Data Structures with Randomness:
Skip Lists

Flashback to Data Structures…

Recall the `List` interface

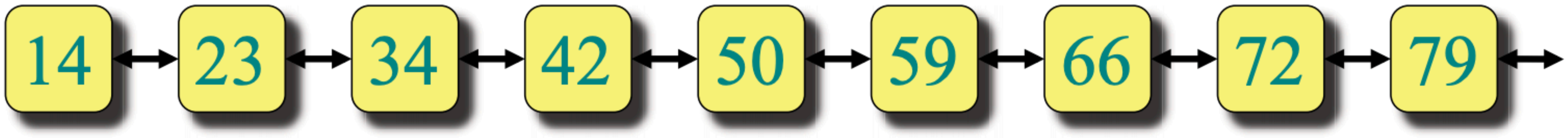
- What are the `List` operations?
- What concrete `List` implementations did we study?

- What are the tradeoffs between arrays and linked lists?
- Do those tradeoffs change when our lists are sorted?
- How does this compare to a binary search tree?

Let's develop a data structure with the strengths of a Binary Search Tree but the (relative) simplicity of a `List`

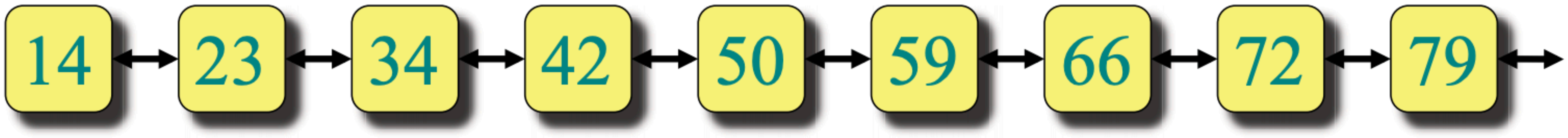
One Linked List

- Start from simplest data structure: (sorted) linked list
- Search cost?
  - $\Theta(n)$
- How can we improve it?



Two Linked Lists

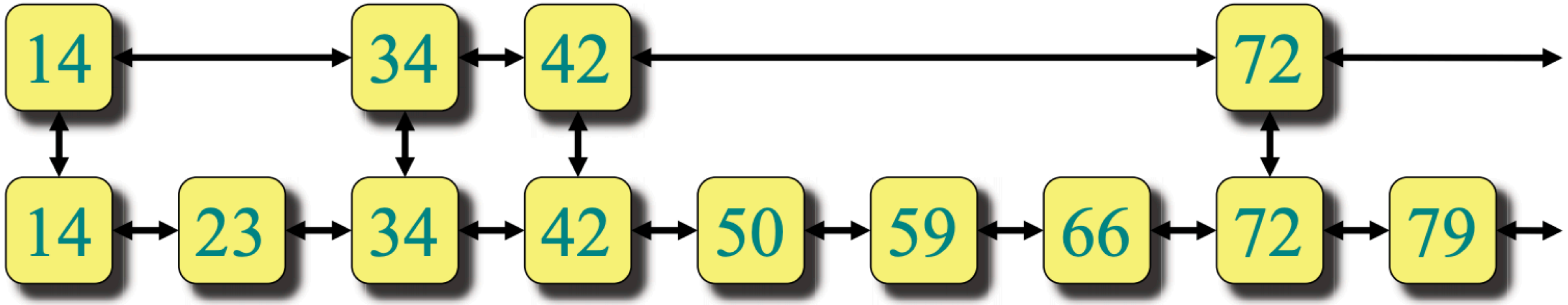
- Suppose you instead had *two* sorted linked lists
  - Each list can contain a subset of the total elements
  - Elements can appear in one or both lists
- **Class exercise.** How can you use two lists to speed up searches?



