

COL106: Data Structures and Algorithms

Lecture 12

Keerti Choudhary
Department of Computer Science, IIT Delhi

Combinatorial Tool

Identity:

$$\sum_{n \geq 1} n \cdot \frac{1}{2^n} = 2$$

Proof: Consider the sum $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$, where $|x| < 1$.

On differentiating we get, $\sum_{n=1}^{\infty} n x^{n-1} = \frac{1}{(1-x)^2}$.

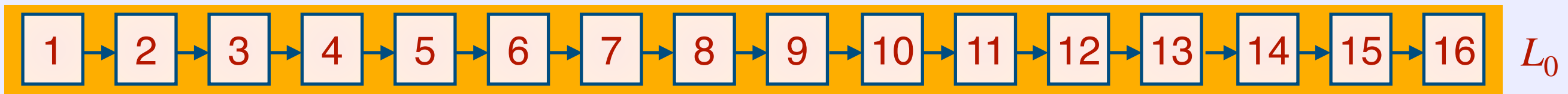
Putting $x = 1/2$ gives us: $\sum_{n=1}^{\infty} n \frac{1}{2^{n-1}} = 4$.

Dividing by two gives us: $\sum_{n=1}^{\infty} n \cdot \frac{1}{2^n} = 2$.

Skip Lists

Generation Rule

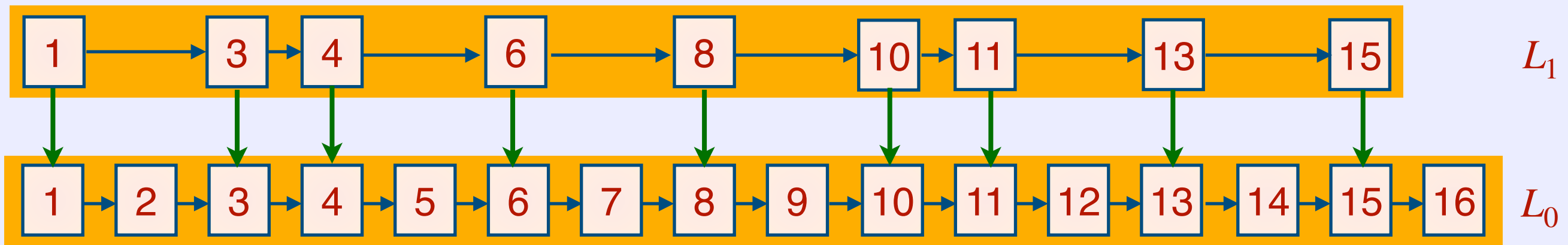
Each element of L_i is chosen
in L_{i+1} with prob $1/2$.



Skip Lists

Generation Rule

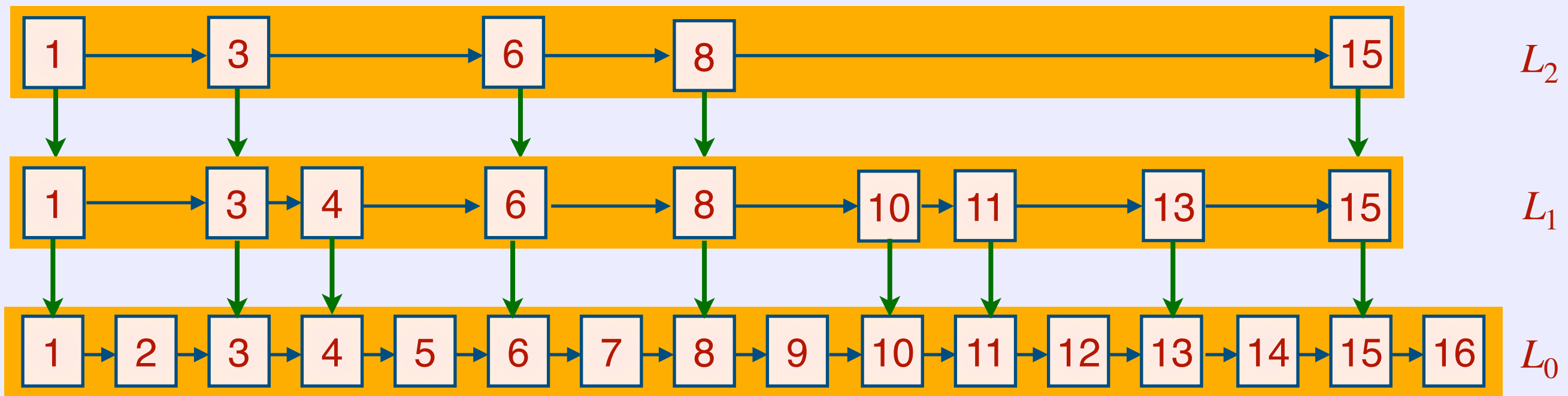
Each element of L_i is chosen in L_{i+1} with prob $1/2$.



Skip Lists

Generation Rule

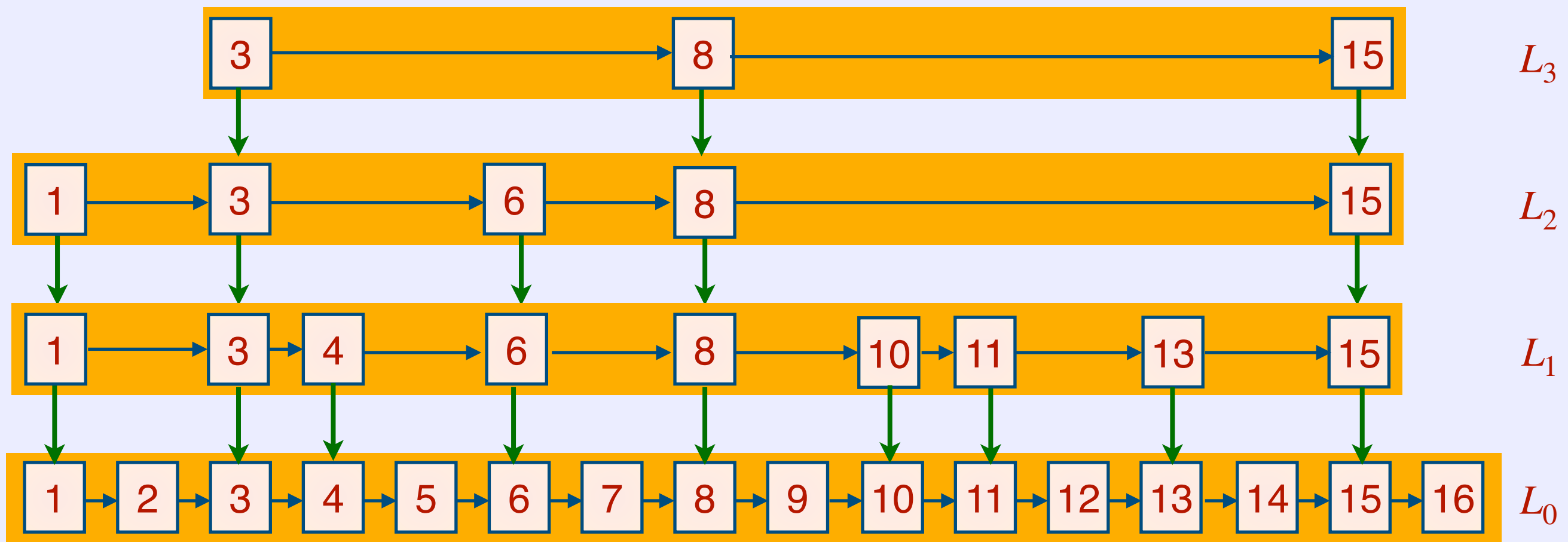
Each element of L_i is chosen
in L_{i+1} with prob $1/2$.



Skip Lists

Generation Rule

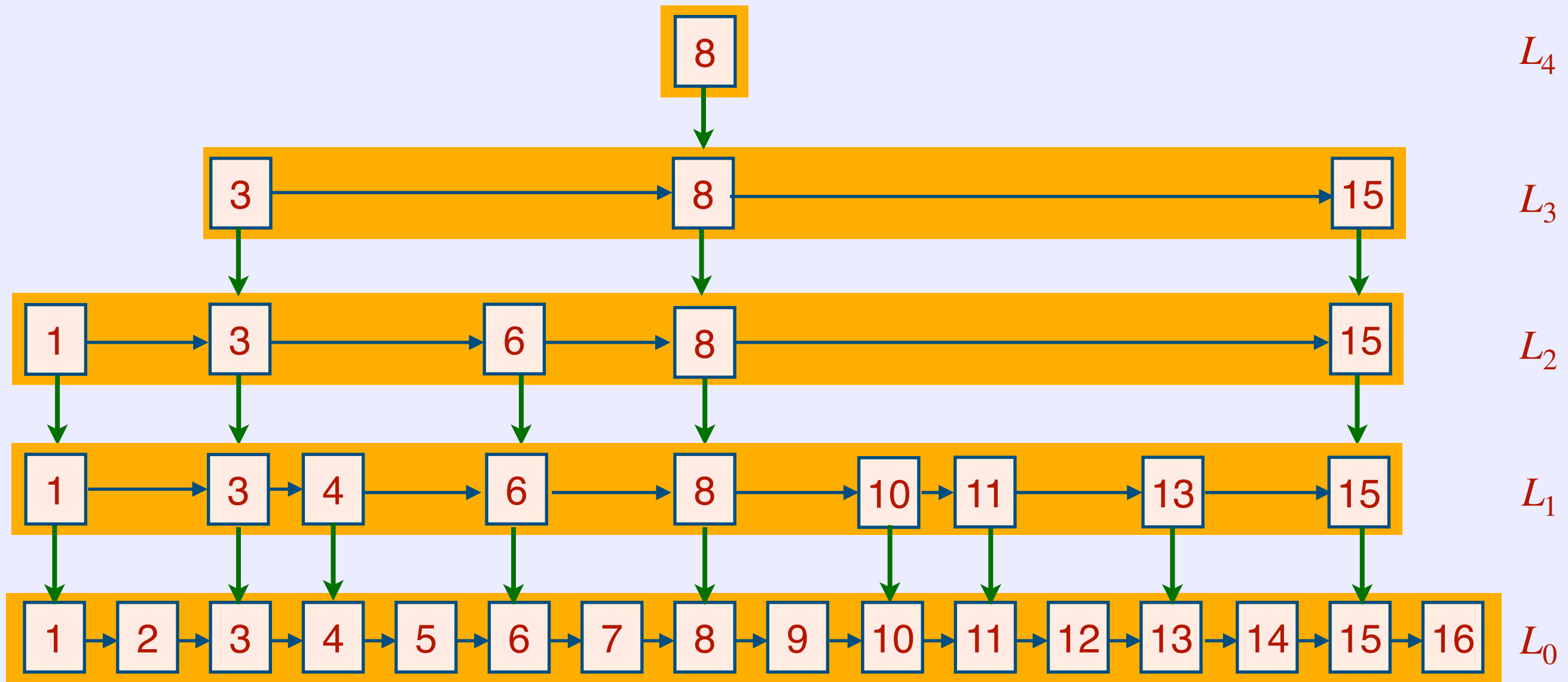
Each element of L_i is chosen
in L_{i+1} with prob $1/2$.



Skip Lists

Generation Rule

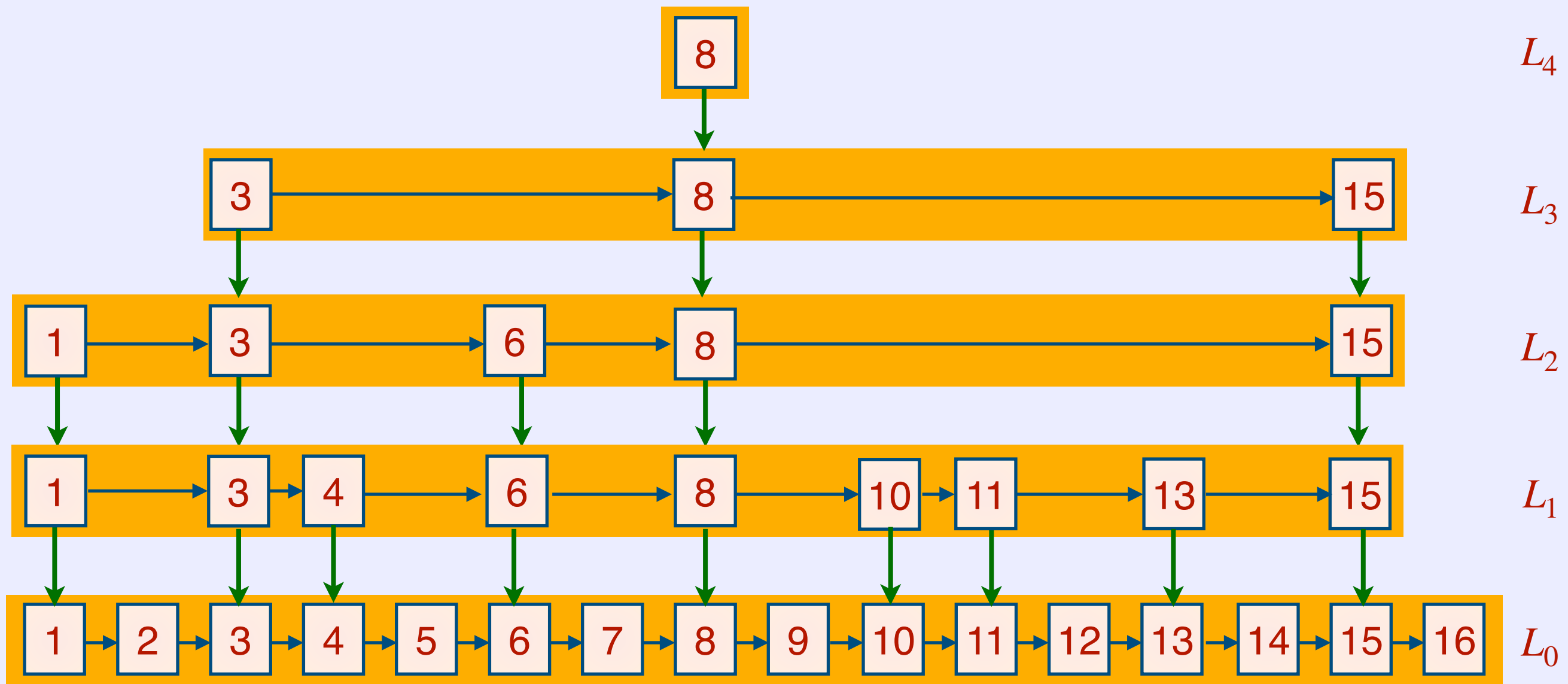
Each element of L_i is chosen in L_{i+1} with prob $1/2$.



