# CS240
# Fall 2015

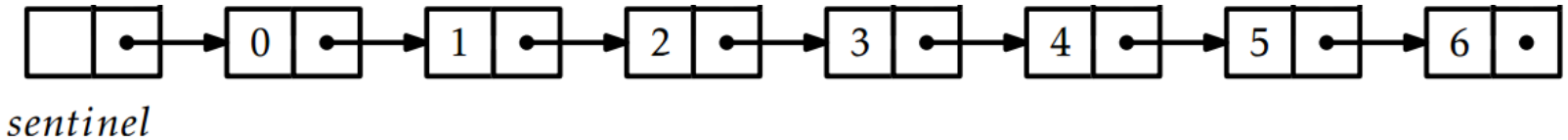
Mike Lam, Professor

# Skip Lists

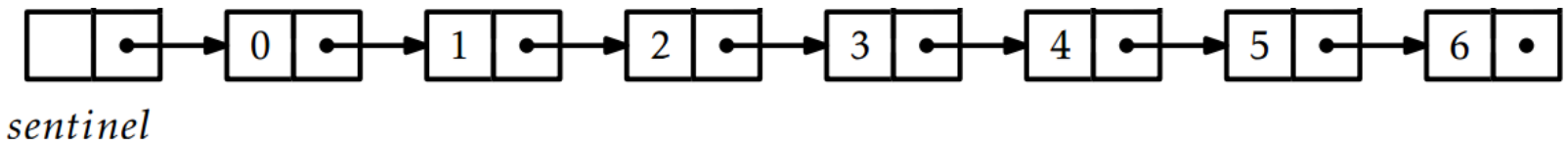
# Skip Lists

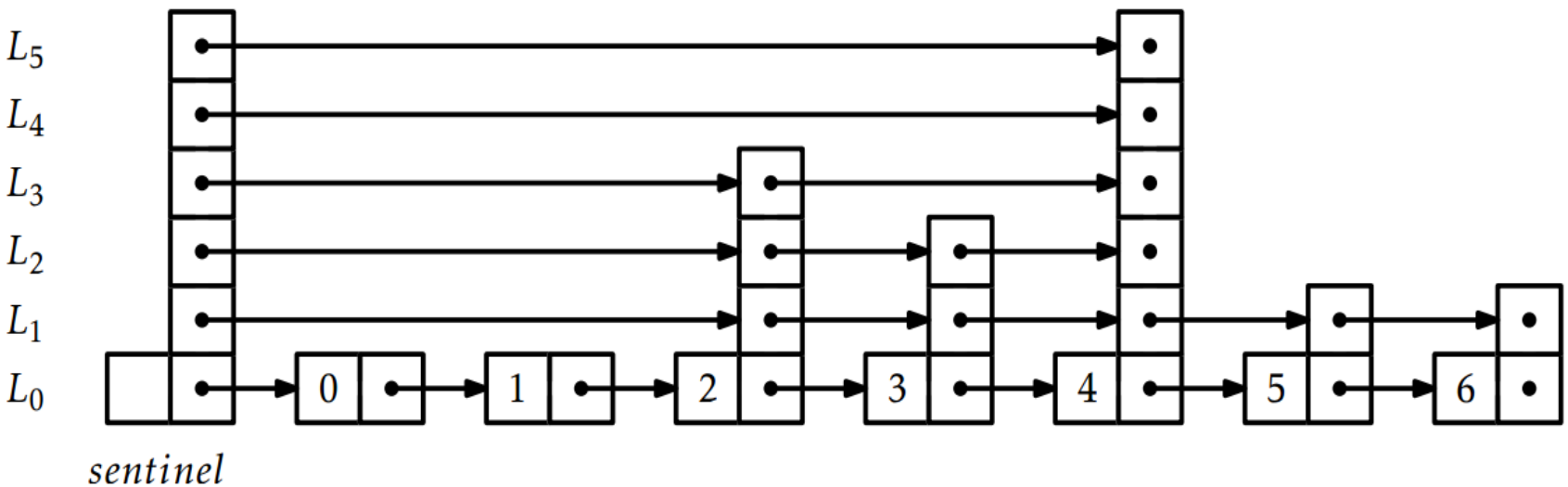
- Multi-level sorted linked list
  - Multiple `next` pointers per node
- Higher levels contain “skip” links to locations further down the list
- All nodes reside in bottom level ($L_0$)
- Some nodes reside in higher levels as well
  - The height of the node is the number of levels in which that node resides
    - i.e., the height of a node is the number of `next` pointers
  - **Caution**: heights are 1-based while levels are 0-based