NYC Subway System

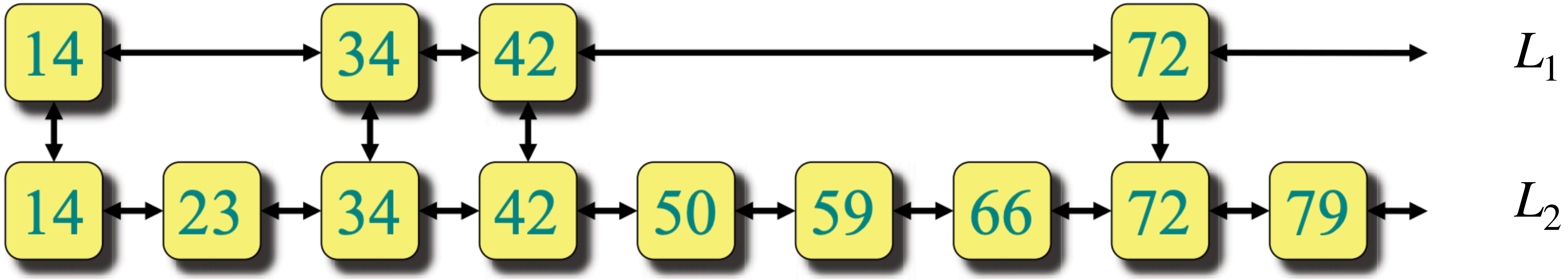
Two Linked Lists

- **Idea**: we have both express and local subways
- Express lines connect a few main stations (and skip a bunch)
- Local lines connect all stations but are slow
- All express stops are also local stops so you can switch



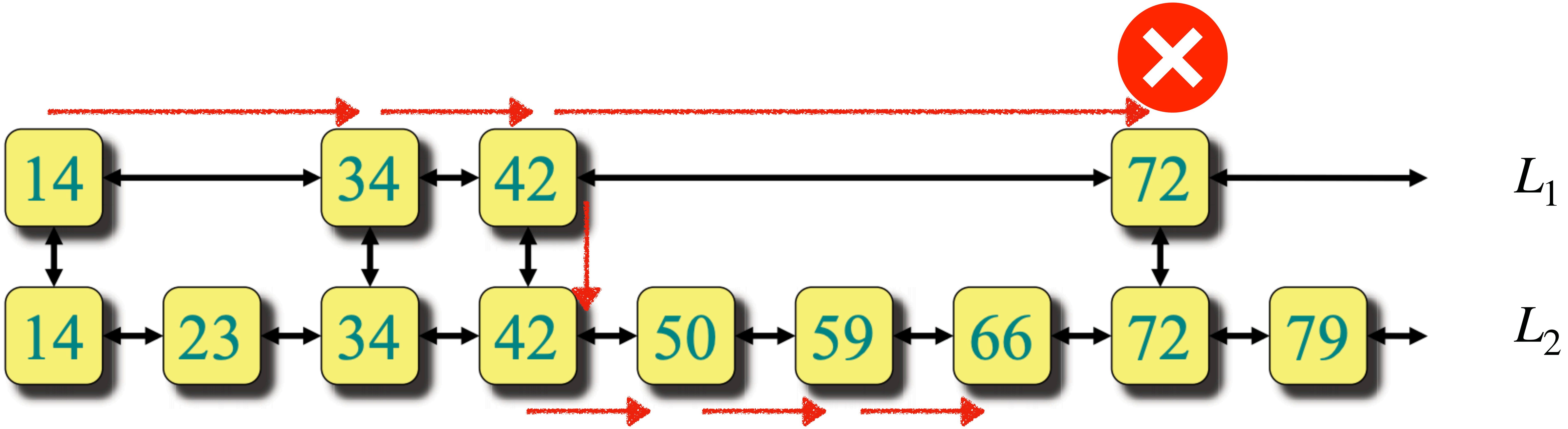
Two Linked Lists

- **Search**(x):
  - Walk right in top linked list L_1 until going right would be too far
  - Walk down to bottom linked list L_2
  - Walk right in L_2 until x is found or reach end (report not found)



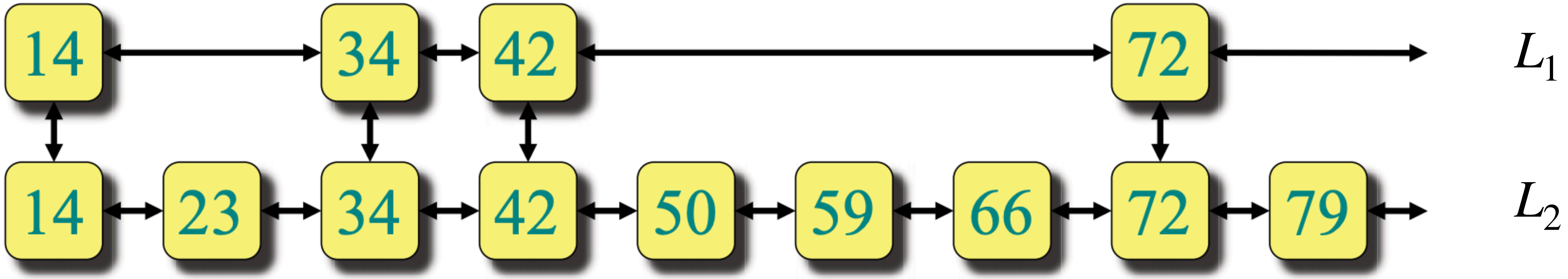
Two Linked Lists

- **Search**(66):
  - Walk right in top linked list L_1 until going right would be too far
  - Walk down to bottom linked list L_2
  - Walk right in L_2 until x is found or reach end (report not found)