Skip Lists

Generation Rule

Each element of L_i is chosen in L_{i+1} with prob $1/2$.



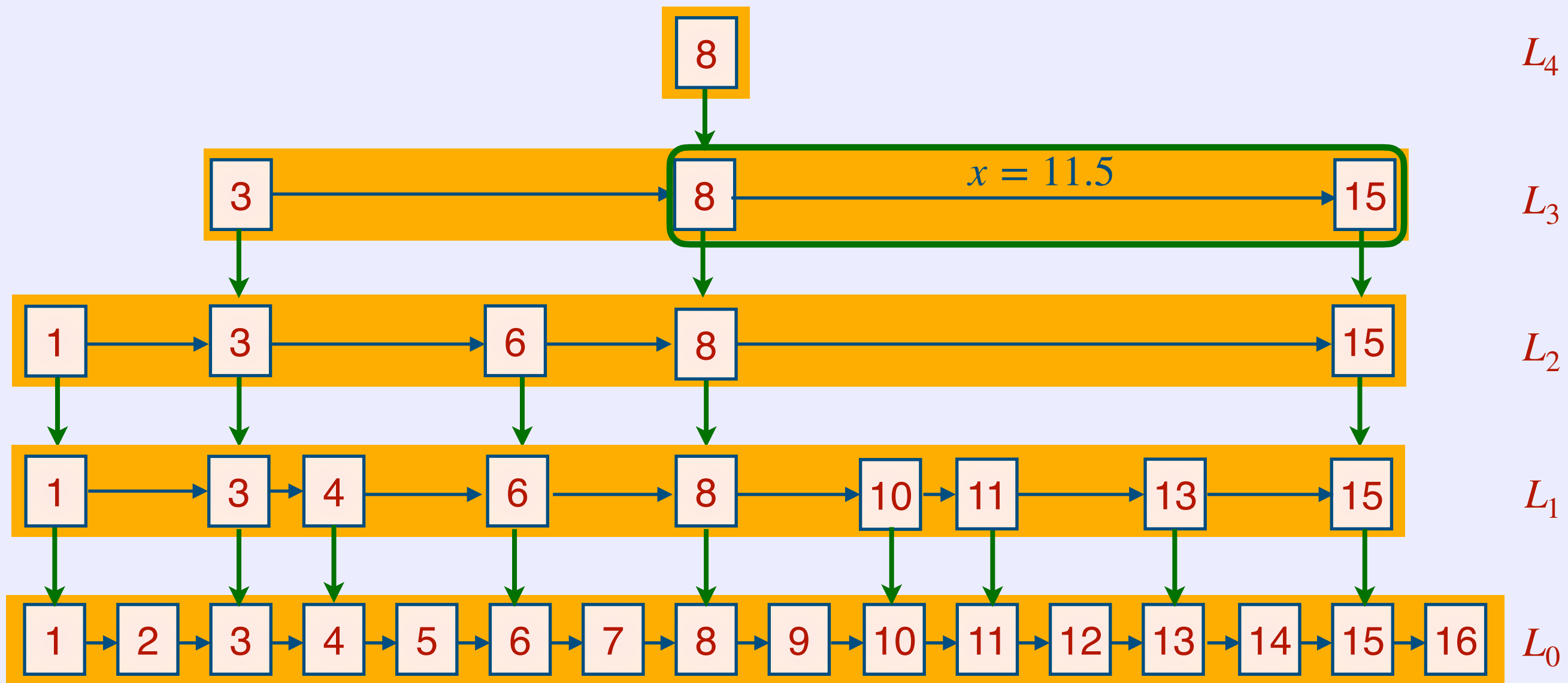
Search(x, L_i):

Search x in L_i , if not found then find Predecessor and Successor of x in L_i .

Skip Lists

Generation Rule

Each element of L_i is chosen in L_{i+1} with prob $1/2$.



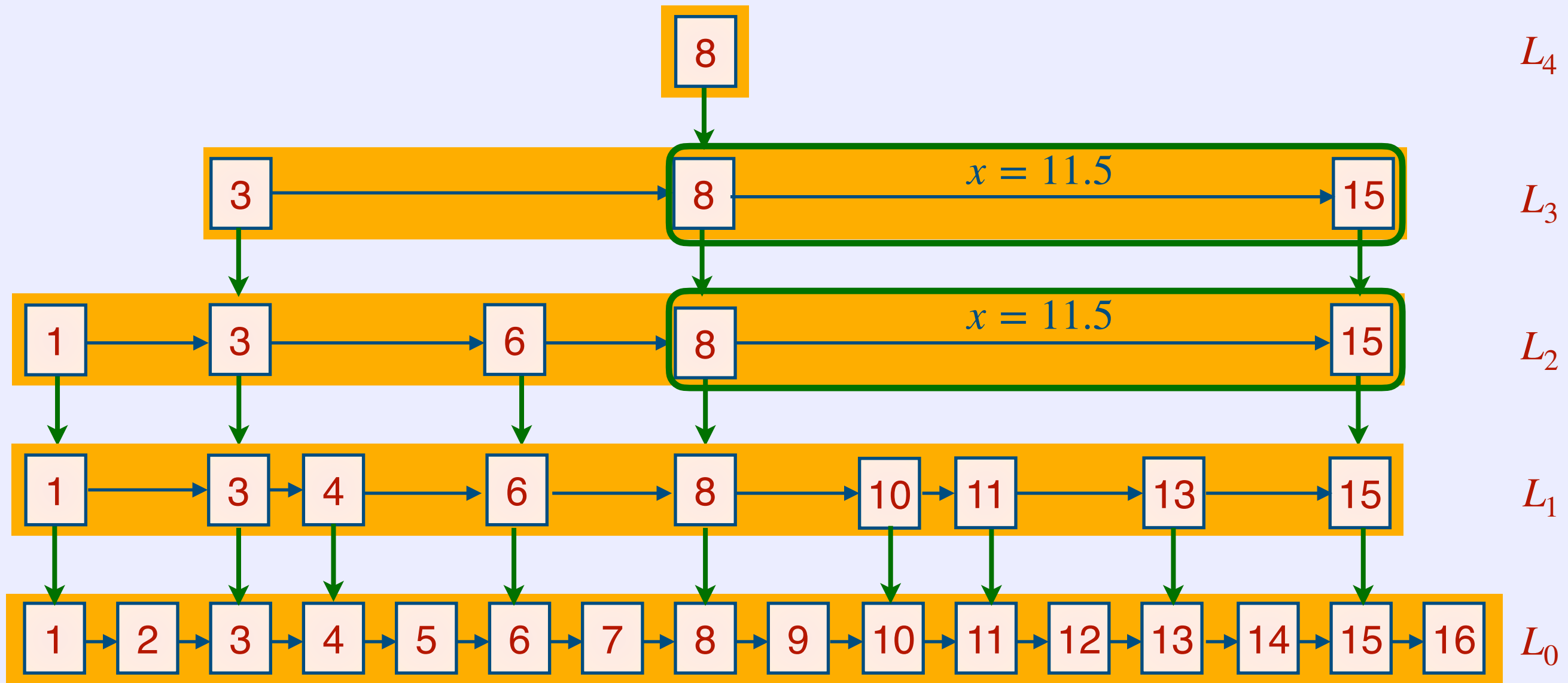
Search(x , L_i):

Search x in L_i , if not found then find Predecessor and Successor of x in L_i .

Skip Lists

Generation Rule

Each element of L_i is chosen in L_{i+1} with prob $1/2$.



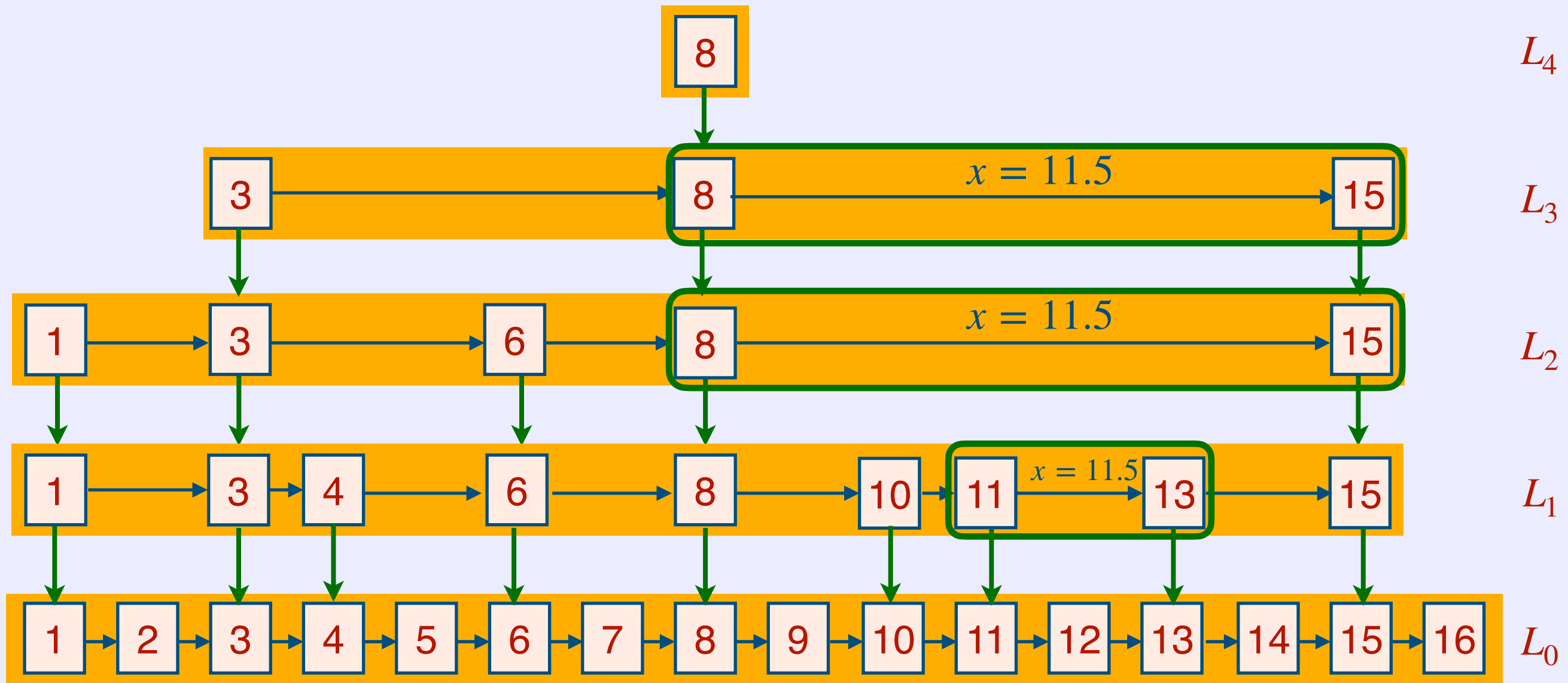
Search(x, L_i):

Search x in L_i , if not found then find Predecessor and Successor of x in L_i .

Skip Lists

Generation Rule

Each element of L_i is chosen in L_{i+1} with prob $1/2$.



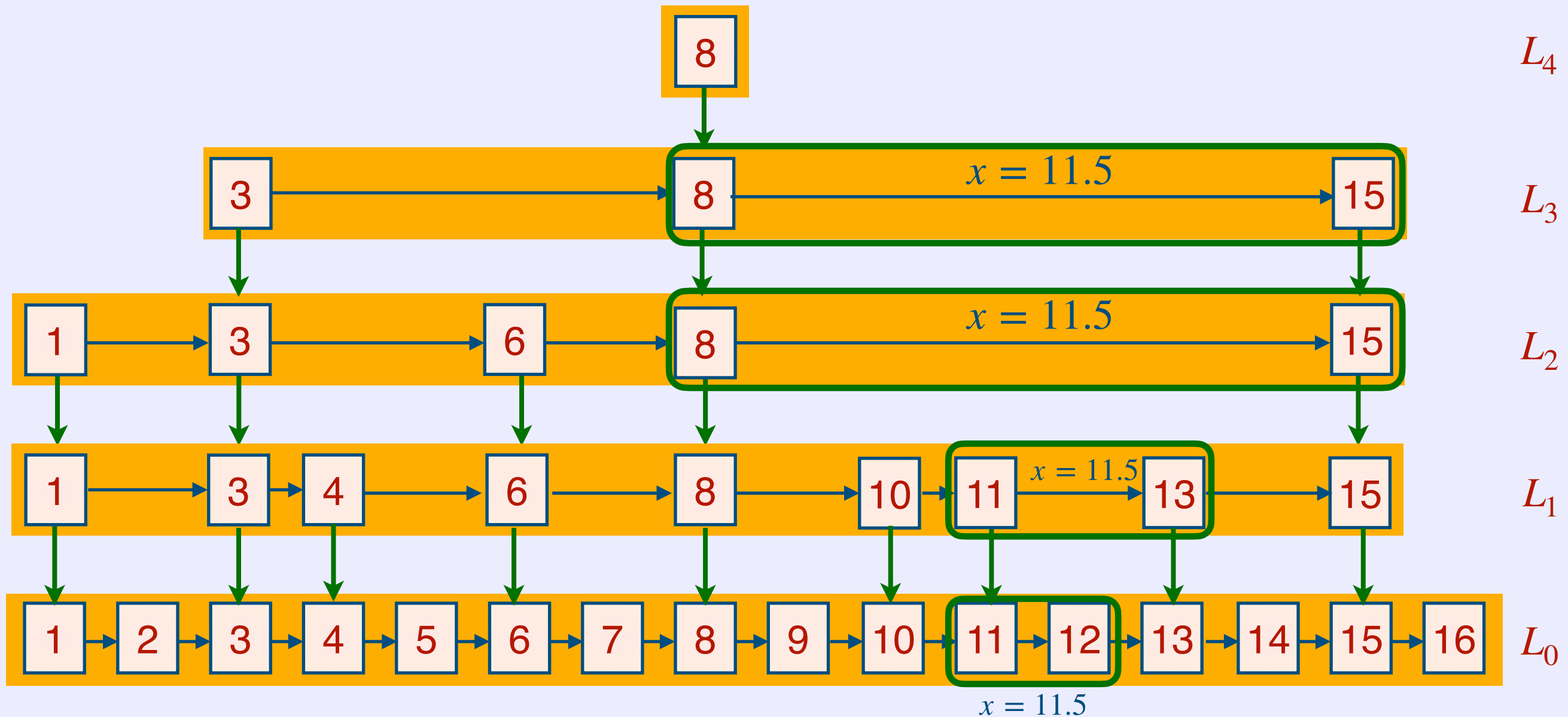
Search(x, L_i):

Search x in L_i , if not found then find Predecessor and Successor of x in L_i .

