

ACOL106: Data Structures and Algorithms

Ragesh Jaiswal, IIT Delhi

Randomised Data Structure: Skip List

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining $(key, value)$ pairs with the following running time:
 - *Search*: $O(\log n)$
 - *Insert*: $O(\log n)$
 - *Delete*: $O(\log n)$
- Is there such a data structure?

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining $(key, value)$ pairs with the following running time:
 - *Search*: $O(\log n)$
 - *Insert*: $O(\log n)$
 - *Delete*: $O(\log n)$
- Is there such a data structure?
 - Yes, we have AVL-Tree and $(2, 4)$ -Tree that ensures these running time.
- *Is there something simpler that does not require careful balancing operations... perhaps using randomization... in the spirit of our latest discussion on a randomized hash function?*

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining (key, value) pairs with the following running time:
 - Search: $O(\log n)$
 - Insert: $O(\log n)$
 - Delete: $O(\log n)$
- Is there such a data structure?
 - Yes, we have AVL-Tree and (2, 4)-Tree that ensures these running time.
- *Is there something simpler that does not require careful balancing operations... perhaps using randomization... in the spirit of our latest discussion on a randomized hash function?*
- We'll see a **randomized data structure** that achieves the running time bounds, but only in **expectation**.
 - Randomized data structure: A data structure that uses random coin tosses in operations.

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining $(key, value)$ pairs with the following running time:
 - *Search*: $O(\log n)$
 - *Insert*: $O(\log n)$
 - *Delete*: $O(\log n)$
- Let's try to achieve the above using ideas from array/list and linked-list.

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining $(key, value)$ pairs with the following running time:
 - *Search*: $O(\log n)$
 - *Insert*: $O(\log n)$
 - *Delete*: $O(\log n)$
- Let's try to achieve the above using ideas from array/list and linked-list.
 - A sorted list/array helps in search in $O(\log n)$ time, but insertion/deletion costs $O(n)$ time due to shifting.
 - A linked-list makes it easy to insert/delete elements, but searching for an element costs $O(n)$ time, even when the list is sorted.

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining $(key, value)$ pairs with the following running time:
 - *Search*: $O(\log n)$
 - *Insert*: $O(\log n)$
 - *Delete*: $O(\log n)$
- Let's try to achieve the above using ideas from array/list and linked-list.
 - A sorted list/array helps in search in $O(\log n)$ time, but insertion/deletion costs $O(n)$ time due to shifting.
 - A linked-list makes it easy to insert/delete elements, but searching for an element costs $O(n)$ time, even when the list is sorted.
- A **Skip List** is a compromise between these two ideas. We maintain a sorted linked list with provision for faster search than a linear scan using randomization.

Randomised Data Structure

Skip List

- Suppose we are interested in a data structure for maintaining $(key, value)$ pairs with the following running time:
 - *Search*: $O(\log n)$
 - *Insert*: $O(\log n)$
 - *Delete*: $O(\log n)$
- Let's try to achieve the above using ideas from array/list and linked-list.
 - A sorted list/array helps in search in $O(\log n)$ time, but insertion/deletion costs $O(n)$ time due to shifting.
 - A linked-list makes it easy to insert/delete elements, but searching for an element costs $O(n)$ time, even when the list is sorted.
- A **Skip List** is a compromise between these two ideas. We maintain a sorted linked list with provision for faster search than a linear scan using randomization.

Randomised Data Structure

Skip List

- Skip list through images:

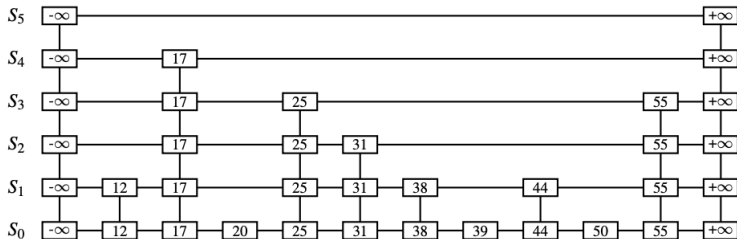


Figure: A node is copied to the subsequent higher level with probability $1/2$. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list through images: Search

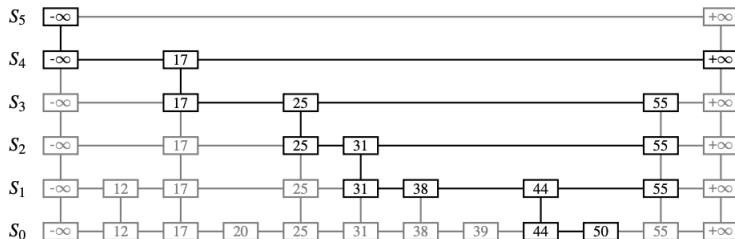


Figure: Searching for 50. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list through images: **Insertion**