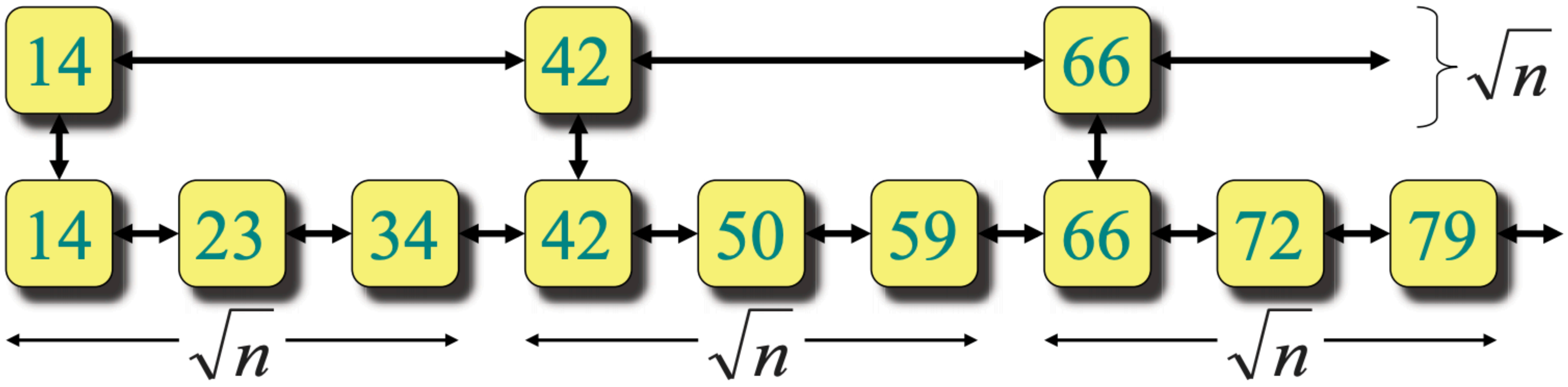
Two Linked Lists

- How should we organize the two lists?
  - Which nodes go in L_1?
  - How much of gap to leave between L_1 elements?
  - **Best approach**: evenly space and promote elements