Skip Lists

Generation Rule

Each element of L_i is chosen in L_{i+1} with prob $1/2$.



Search(x, L_i):

Search x in L_i , if not found then find Predecessor and Successor of x in L_i .

Space Complexity

Consider an element $x \in L_0$,
and an index $i \geq 0$.

Probability that x is present in
just the lists $L_0, L_1, \dots, L_i$ is:

$$\frac{1}{2^{i+1}}$$

So, expected no. of copies of x is:

$$\sum_{i \geq 0} (i + 1) \cdot \frac{1}{2^{i+1}} = 2$$

Let $x_1, \dots, x_n$ be elements of L_0

For any i , the expected number of
copies of x_i is $O(1)$.

So, the space complexity is $O(n)$
on expectation.

Theorem: The expected space-complexity of Skip Lists data-structure is $O(n)$.

Number of levels in Skip List

Consider an element $x \in L_0$

Let $K = 5 \log_2 n$

Probability that x is present in list $L_1, L_2, \dots, L_K$ is:

$$\frac{1}{2^K} = \frac{1}{n^5}$$

Let $x_1, \dots, x_n$ be elements of L_0

For any $i \in [1, n]$,

$$\text{Prob}(x_i \in L_K) = \frac{1}{n^5}$$

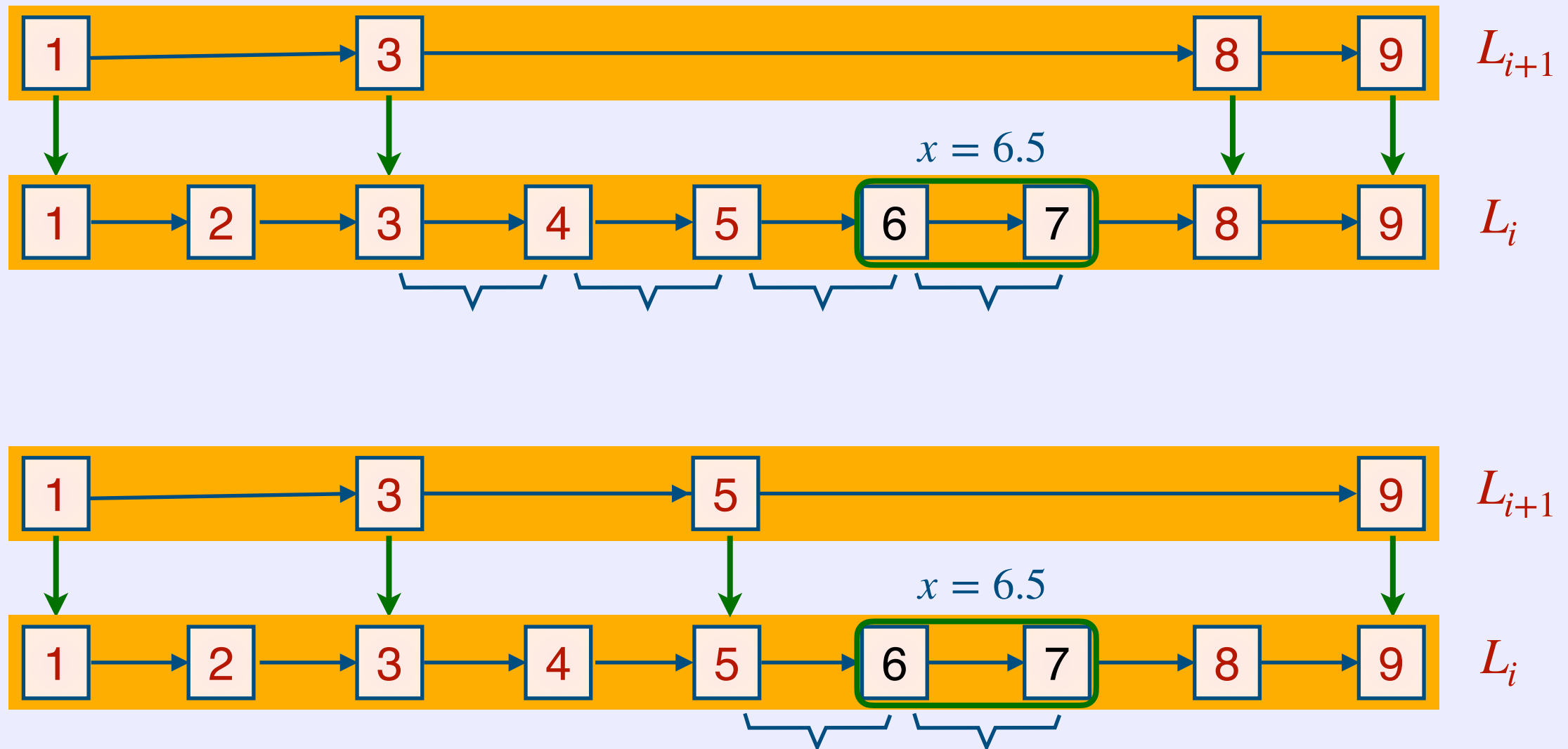
By union bound, probability that there exists an x_i in level L_K is **at most**:

$$n \cdot \frac{1}{n^5} = \frac{1}{n^4}$$

Lemma 1: If $n \geq 10$, then probability that the number of levels is more than $5 \log_2 n$ is at most $(1 - 1/n^4)$.

Search in Skip List at level 'i'

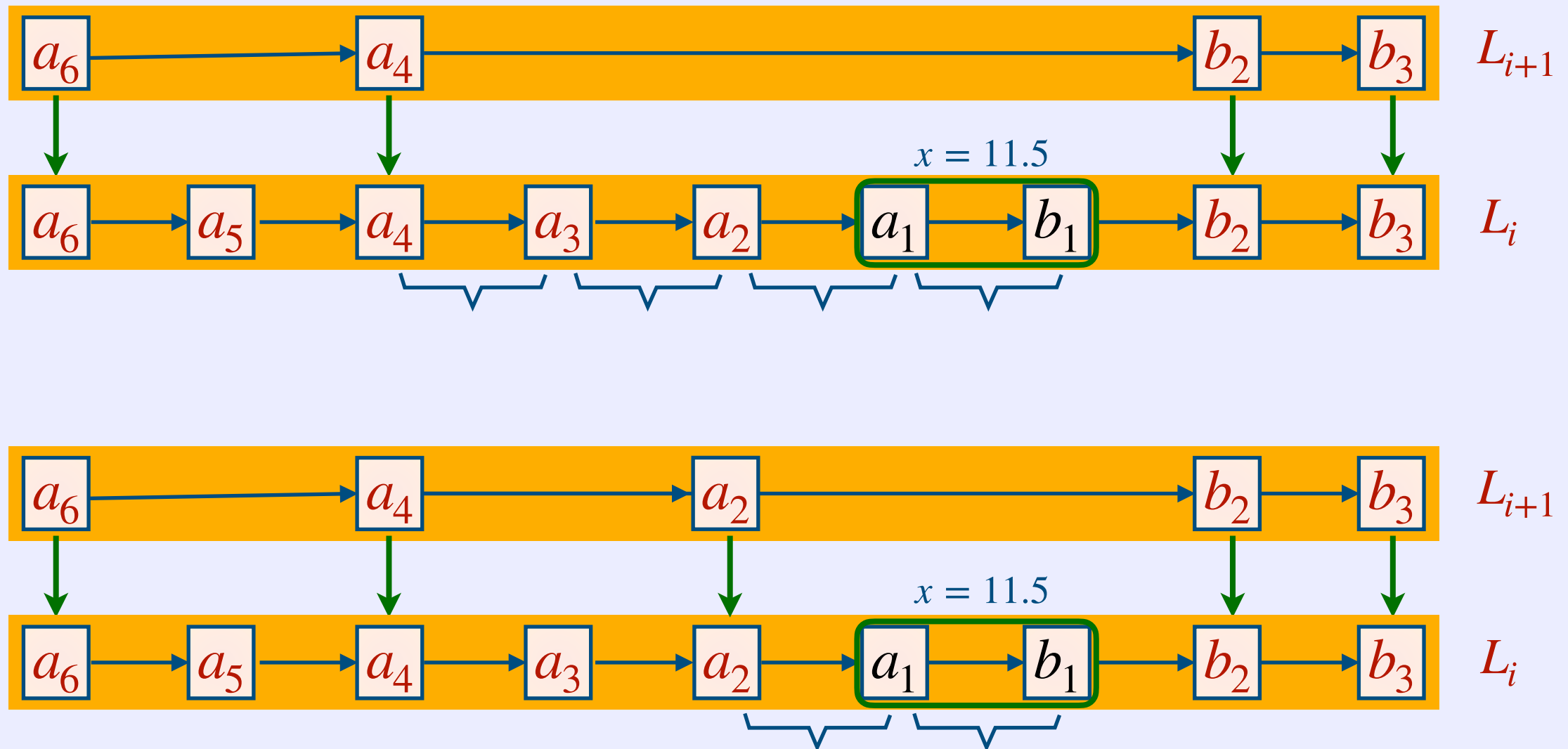
Consider two cases wherein list L_i is identical, but L_{i+1} is different.



Note: For any $x \in [6, 7)$, $\text{Search}(x, L_i)$ will take **lesser time** in the second case.

Search in Skip List at level 'i'

Consider two cases wherein list L_i is identical, but L_{i+1} is different.



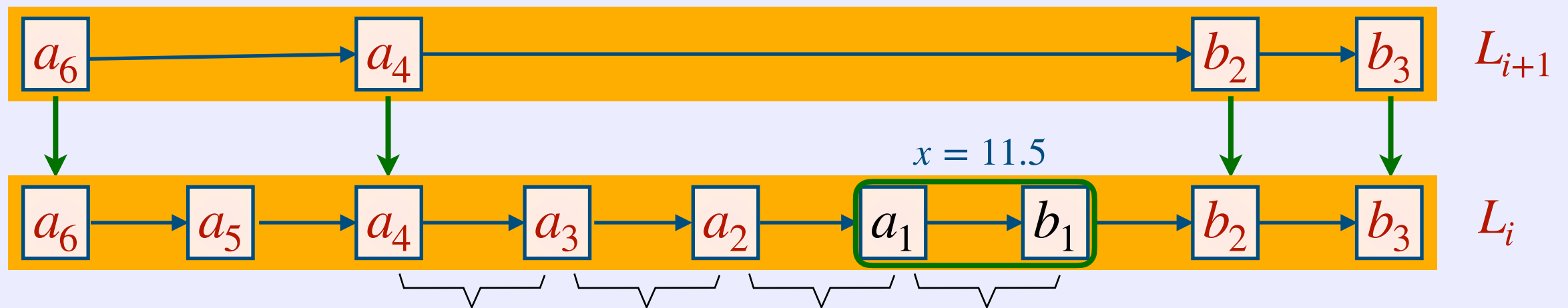
Note: For any $x_i \in [a_1, b_1)$, $\text{Search}(x, L_i)$ will take **lesser time** in the second case.

Search in Skip List at level 'i'

Lemma 2:

Suppose $L_i = (a_r, \dots, a_2, a_1, b_1, b_2, \dots, b_s)$, and let x be a query element in $[a_1, b_1]$.

If j is **smallest** index such that a_j lies in L_{i+1} , then time taken by $\text{Search}(x, L_i)$ is $O(j)$.

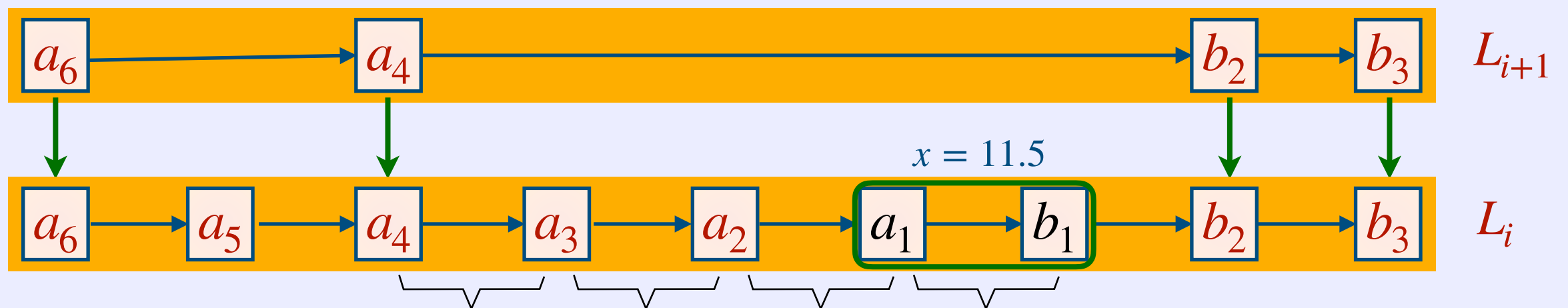


Search in Skip List at level 'i'

Lemma 2:

Suppose $L_i = (a_r, \dots, a_2, a_1, b_1, b_2, \dots, b_s)$, and let x be a query element in $[a_1, b_1]$.

If j is **smallest** index such that a_j lies in L_{i+1} , then time taken by $\text{Search}(x, L_i)$ is $O(j)$.



Expected value of j:

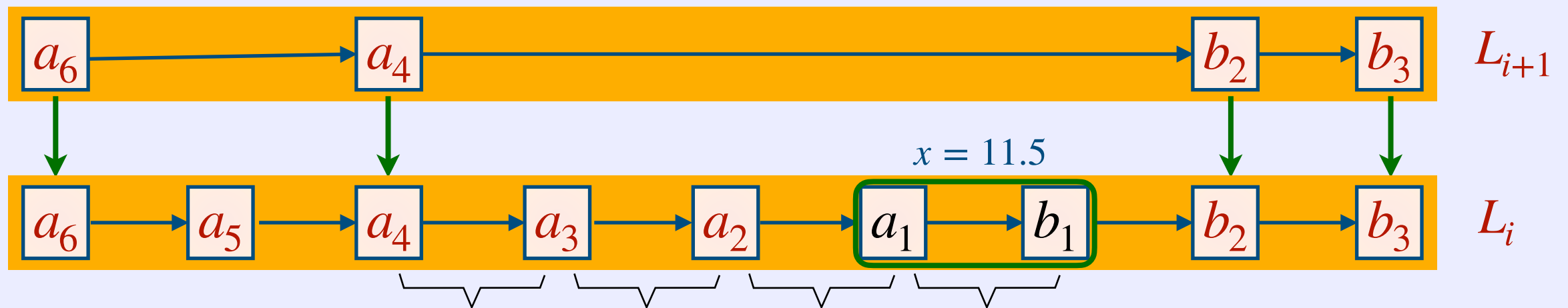
$$\sum_{t \geq 1} t \times \text{Prob}(t \text{ is smallest index such that } a_t \text{ lies in } L_{i+1})$$

Search in Skip List at level 'i'

Lemma 2:

Suppose $L_i = (a_r, \dots, a_2, a_1, b_1, b_2, \dots, b_s)$, and let x be a query element in $[a_1, b_1]$.

