List Memory Complexity

- What is the expected height of a tower?
  - 1 if random flip sequence is T, 2 if it is H, T, 3 if it is H, H, T.

Skip List Memory Complexity

- What is the expected height of a tower?
  - 1 if random flip sequence is T, 2 if it is H, T, 3 if it is H, H, T.
  - The chance of a tower having height i is $\frac{1}{2^i}$.
    - For that the first $i-1$ flips should be heads and the i'th one a tail.

Skip List Memory Complexity

- What is the expected height of a tower?
  - 1 if random flip sequence is T, 2 if it is H, T, 3 if it is H, H, T.
  - The chance of a tower having height i is $\frac{1}{2^i}$.
    - For that the first $i - 1$ flips should be heads and the i'th one a tail.
  - The expected height of a tower will be $X = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \ldots$

Skip List Memory Complexity

- What is the expected height of a tower?
  - 1 if random flip sequence is T, 2 if it is H, T, 3 if it is H, H, T.
  - The chance of a tower having height i is $\frac{1}{2^i}$.
    - For that the first $i-1$ flips should be heads and the i'th one a tail.
  - The expected height of a tower will be $X = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \ldots$
  - We have $X = 1/2 + 2/4 + 3/8 + 4/16 + 5/32 + 6/64 \ldots$, i.e.,
    $X/2 = 1/4 + 2/8 + 3/16 + 4/32 + 5/64 + \ldots$;
    So, $X - X/2 \leq 1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/64 + \ldots = 1$,
    i.e., $X = 2$.
- So, the expected height of a tower is 2, i.e., the expected size of the skip list is $2n \in \Theta(n)$.

Theorem

A skip list that includes n keys is expected to have $\Theta(n)$ nodes.

Skip List Height

- How many levels are expected to be in a linked list of size n?

$$
\begin{aligned}
\text{Prob}(\text{max height} > c \log n) &= \text{Prob}(\text{some element flipped } > c \log n \text{ heads}) \\
&\leq n \cdot \text{Prob}(\text{element} x \text{ flipped } > c \log n \text{ heads}) \quad [\text{Boole's ineq.}] \\
&= n(1/2)^{c \log n} = n/n^c = \frac{1}{n^{c-1}}
\end{aligned}
$$

Skip List Height

- How many levels are expected to be in a linked list of size n?

$$
\begin{aligned}
\text{Prob}(\text{max height} > c \log n) &= \text{Prob}(\text{some element flipped} \; > c \log n \text{ heads}) \\
&\leq n \cdot \text{Prob}(\text{element}x \text{ flipped} \; > c \log n \text{ heads}) \quad [\text{Boole's ineq.}] \\
&= n(1/2)^{c \log n} = n/n^c = \frac{1}{n^{c-1}}
\end{aligned}
$$

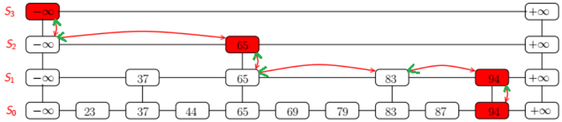
- With a chance of at least $1 - 1/n^{c-1}$, the height of the skip list is at most $c \log n$.
- This can be used to show the number of levels in a skip list is expected to be $\Theta(\log n)$

Theorem

The height of a skip list on n items is expected to be $\Theta(\log n)$.

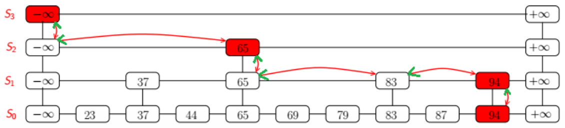
Search Time in Skip Lists

- How many nodes are visited for searching a key k?



Search Time in Skip Lists

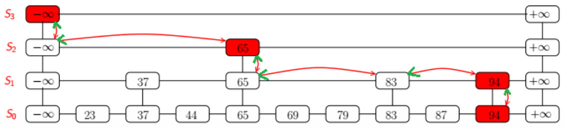
- How many nodes are visited for searching a key k?
- Think of backward moves from the lowest level that includes k
  - If it is possible to go up (the key appears in the next level), we go up (with a chance of 1/2).
  - If not, we stay in the same level and go left (again, with a chance of 1/2).



Search Time in Skip Lists

- How many nodes are visited for searching a key k?
- Think of backward moves from the lowest level that includes k
  - If it is possible to go up (the key appears in the next level), we go up (with a chance of 1/2).
  - If not, we stay in the same level and go left (again, with a chance of 1/2).
- Let $C(j)$ be the maximum number of nodes to be visited when there are j levels above us.
- After a visiting a node at the current level (with cost 1) we have:

$$C(j) \leq 1 + 1/2 \cdot C(j-1) + 1/2 \cdot C(j) \text{ which gives } \quad C(j) \leq 2j$$

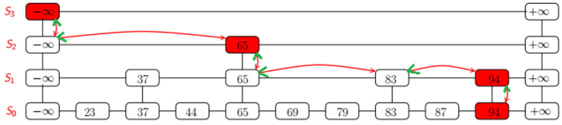


Search Time in Skip Lists

- How many nodes are visited for searching a key k?
- Think of backward moves from the lowest level that includes k
  - If it is possible to go up (the key appears in the next level), we go up (with a chance of 1/2).
  - If not, we stay in the same level and go left (again, with a chance of 1/2).
- Let $C(j)$ be the maximum number of nodes to be visited when there are j levels above us.
- After a visiting a node at the current level (with cost 1) we have:

$$C(j) \leq 1 + 1/2 \cdot C(j-1) + 1/2 \cdot C(j) \text{ which gives } \quad C(j) \leq 2j$$

- From the previous slide, we know j is expected to be $\Theta(\log n)$.



Search Time in Skip Lists

Theorem

The number of nodes visited when searching for an item in the skip list of n keys is expected to be $\Theta(\log n)$.

- For insert, we do search and add an expected $\Theta(1)$ number of nodes; search time dominates.
- Similarly, for delete, search time dominates.

Summary of Skip Lists

- Expected **space** usage: $O(n)$
- Expected **height**: $O(\log n)$
- *Skip-Search*: $O(\log n)$ expected time
- *Skip-Insert*: $O(\log n)$ expected time
- *Skip-Delete*: $O(\log n)$ expected time
- Skip lists are fast and simple to implement in practice