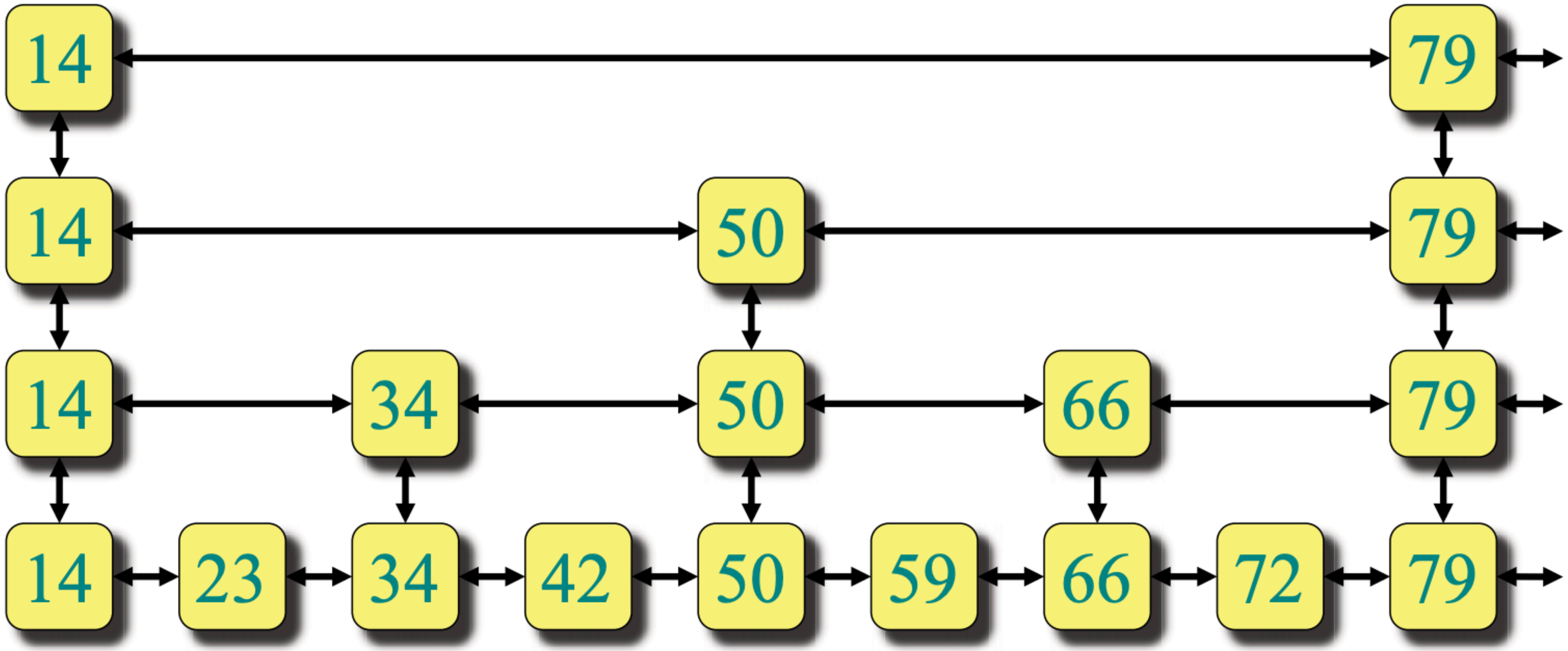
Two Linked Lists

- If gap between elements in top list is g, then the number of elements traversed (search cost) is at most $g + n/g$
- Optimized by setting $g = \sqrt{n}$
- So the search cost is at most $2\sqrt{n}$



