Warm-Up

Suppose we are given a linked list of distinct integers. We know that $-\infty$ and $+\infty$ are in the list\(^1\) and the list is sorted least to greatest. We want to write \texttt{find}(k), which returns the largest key whose value is \textit{at most} \(k\).

![Linked List Example]

\textbf{Skip Lists}

Let’s add another layer to improve the running time. Which node(s) should be layered?

![Updated Linked List Example]

Update the \texttt{find}(k) function. Does this work if we add a third layer? What about more?

---

\(^1\)Technically, $-\infty$ and $+\infty$ aren’t integers, of course. If their presence bothers you, think of them as \texttt{INT\_MIN} and \texttt{INT\_MAX}.
How do we insert a new element into a skip list?
Assume we have `insertAfterAbove(p, q, k, v)`, which inserts \((k, v)\) after \(p\) and above \(q\) (if not `nullptr`) and returns a pointer to the new node.

How long does `find(k)` take in the worst case?

What is the expected height of a Skip List?

How long does `find(k)` take in expectation?

How many nodes does a Skip List with \(n\) keys have?