If j is **smallest** index such that a_j lies in L_{i+1} , then time taken by $\text{Search}(x, L_i)$ is $O(j)$.



Expected value of j:

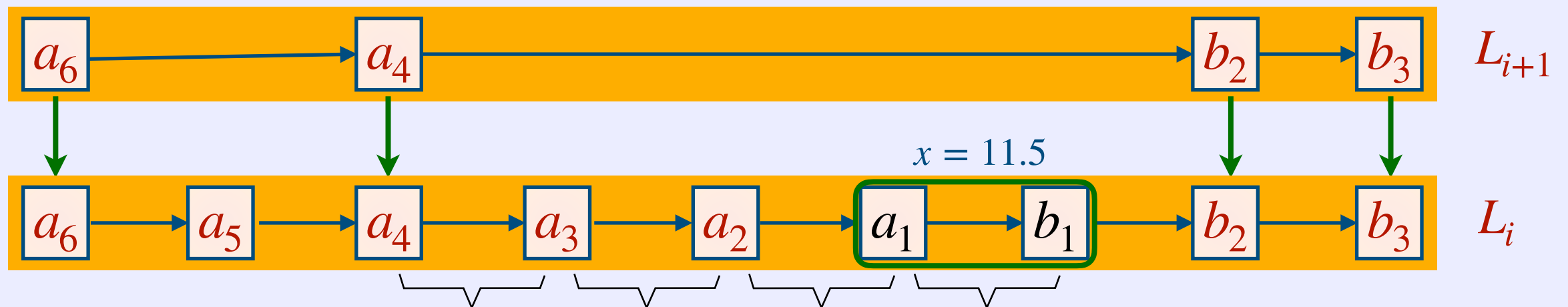
$$\sum_{t \geq 1} t \times \text{Prob}(t \text{ is smallest index such that } a_t \text{ lies in } L_{i+1}) = \sum_{t \geq 1} t \times \frac{1}{2^t}$$

Search in Skip List at level 'i'

Lemma 2:

Suppose $L_i = (a_r, \dots, a_2, a_1, b_1, b_2, \dots, b_s)$, and let x be a query element in $[a_1, b_1]$.

If j is **smallest** index such that a_j lies in L_{i+1} , then time taken by $\text{Search}(x, L_i)$ is $O(j)$.



Expected value of j:

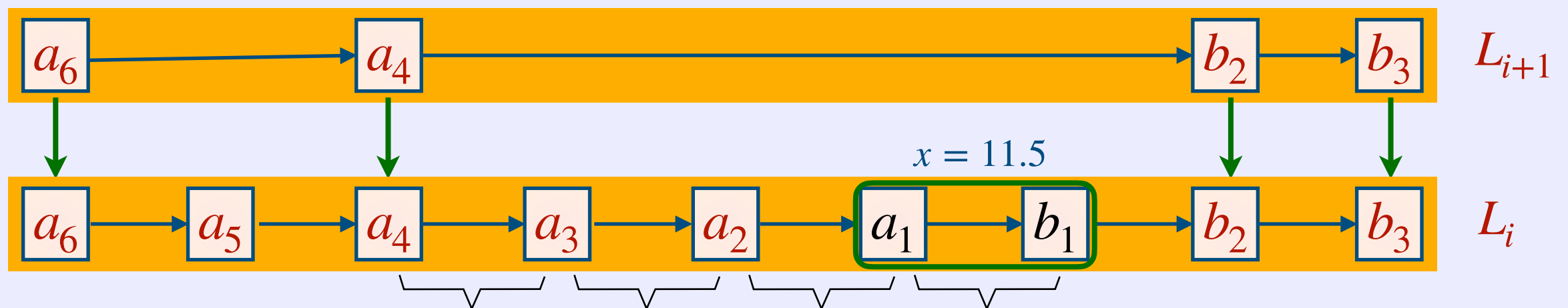
$$\sum_{t \geq 1} t \times \text{Prob}(t \text{ is smallest index such that } a_t \text{ lies in } L_{i+1}) = \sum_{t \geq 1} t \times \frac{1}{2^t} = 2$$

Search in Skip List at level 'i'

Lemma 2:

Suppose $L_i = (a_r, \dots, a_2, a_1, b_1, b_2, \dots, b_s)$, and let x be a query element in $[a_1, b_1]$.

If j is **smallest** index such that a_j lies in L_{i+1} , then time taken by $\text{Search}(x, L_i)$ is $O(j)$.



Lemma 3:

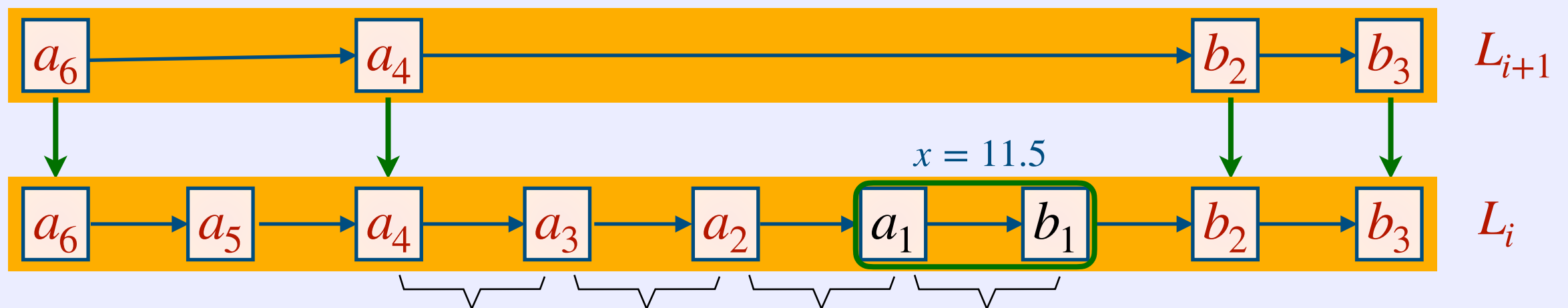
The expected time to search x in L_i using the output of $\text{Search}(x, L_{i+1})$ is $O(1)$.

Search in Skip List at level 'i'

Lemma 2:

Suppose $L_i = (a_r, \dots, a_2, a_1, b_1, b_2, \dots, b_s)$, and let x be a query element in $[a_1, b_1]$.

If j is **smallest** index such that a_j lies in L_{i+1} , then time taken by $\text{Search}(x, L_i)$ is $O(j)$.



Lemma 3:

The expected time to search x in L_i using the output of $\text{Search}(x, L_{i+1})$ is $O(1)$.

Lemma 4:

The expected time to search x in ALL the lists is $O(\log n)$.

Summary

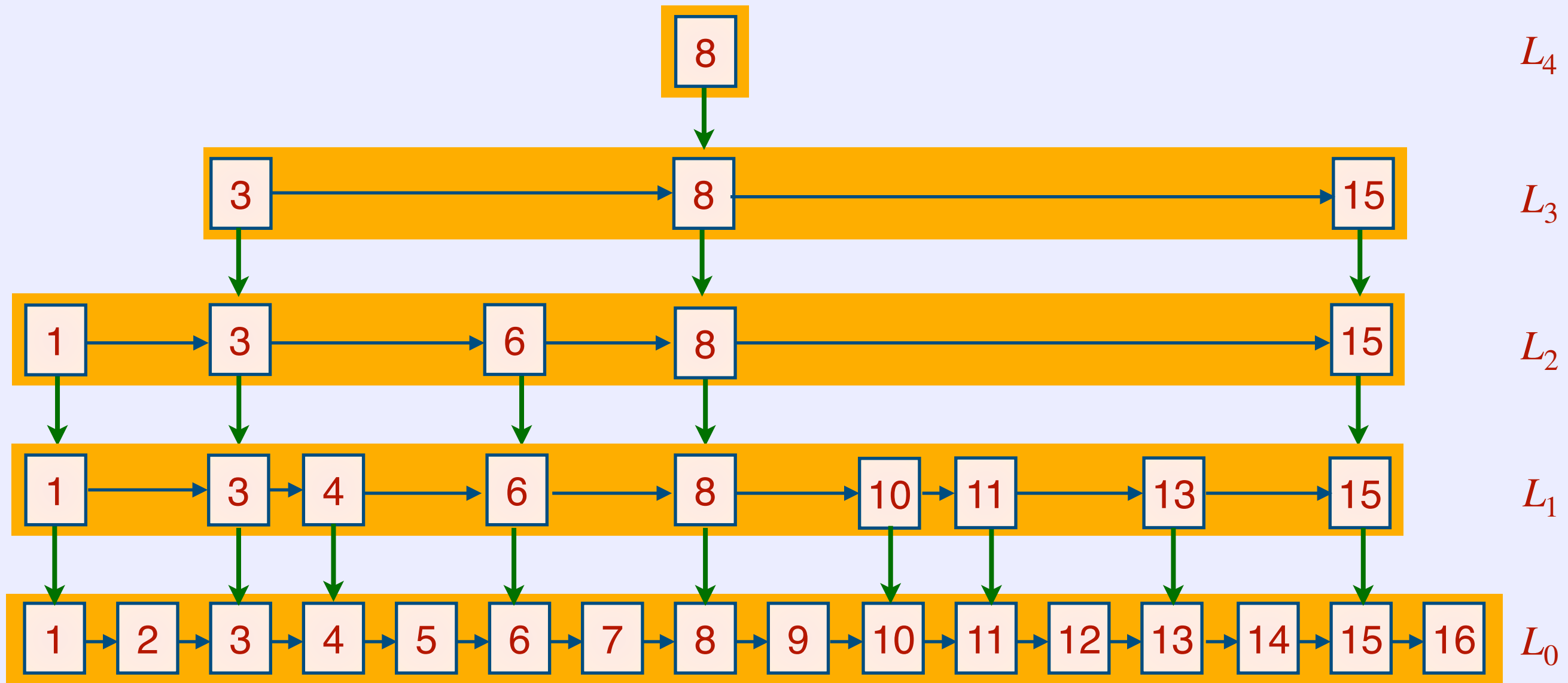
Skip Lists: A dictionary based on lists with following guarantees

OPERATION	EXPECTED TIME
 Search	$O(\log n)$
Deletion	$O(\log n)$
Insertion	$O(\log n)$

Skip Lists - Deletion

Example

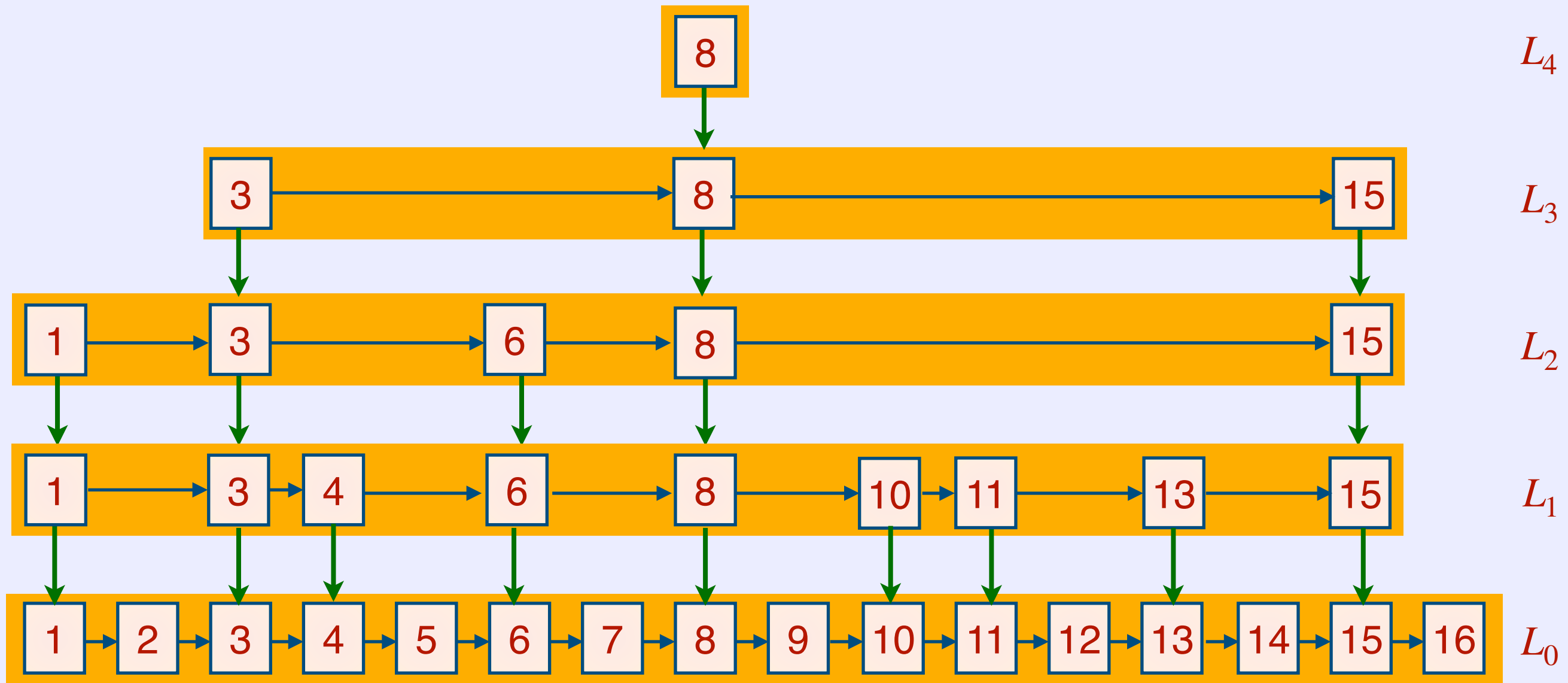
Deletion of element
 $x = 11$ from the skip list



Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list

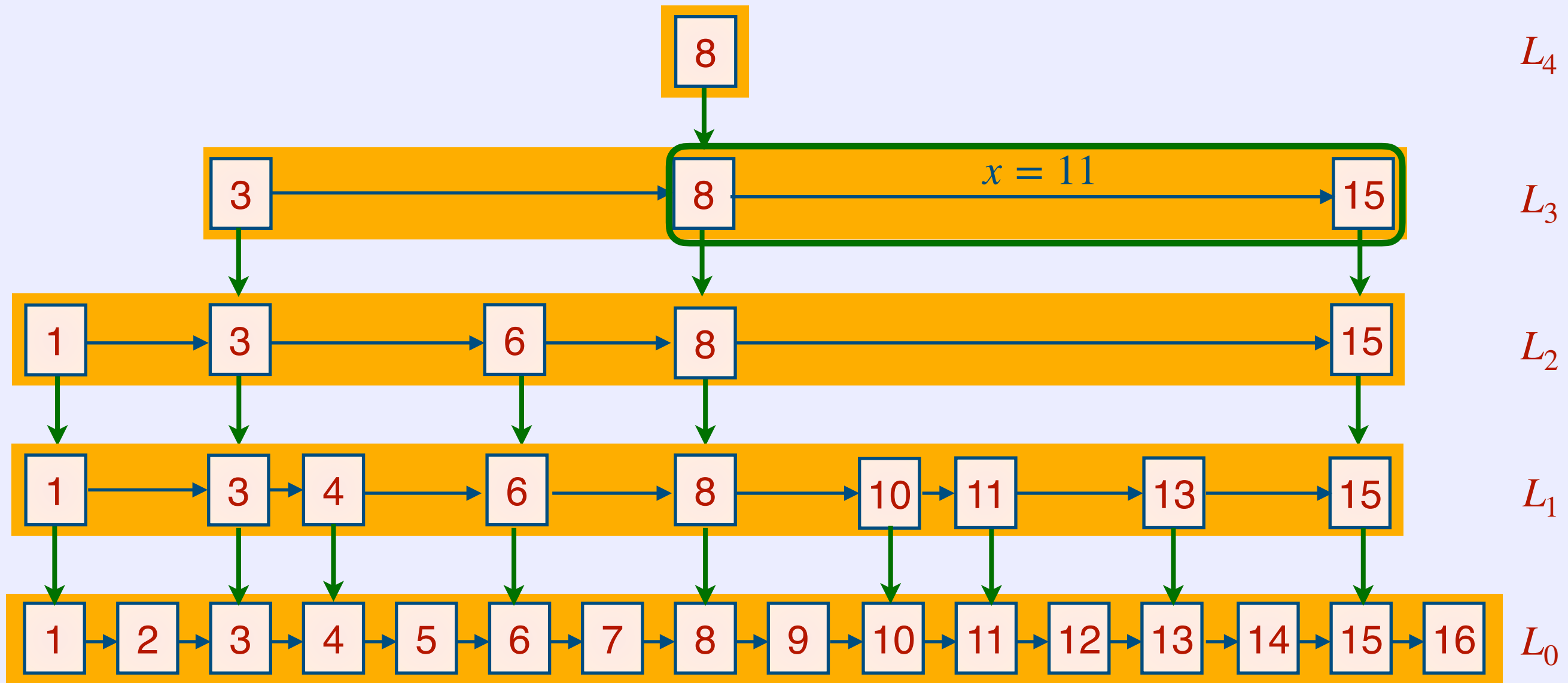


1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .

Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list

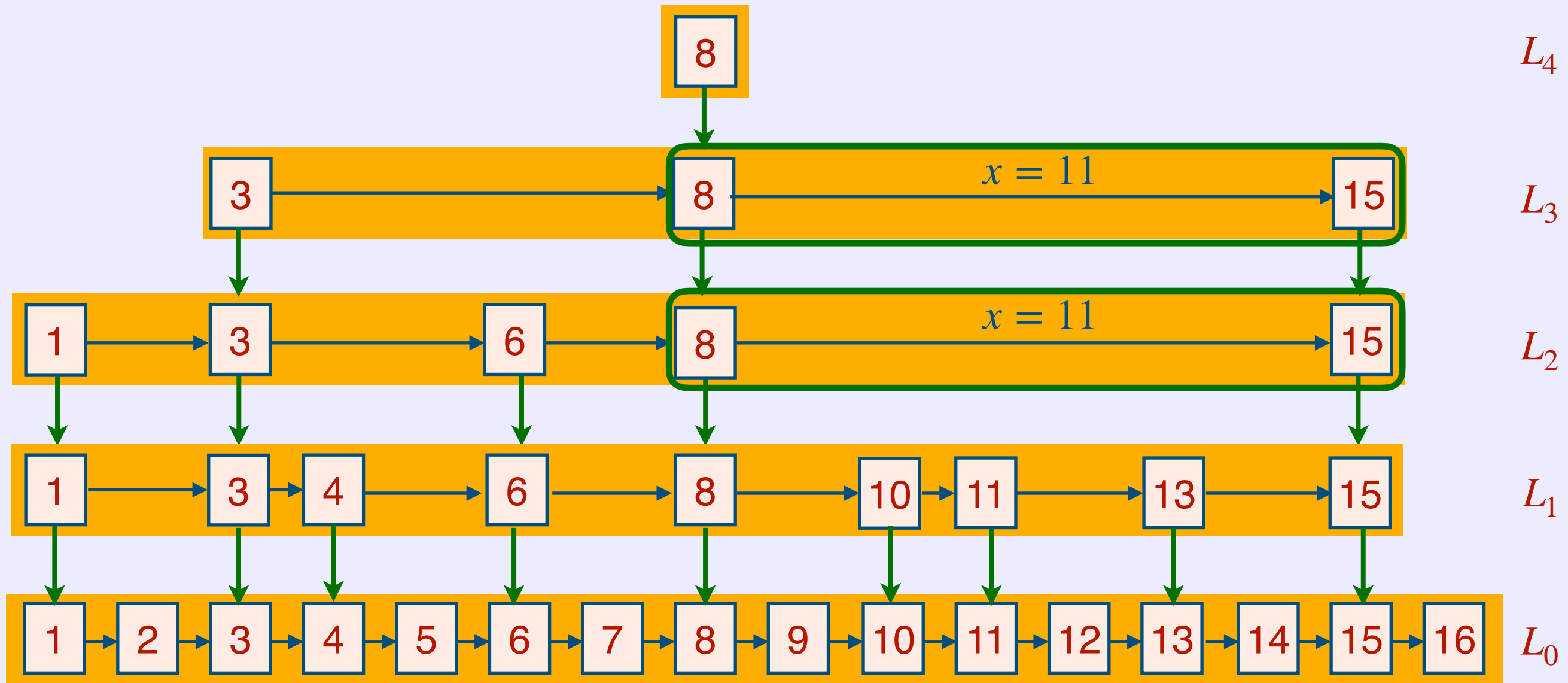


1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .

Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list

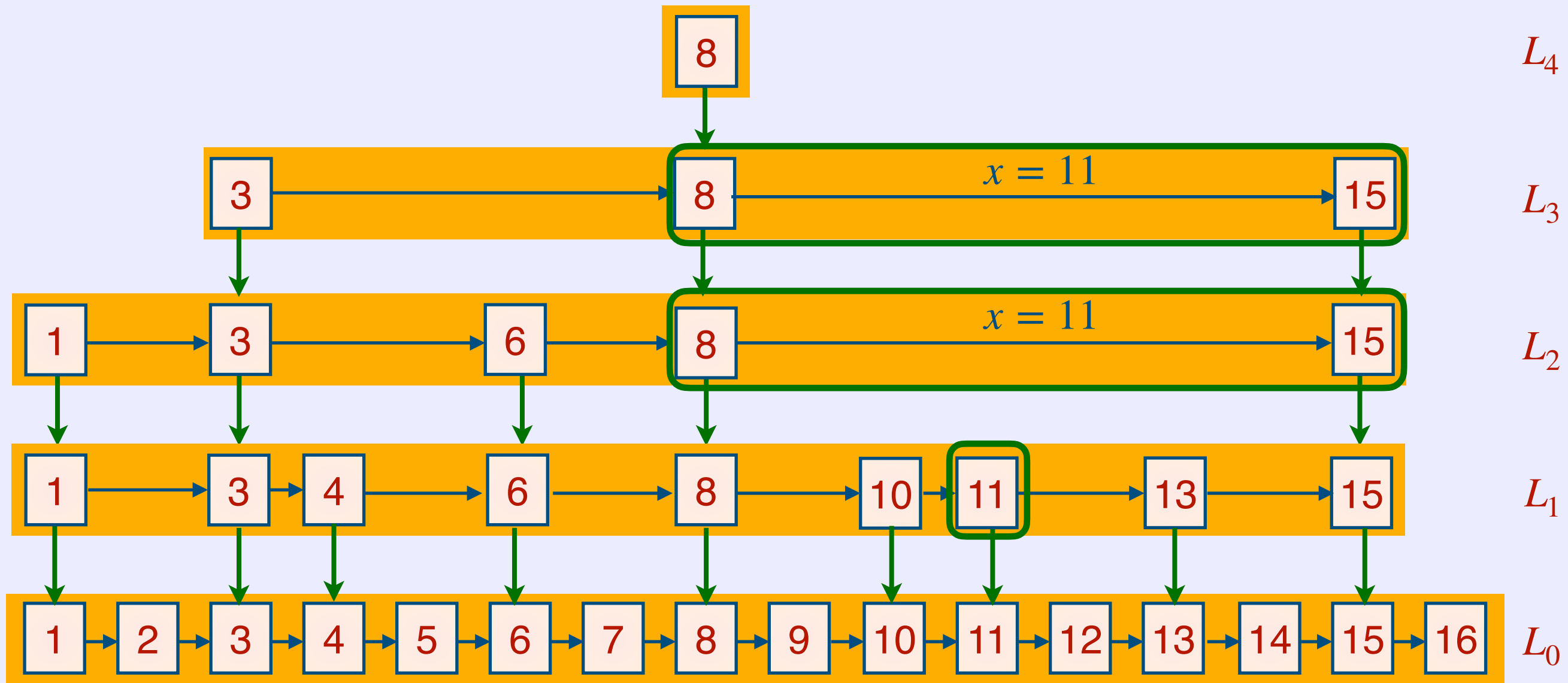


1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .

Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list

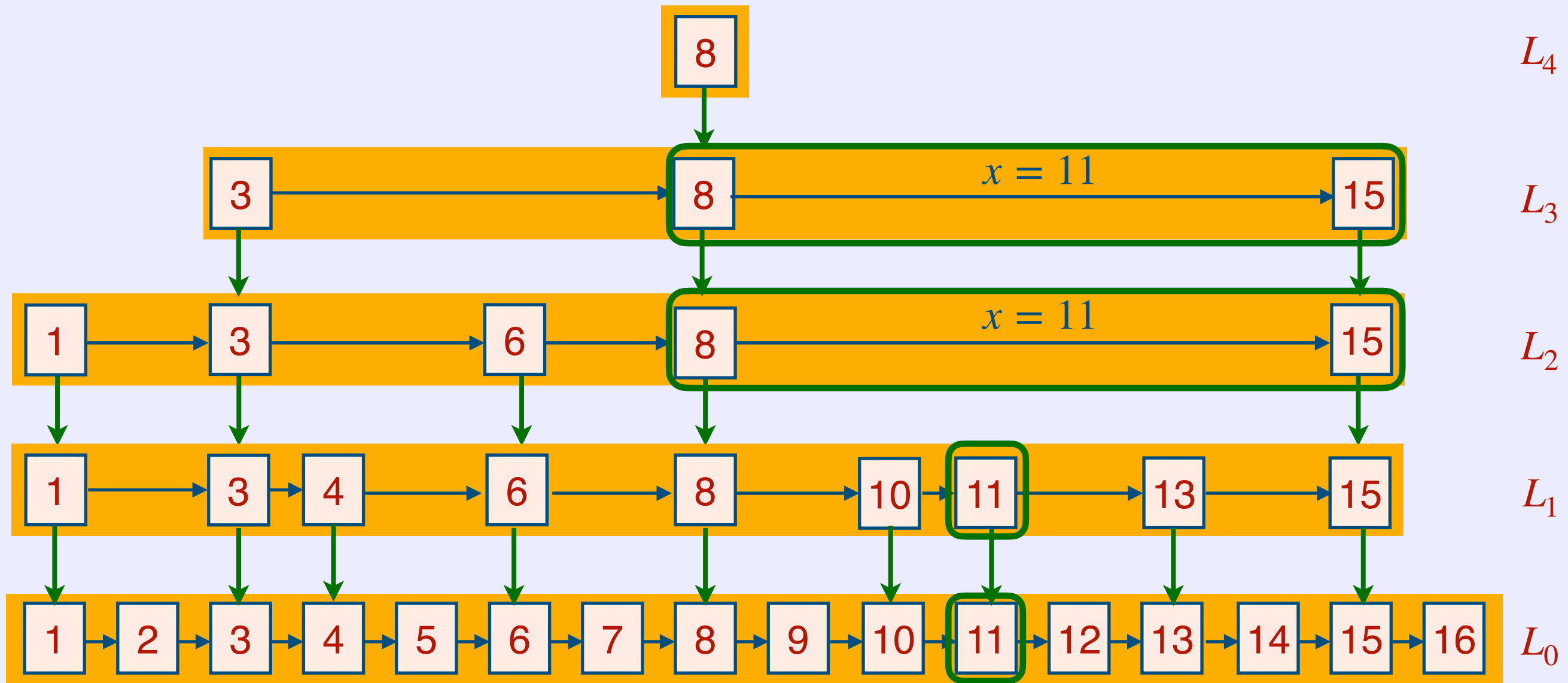


1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .

Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list

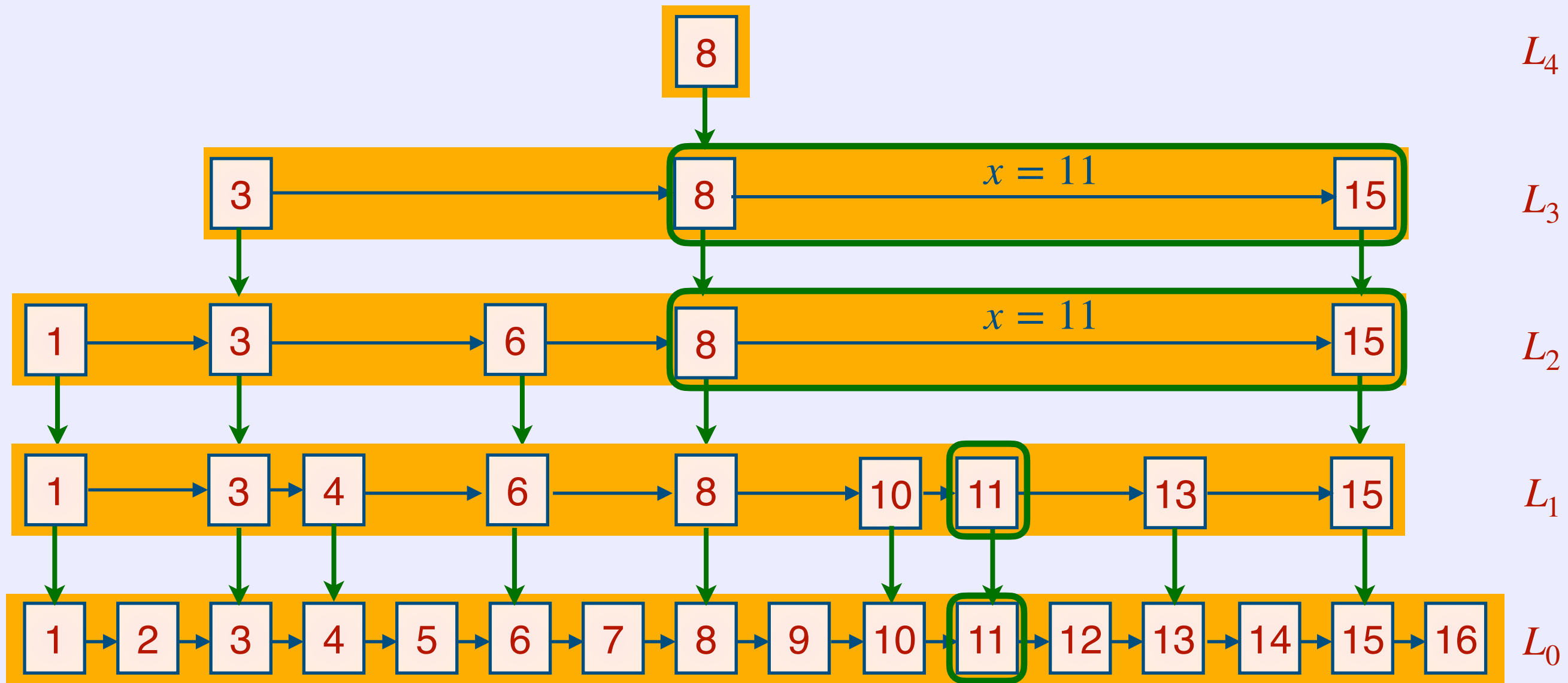


1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .

Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list

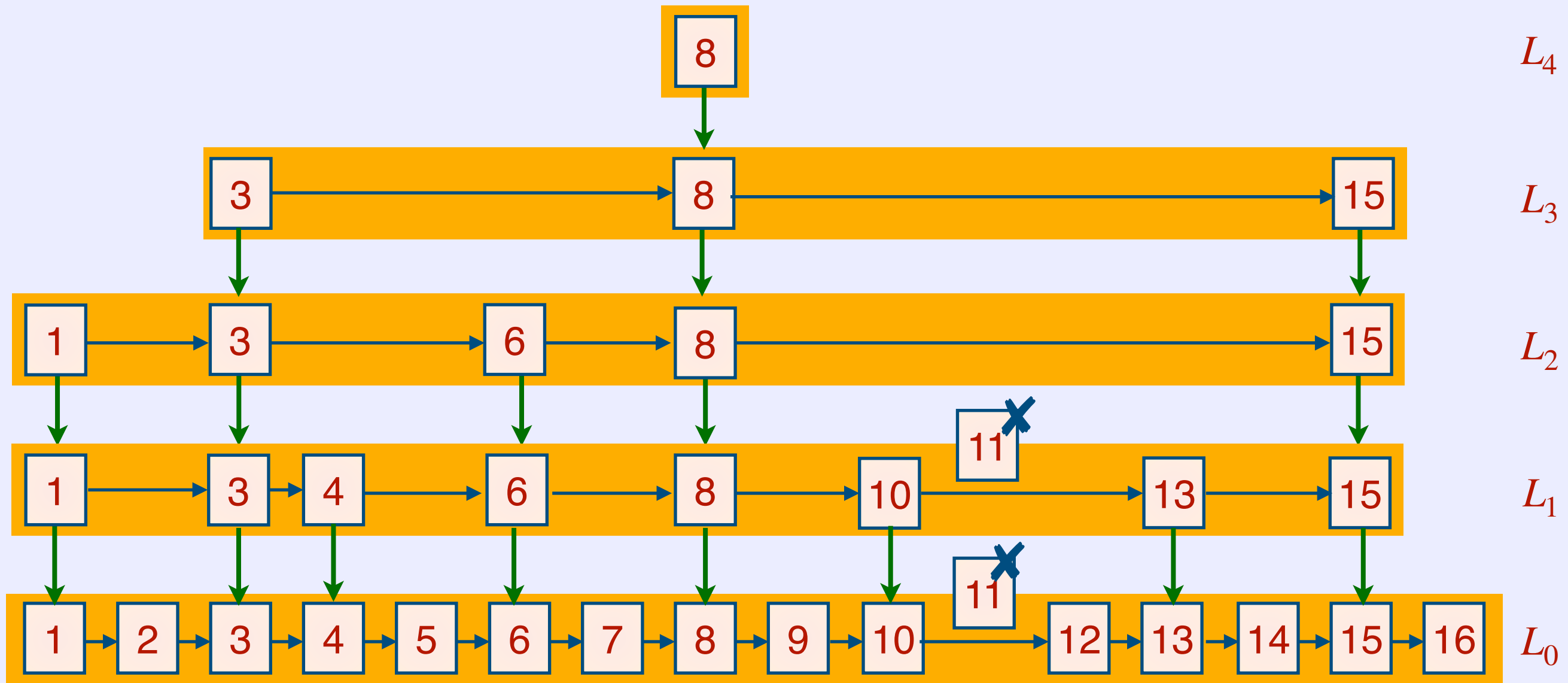


1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .
2. Remove x from the corresponding lists.

Skip Lists - Deletion

Example

Deletion of element
 $x = 11$ from the skip list



1. Perform $\text{Search}(x, L_i)$ to find ALL those lists L_i that contain element x .
2. Remove x from the corresponding lists.

Summary

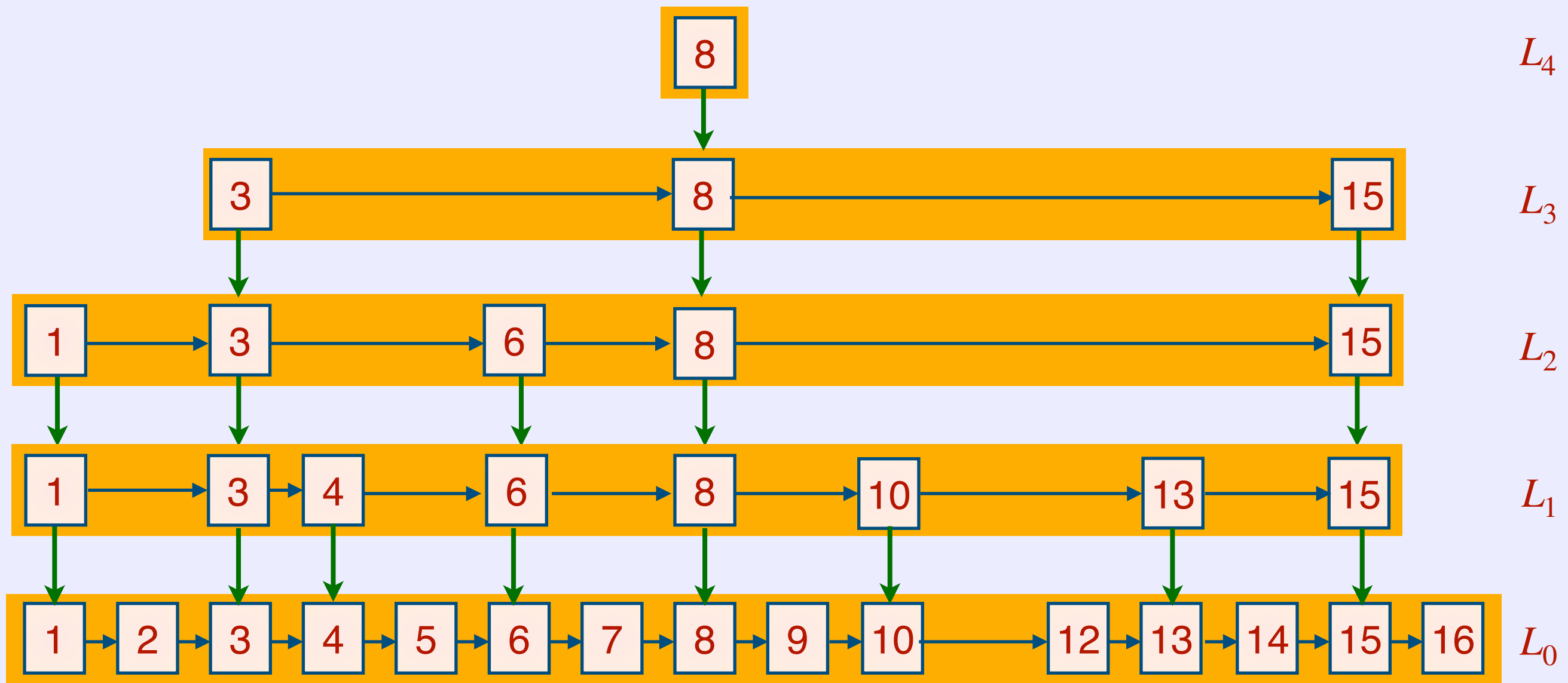
Skip Lists: A dictionary based on lists with following guarantees

OPERATION	EXPECTED TIME
✓ Search	$O(\log n)$
✓ Deletion	$O(\log n)$
Insertion	$O(\log n)$

Skip Lists - Insertion

Example

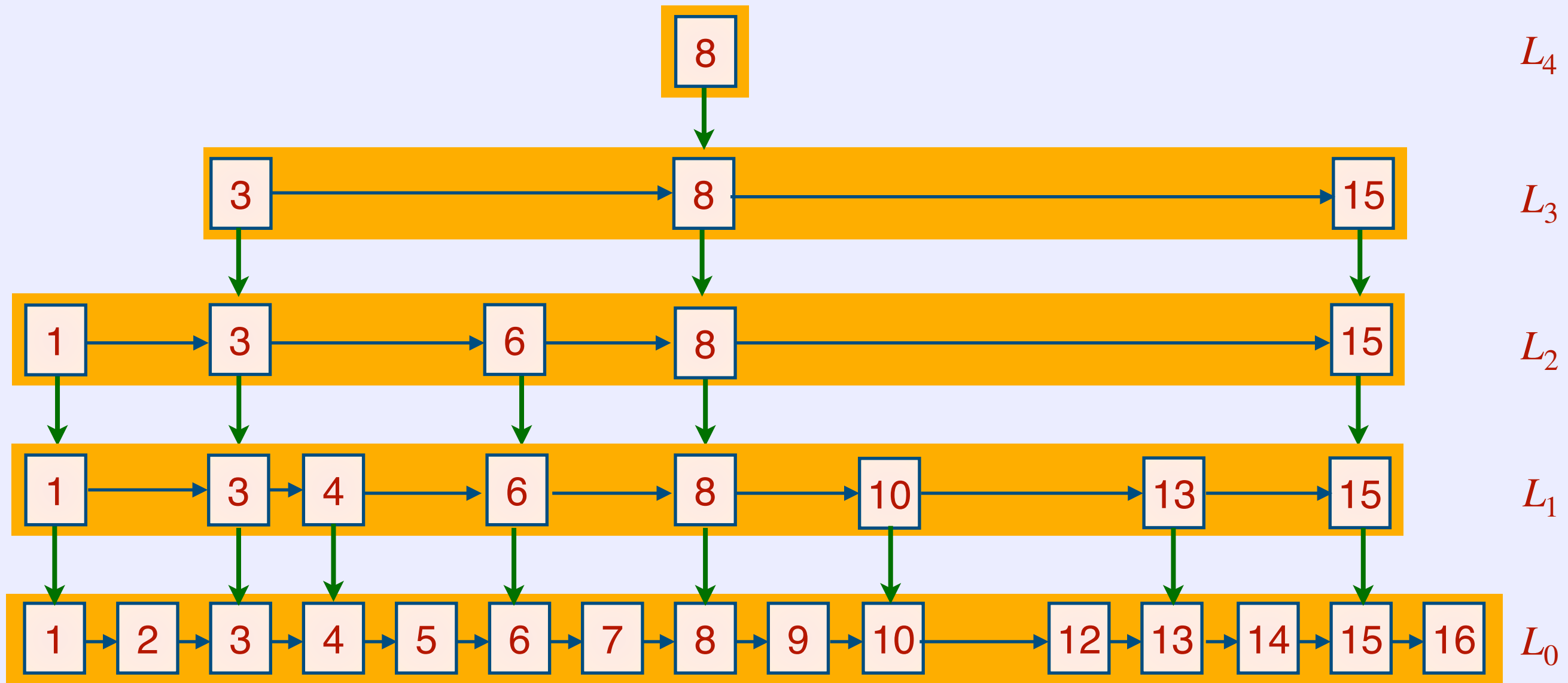
Insertion of element
 $x = 11$ in the skip list



Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list

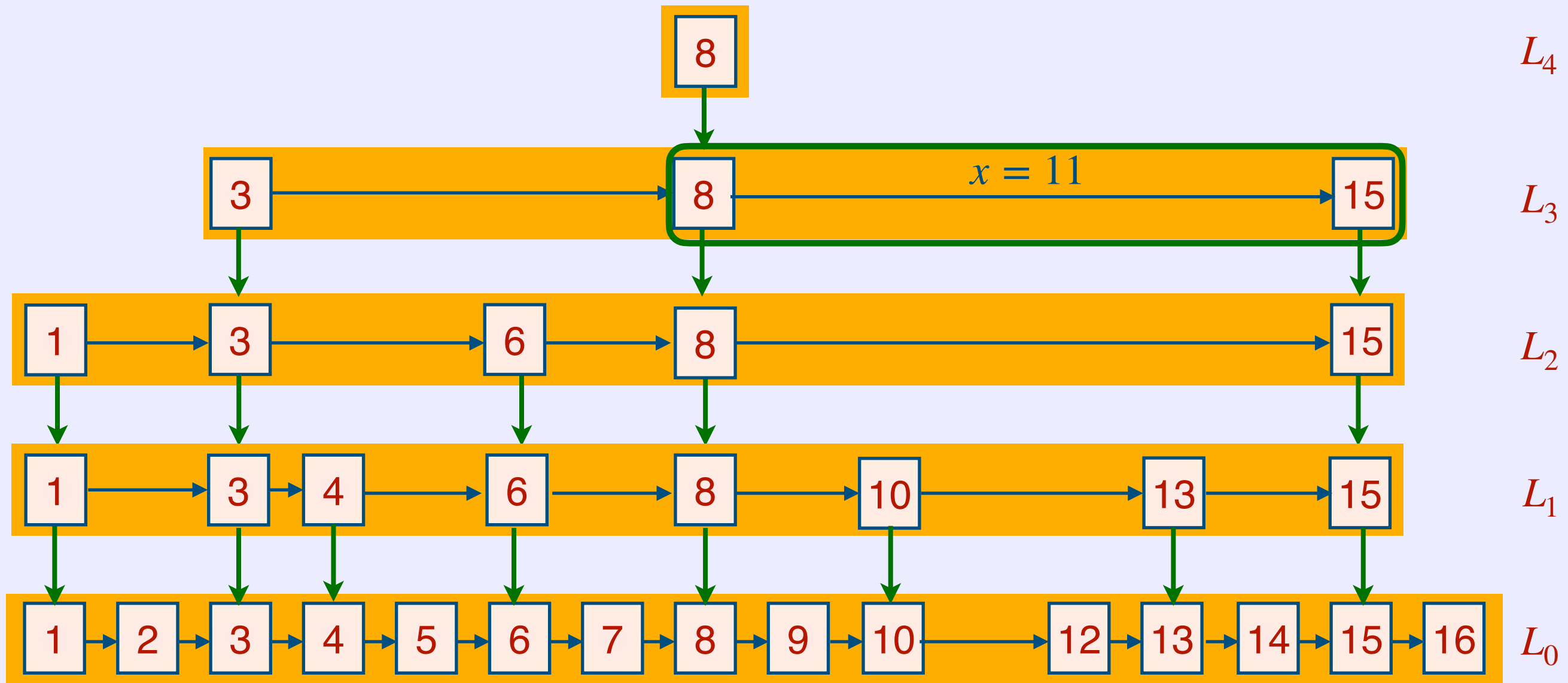


1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list

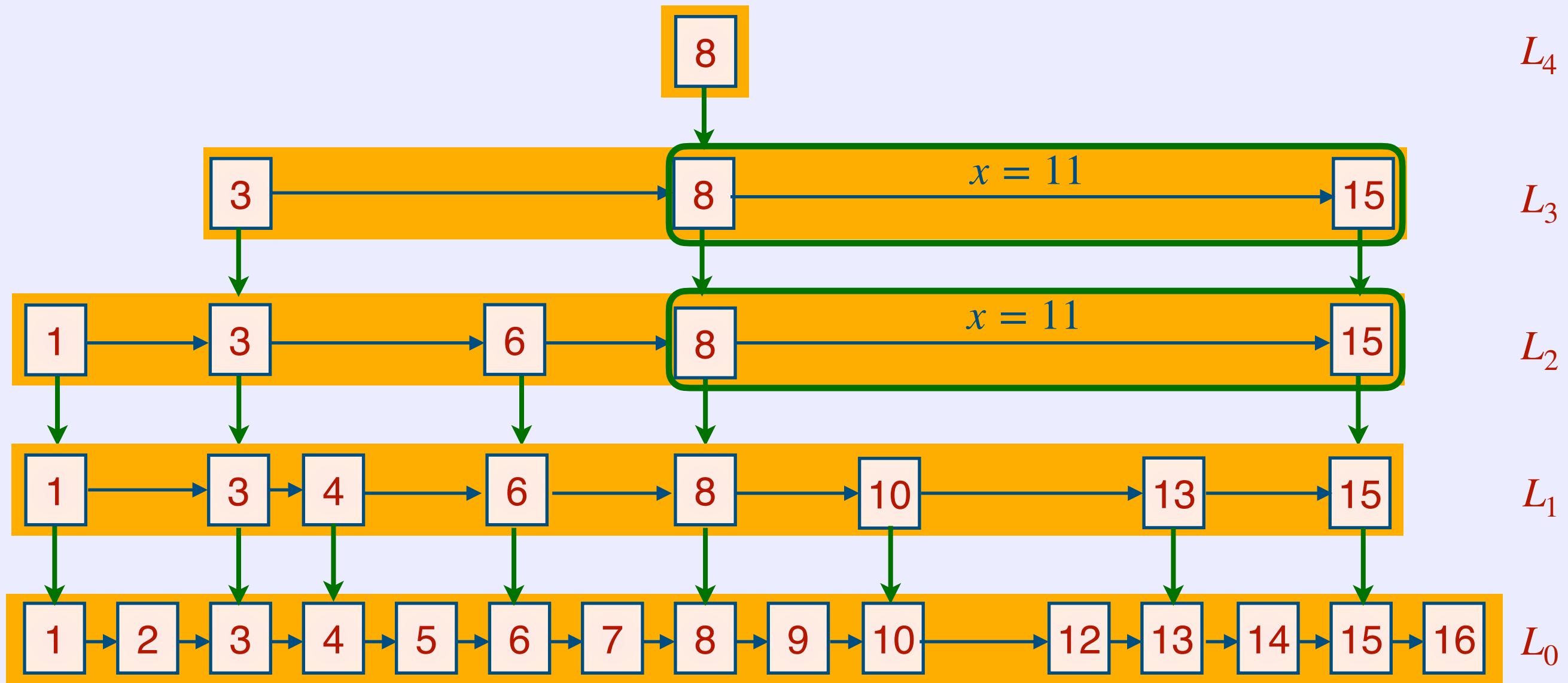


1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list

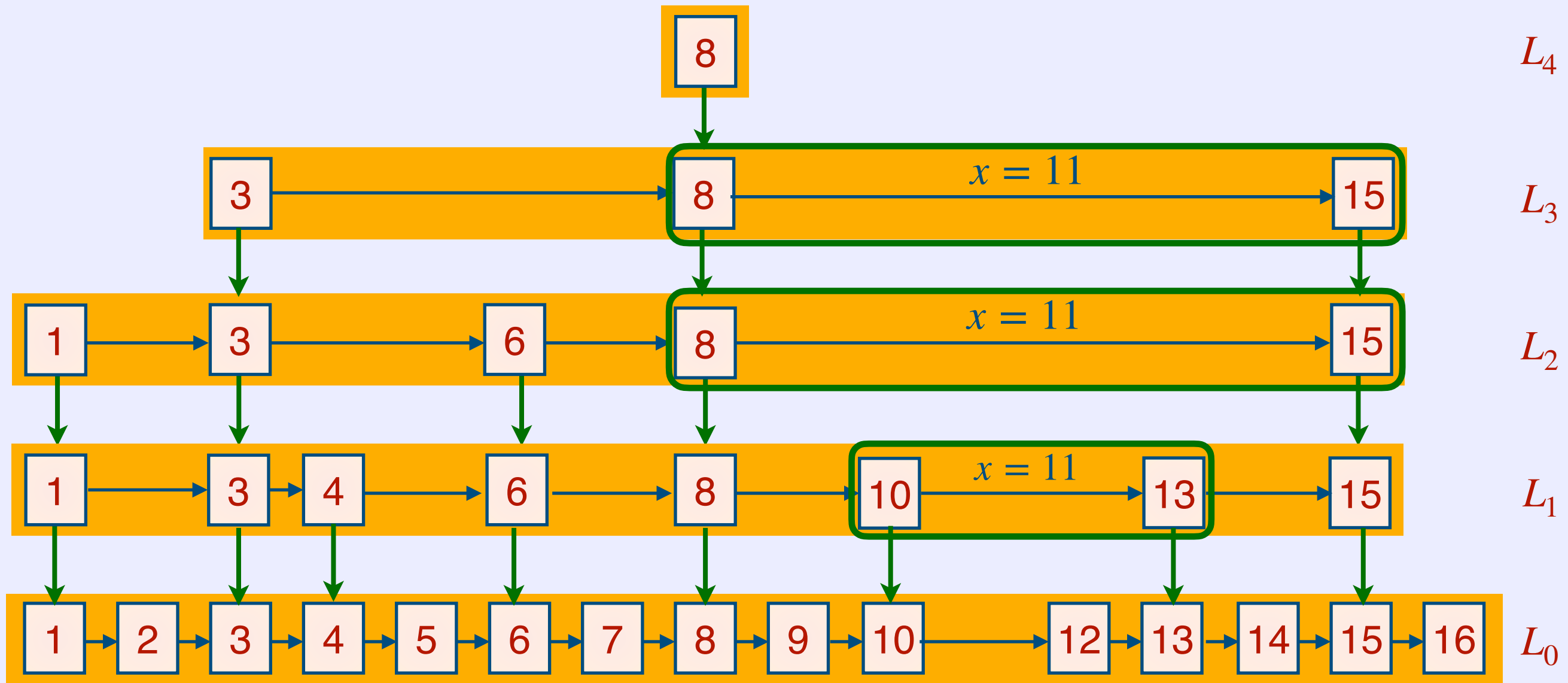


1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list

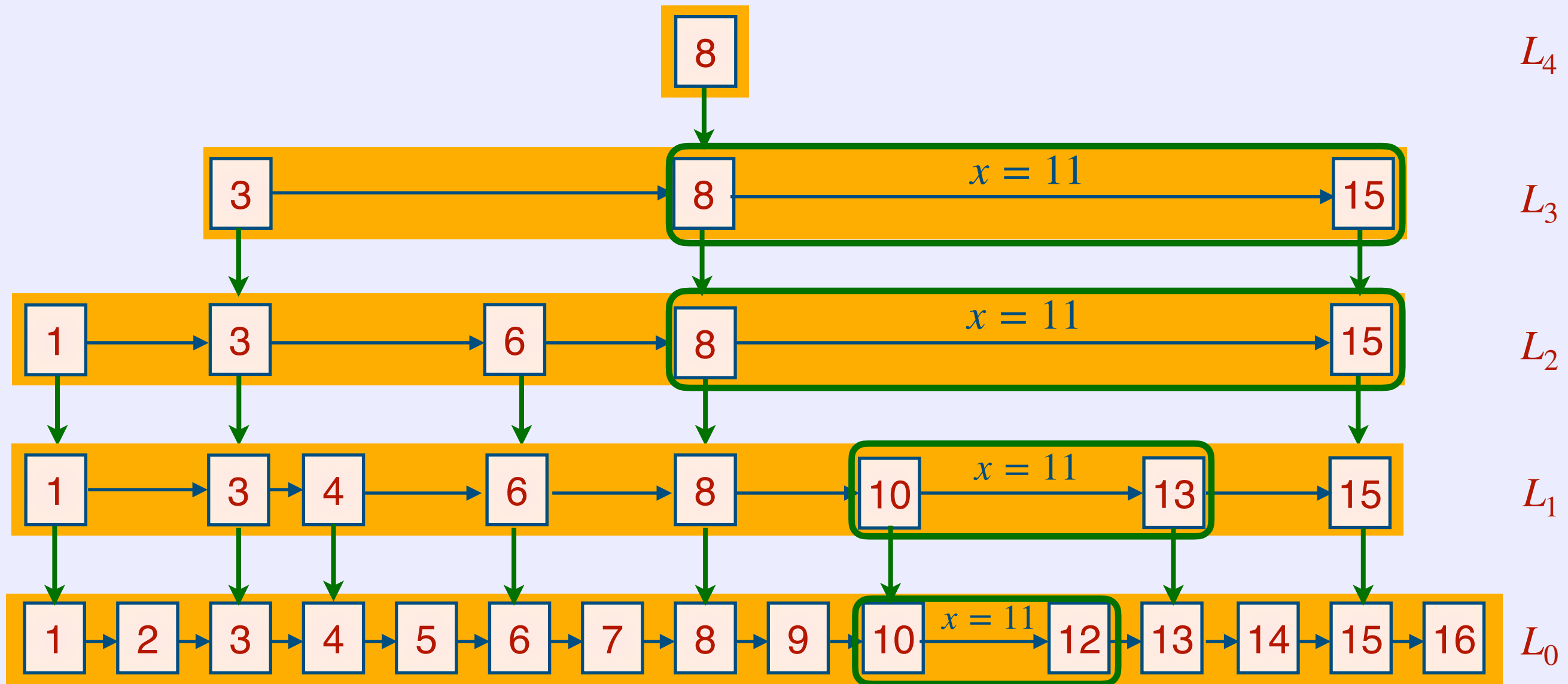


1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list

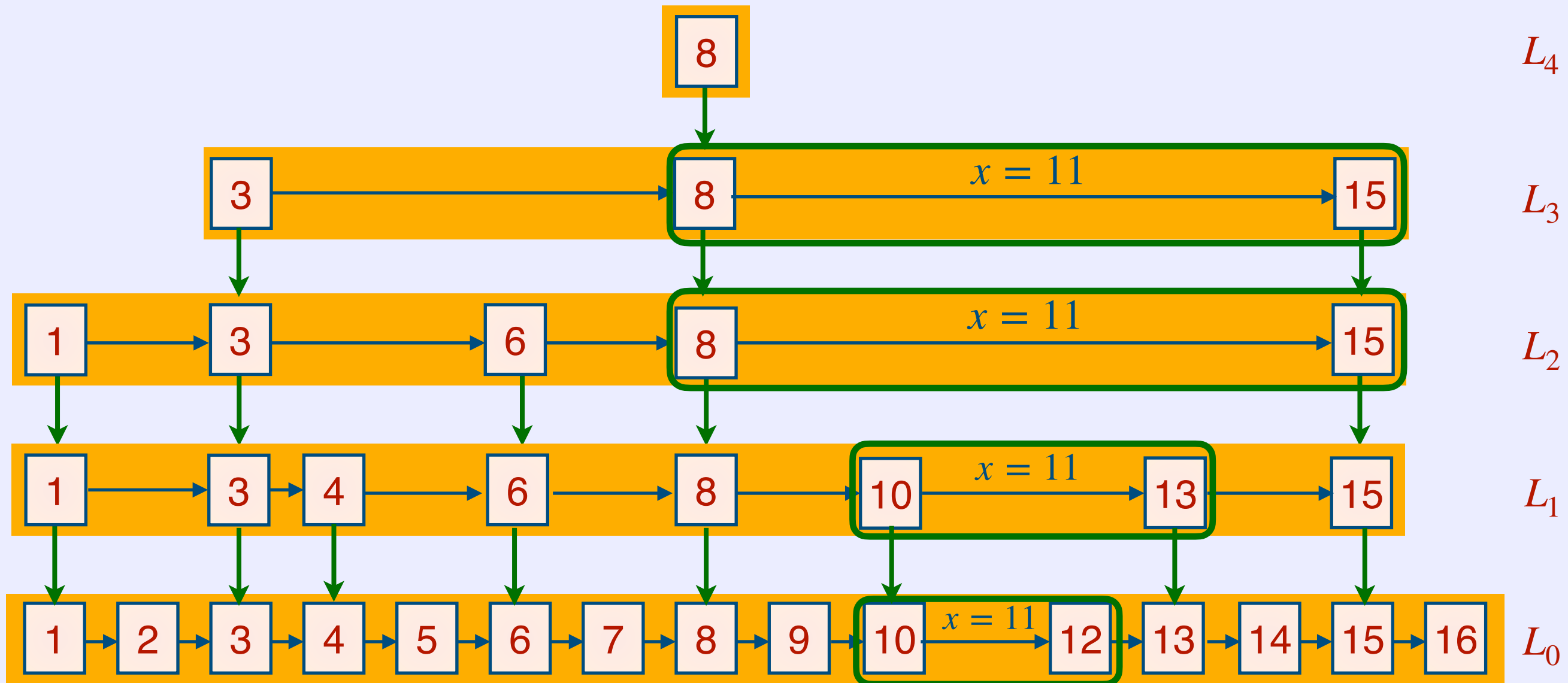


1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list

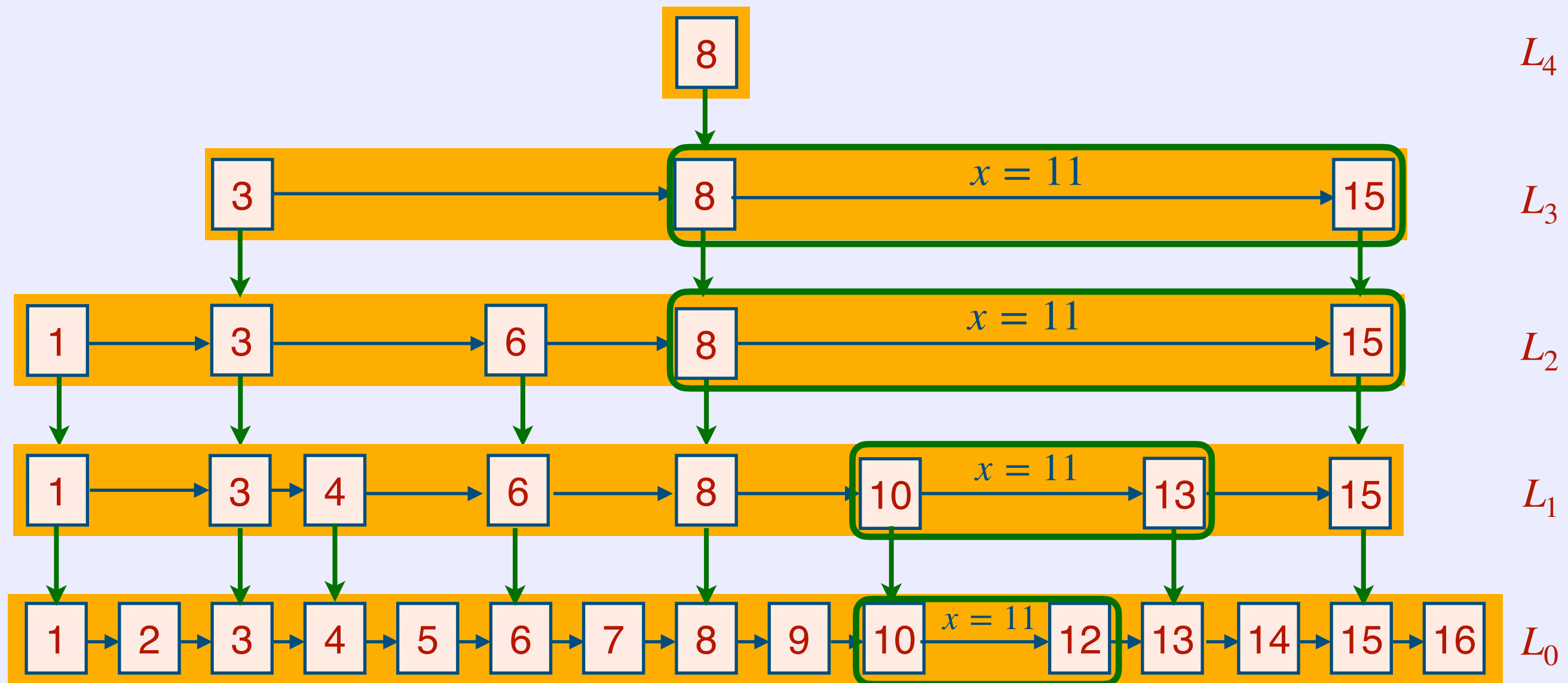


1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.
2. Perform coin tosses to identify lists $L_0, L_1, \dots, L_i$ in which x must be added.

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list



Coin
Tosses



$L_1 \rightarrow$ Keep



$L_2 \rightarrow$ Keep

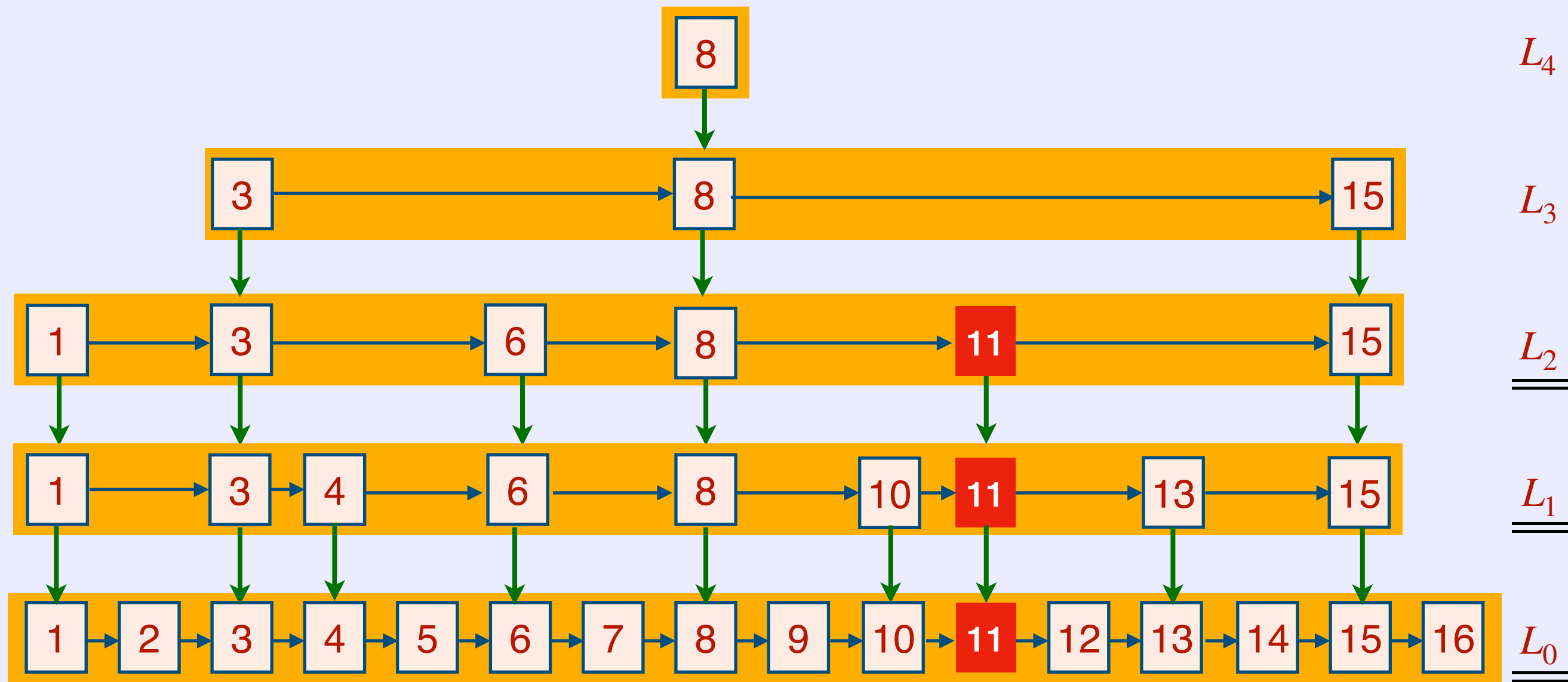