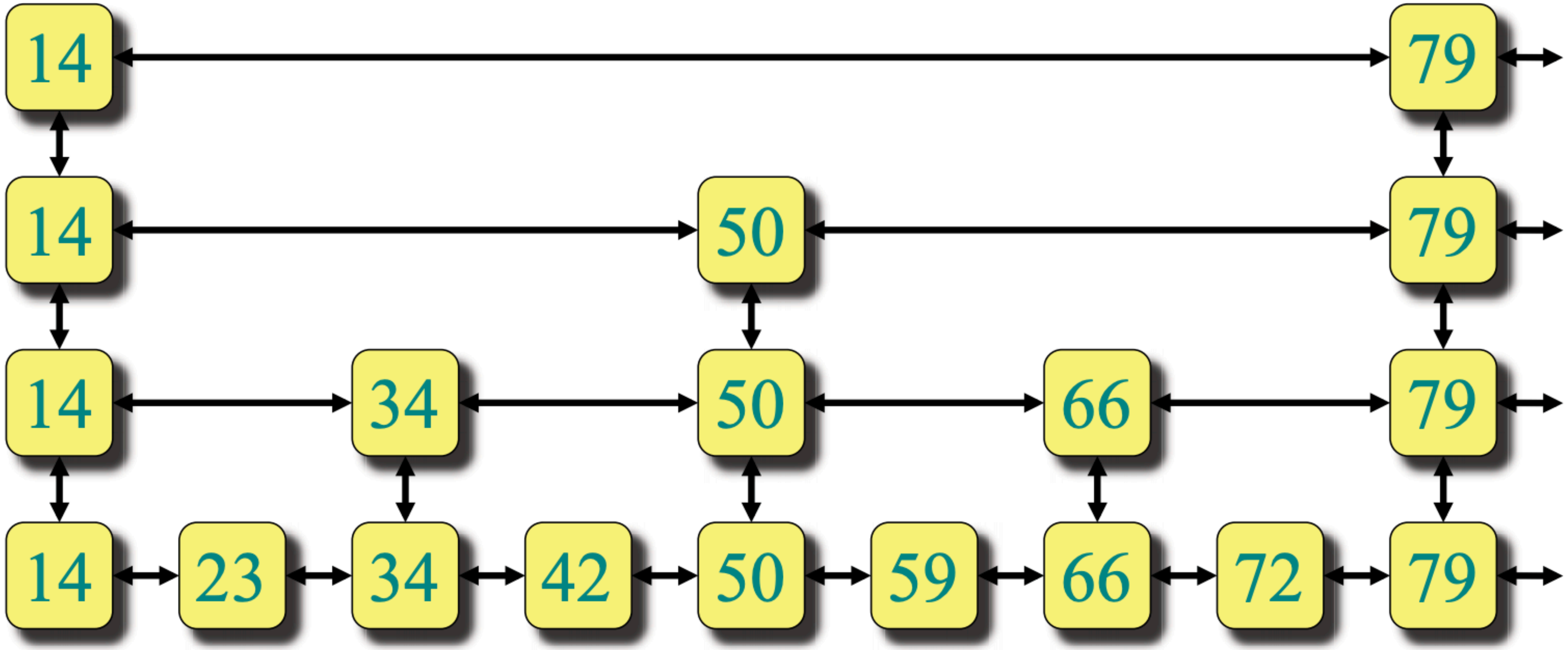
More Linked Lists

- Search cost with two linked list: $2\sqrt{n}$
- Search cost with three linked list: $3n^{1/3}$



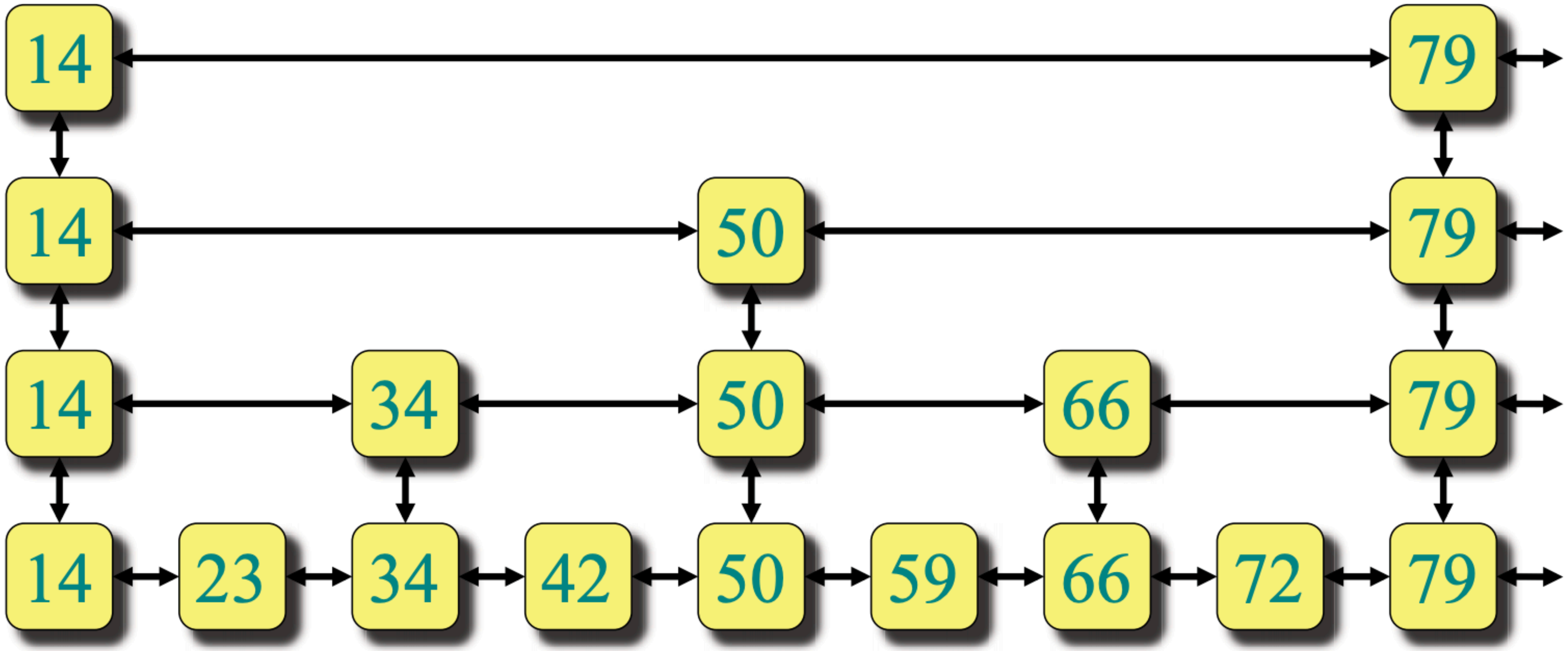
k Linked Lists

- Search cost with k linked lists: $kn^{1/k}$
- Search cost with $\log n$ linked lists: $\log n \cdot n^{1/\log n}$
  - $\log n \cdot n^{1/\log n} = 2 \log n$