# Skip Lists



# Skip Lists



# Skip List

Basic idea: move right then move down (and repeat!)

***find***(*x*):

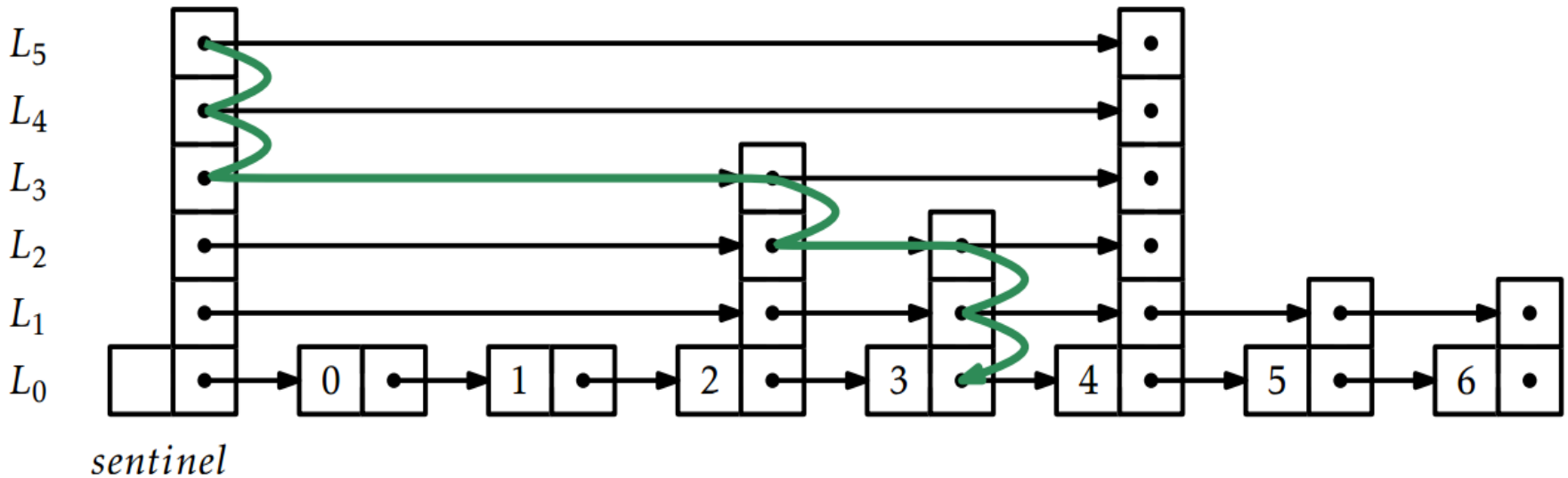
u = head,  r = max_height-1

*while* r ≥ 0*:*

*while* u.next[r] ≠ NULL *and* *u.next[r].data* ≤ *x*:

u = u.next[r]