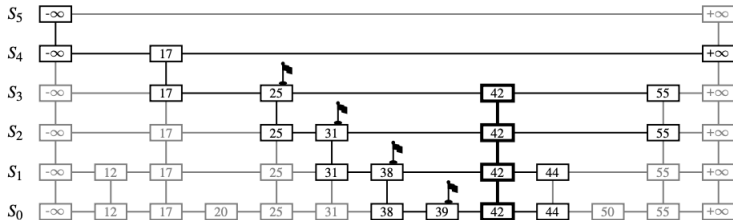


Figure: Inserting 42. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list through images: **Deletion**

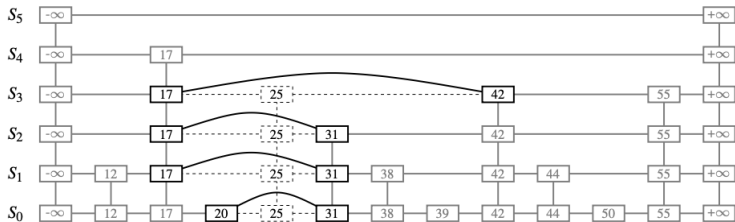


Figure: Deleting 25. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list: **Analysis**

- What is the probability of the i^{th} tower being of height more than $3 \log n$?

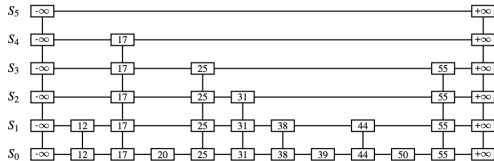


Figure: A node is copied to the subsequent higher level with probability $1/2$. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list: **Analysis**

- What is the probability of the i^{th} tower being of height more than $3 \log n$? $\leq \frac{1}{n^3}$
- What is the probability that the height of the skip list is more than $3 \log n$?

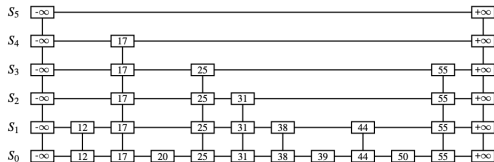


Figure: A node is copied to the subsequent higher level with probability $1/2$. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list: **Analysis**

- What is the probability of the i^{th} tower being of height more than $3 \log n$? $\leq \frac{1}{n^3}$
- What is the probability that the height of the skip list is more than $3 \log n$? $\leq \frac{1}{n^2}$
- What is the expected number of keys examined at a level when searching?

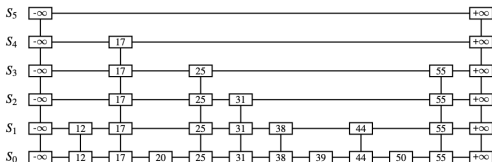


Figure: A node is copied to the subsequent higher level with probability $1/2$. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list: **Analysis**

- What is the probability of the i^{th} tower being of height more than $3 \log n$? $\leq \frac{1}{n^3}$
- What is the probability that the height of the skip list is more than $3 \log n$? $\leq \frac{1}{n^2}$
- What is the expected number of keys examined at a level when searching? $O(1)$

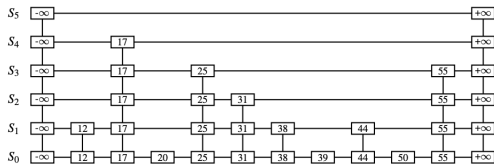


Figure: A node is copied to the subsequent higher level with probability $1/2$. (Picture taken from *Data Structures and Algorithms in Python* by Goodrich, Tamassia, and Goldwasser.)

Randomised Data Structure

Skip List

- Skip list: **Analysis**
 - What is the probability of the i^{th} tower being of height more than $3 \log n$? $\leq \frac{1}{n^3}$
 - What is the probability that the height of the skip list is more than $3 \log n$? $\leq \frac{1}{n^2}$
 - What is the expected number of keys examined at a level when searching? $O(1)$
 - What is the **expected** running time for these operations:
 - Search?
 - Insert?
 - Delete?

Randomised Data Structure

Skip List

- Skip list: **Analysis**
 - What is the probability of the i^{th} tower being of height more than $3 \log n$? $\leq \frac{1}{n^3}$
 - What is the probability that the height of the skip list is more than $3 \log n$? $\leq \frac{1}{n^2}$
 - What is the expected number of keys examined at a level when searching? $O(1)$
 - What is the **expected** running time for these operations:
 - Search? $O(\log n)$
 - Insert? $O(\log n)$
 - Delete? $O(\log n)$
 - What is the **expected** space requirement for a skip list?

Randomised Data Structure

Skip List

- Skip list: **Analysis**
 - What is the probability of the i^{th} tower being of height more than $3 \log n$? $\leq \frac{1}{n^3}$
 - What is the probability that the height of the skip list is more than $3 \log n$? $\leq \frac{1}{n^2}$
 - What is the expected number of keys examined at a level when searching? $O(1)$
 - What is the **expected** running time for these operations:
 - Search? $O(\log n)$
 - Insert? $O(\log n)$
 - Delete? $O(\log n)$
 - What is the **expected** space requirement for a skip list? $O(n)$

End