Insertion Cost

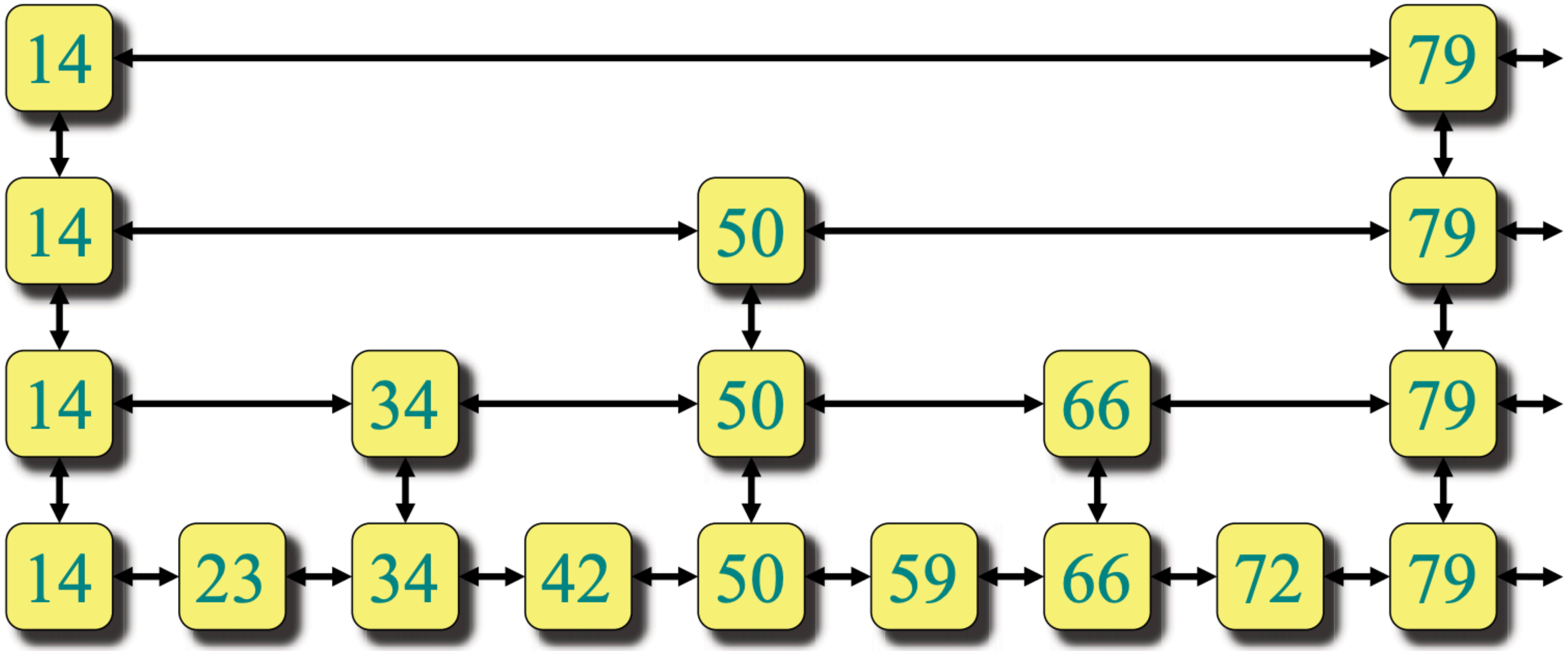
- This is good, but how can we insert?
- Every new element disrupts our spacing
- Idea: use randomness!