r = r – 1



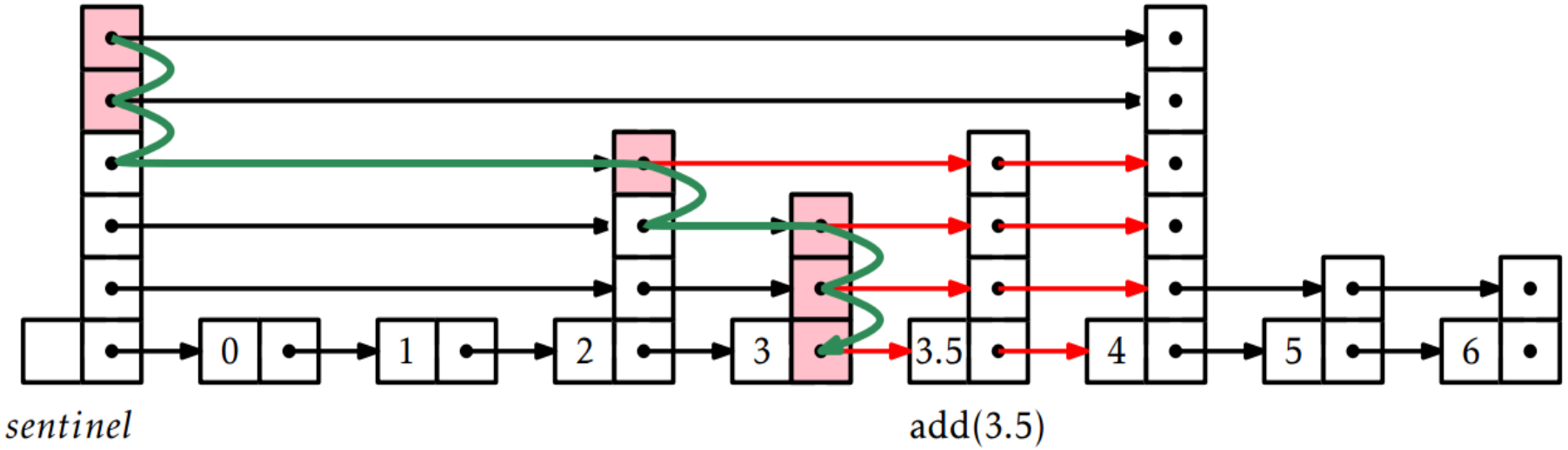
# Adding to a Skip List

- Key concept: randomization
  - Flip a coin several times to determine the maximum height of a node
    - If we get heads, keep flipping!
  - Examples
    - Tails on first try: height = 1
    - Tails on second try: height = 2
    - Tails on third try: height = 3
    - etc.
  - Expected height of a node is 1
    - Consequence: half of all nodes only reside on bottom level

# Adding to a Skip List

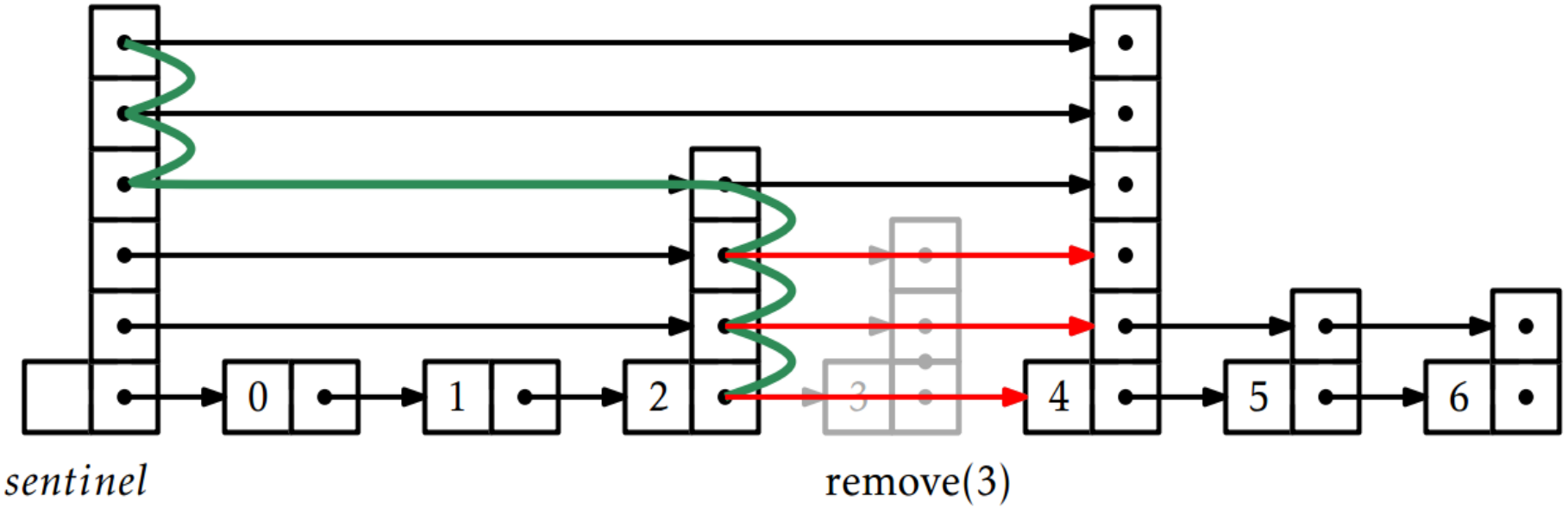
Basic idea: move right then move down (and repeat!)