$L_3 \rightarrow$ Don't Keep

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list



Coin
Tosses



$L_1 \rightarrow$ Keep



$L_2 \rightarrow$ Keep

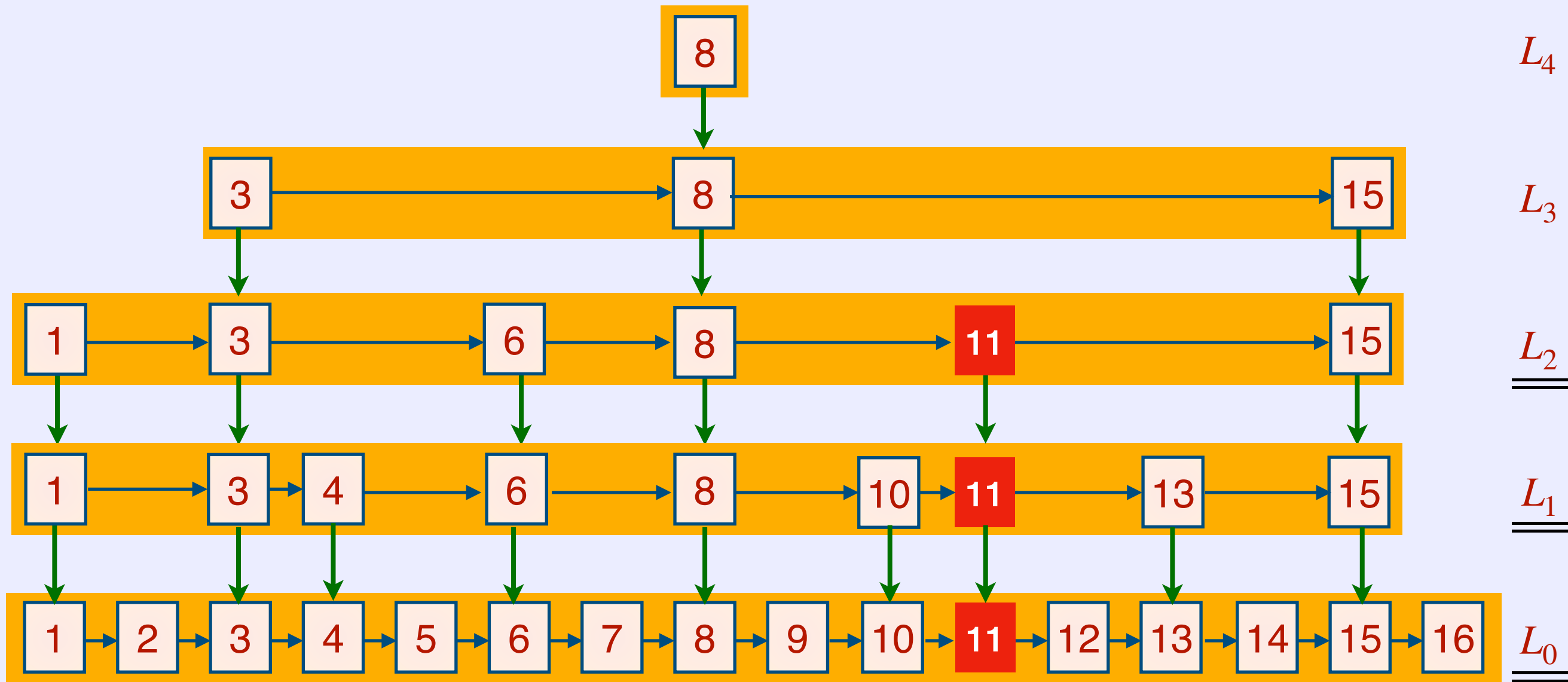


$L_3 \rightarrow$ Don't Keep

Skip Lists - Insertion

Example

Insertion of element
 $x = 11$ in the skip list



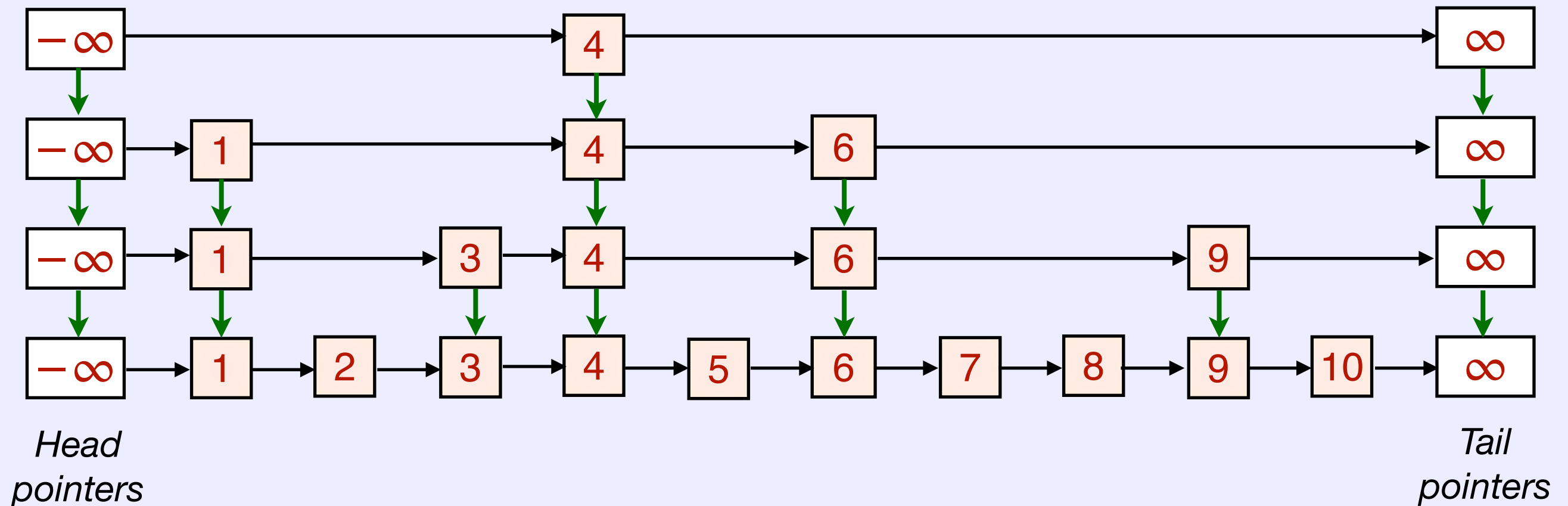
1. For each $i \geq 0$, find *Predecessor* and *Successor* of x in L_i using Search operation.
2. Perform coin tosses to identify lists $L_0, L_1, \dots, L_i$ in which x must be added.
3. Add x to the corresponding lists.

Summary

Skip Lists: A dictionary based on lists with following guarantees

OPERATION	EXPECTED TIME
✓ Search	$O(\log n)$
✓ Deletion	$O(\log n)$
✓ Insertion	$O(\log n)$

Complete Structure of Skip Lists



We include $-\infty$, ∞ in our lists, these are **included in each level**.

So, $-\infty$ corresponds to **Head** pointers, and $+\infty$ corresponds to **Tail** pointers.