Implementing Skip Lists

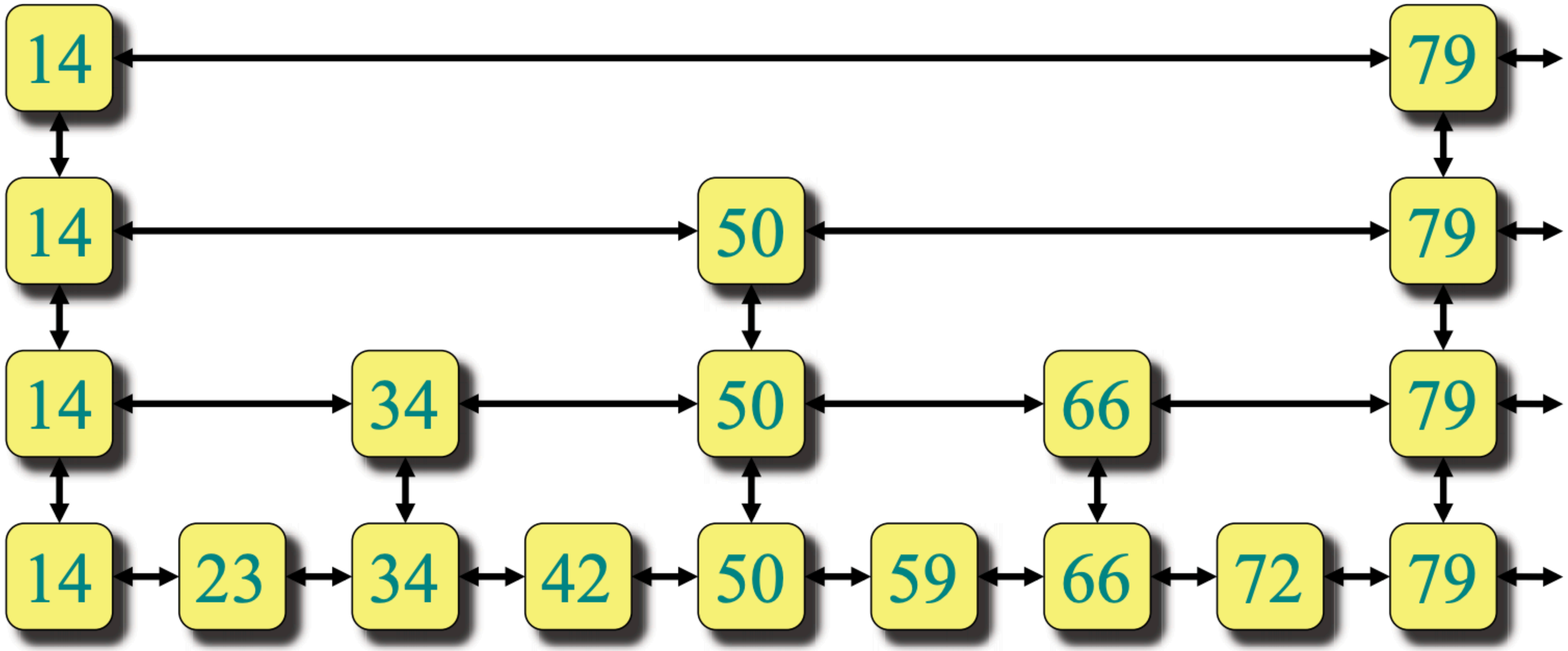
k Linked Lists

- Search cost with k linked lists: $kn^{1/k}$
- Search cost with $\log n$ linked lists: $\log n \cdot n^{1/\log n}$
  - $\log n \cdot n^{1/\log n} = 2\log n$



Insertion Cost

- This is good, but how can we insert?
- Every new element disrupts our spacing
- Reconfiguring to “rebalance” would be expensive



Skip List Details

- Big question: how should we implement Insert(x)?
  - Clearly x must be inserted into at least one list… so the first question is which list(s) should it be added to?
    - Recall our “local line” **invariant**: bottommost list contains all elements (just like the local subway line makes all stops).
    - We must search for x's position in bottommost list and insert it there
    - Any other lists?
      - **Goal:** we want half of the elements to go to next level, similar to a balanced binary tree

Skip List Details

- Big question: how should we implement Insert(x)?
  - **Goal:** we want half of the elements to go to next level, similar to a balanced binary tree
  - **Idea:** Insert x at level 1 (required), then flip a coin
    - If heads: element gets promoted to next level
    - If tails: element stays put at current level
    - Continue flipping until we get a tails
  - Does this achieve our goal (in expectation)?