Insert new node into all levels lower than its height



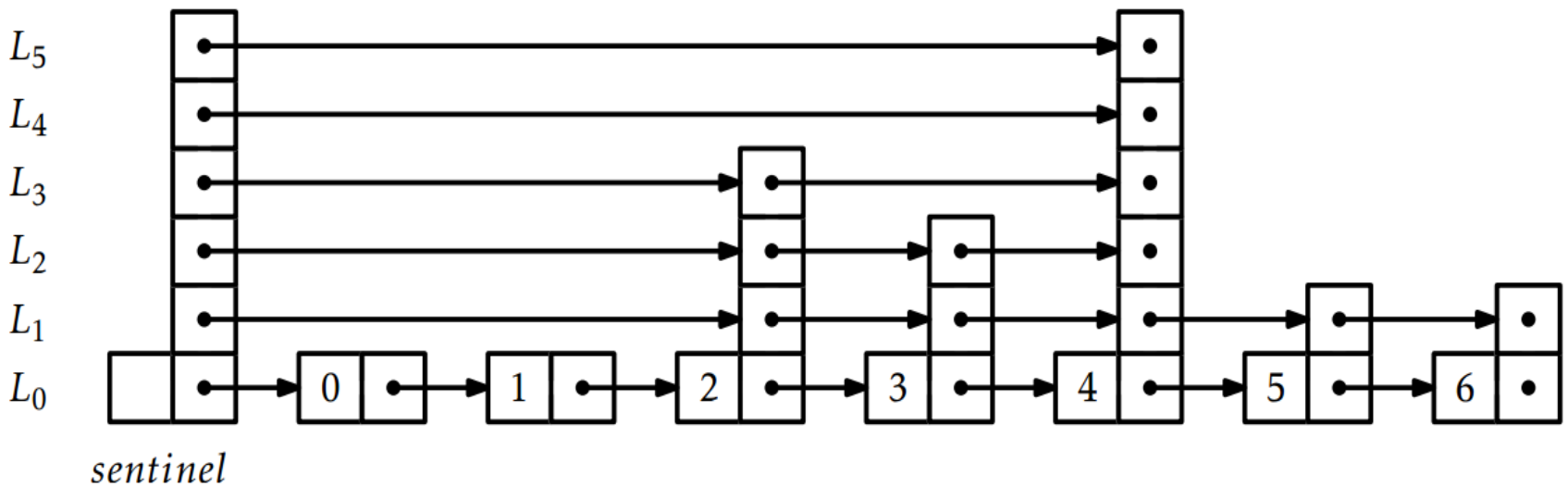
# Removing from a Skip List

Basic idea: move right then move down (and repeat!)
Remove node from any level where it is present



# Skip Lists

- Draw the results of the following operations:
  - **`find`**`(1)` and **`find`**`(2)` and **`find`**`(5)`
  - **`add`**`(1.5)` and **`add`**`(4.5)`
    - `height = 3`
  - **`remove`**`(0)` and **`remove`**`(2)`

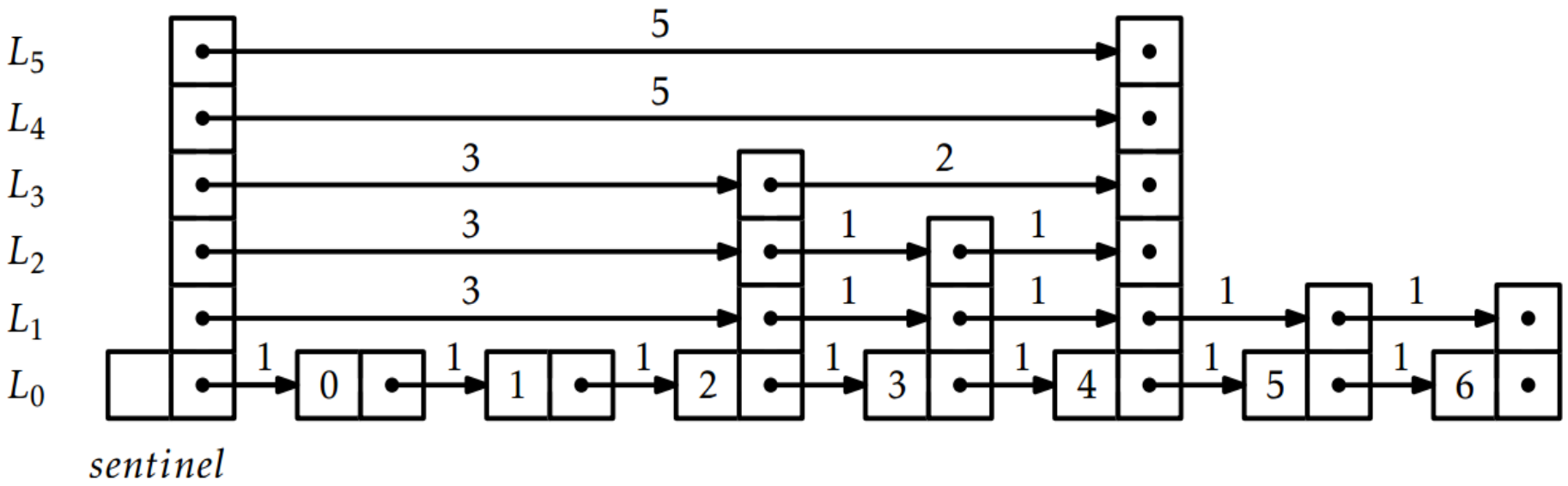


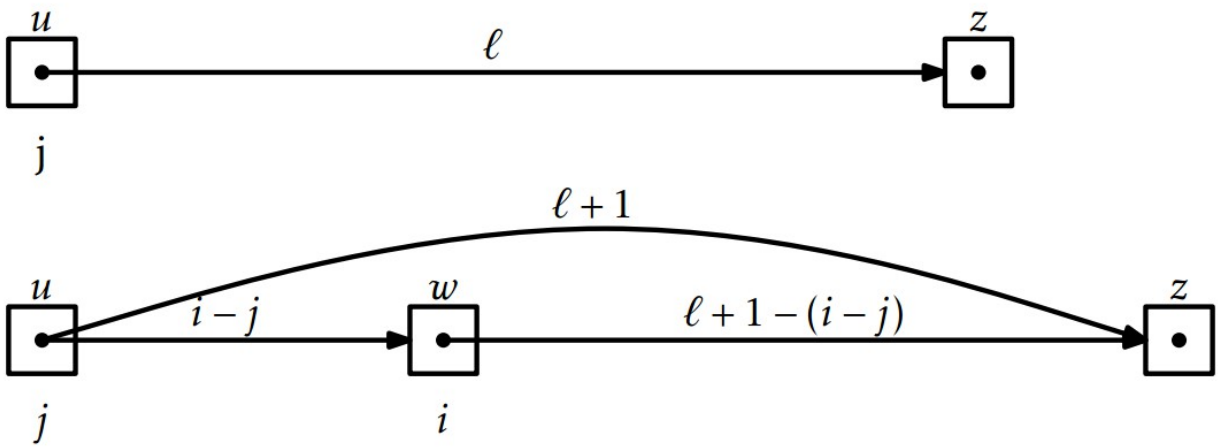
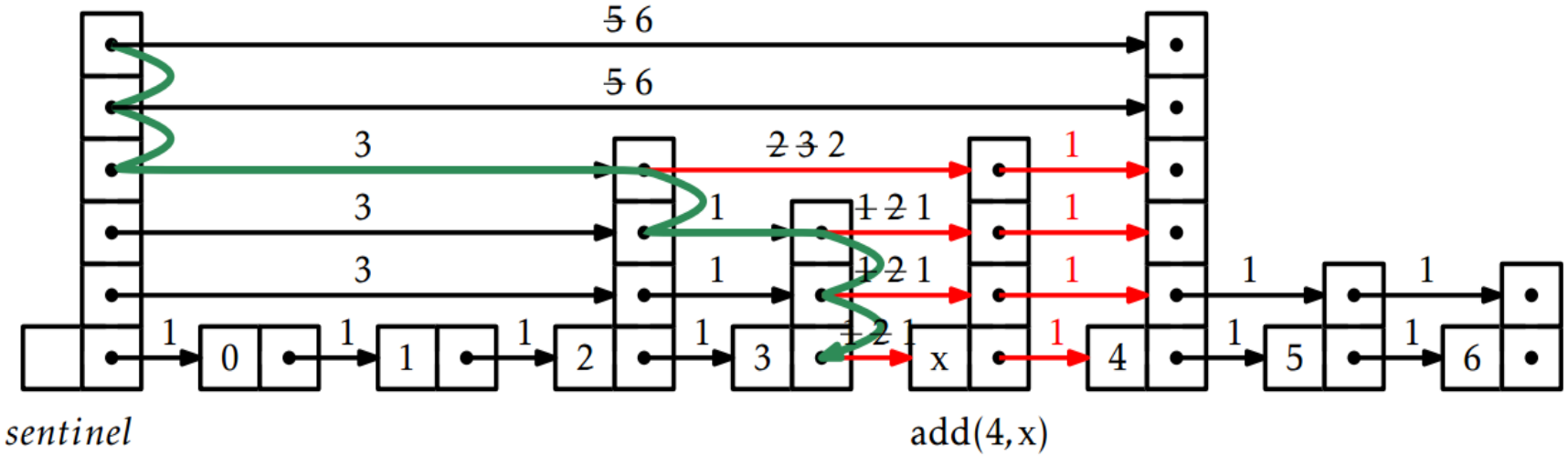
# Skip Lists

- Insert, remove, and search all involve a traversal of the list
  - Move right as far as you can, then go down a level and repeat
  - Skips as many nodes as possible
- Visit an expected constant number of nodes per level
  - The constant is exactly 2 in a "perfect" skip list
- There will be approximately *log n* levels
  - 1/2 of nodes are only on bottom level, 1/4 are in bottom two levels, 1/8 are in bottom three levels, etc.
  - Halving of input space at each level
- Thus, all three operations are *O(log n)*!

# Link Lengths

- Annotate a skip list with *length* info on links
  - Each length is the sum of the lengths of links below it




**Now we can find the item at a given index in *O(log n)* as well!**

# Updating Link Lengths