Skip List Details

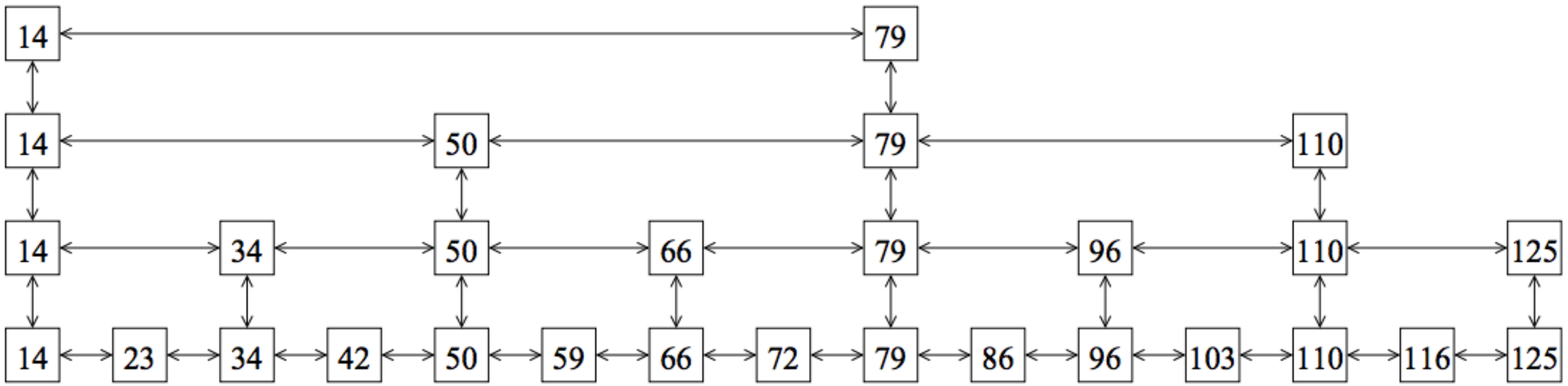
- On average:
  - $1/2$ of the elements are exclusively on the bottom level (T)
  - $1/2$ of the elements go up 1 level (HT)
  - $1/4$ of the elements go up 2 levels (HHT)
  - $1/8$ go up to 3 levels (HHHT)
  - etc.

- Question: Does this randomness on insertion affect any other operations?

Skip List Details

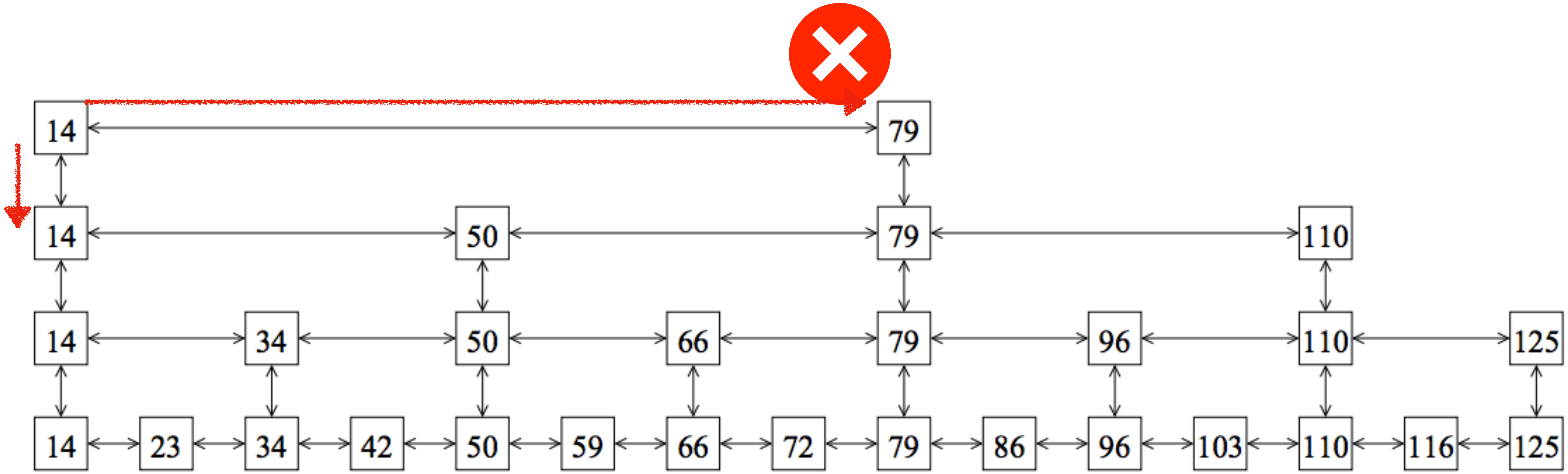
- Search(x):
  - Remains unchanged
  - Start at top list, walk right until just before value gets $>$ target
  - Go down and repeat until:
    - find value $>$ target in bottom list (can't go down any farther)
    - reach last element in bottom list (ran out of elements)
    - element is found (hooray!)

Skip List Search Example



– **Example:** Search for 72

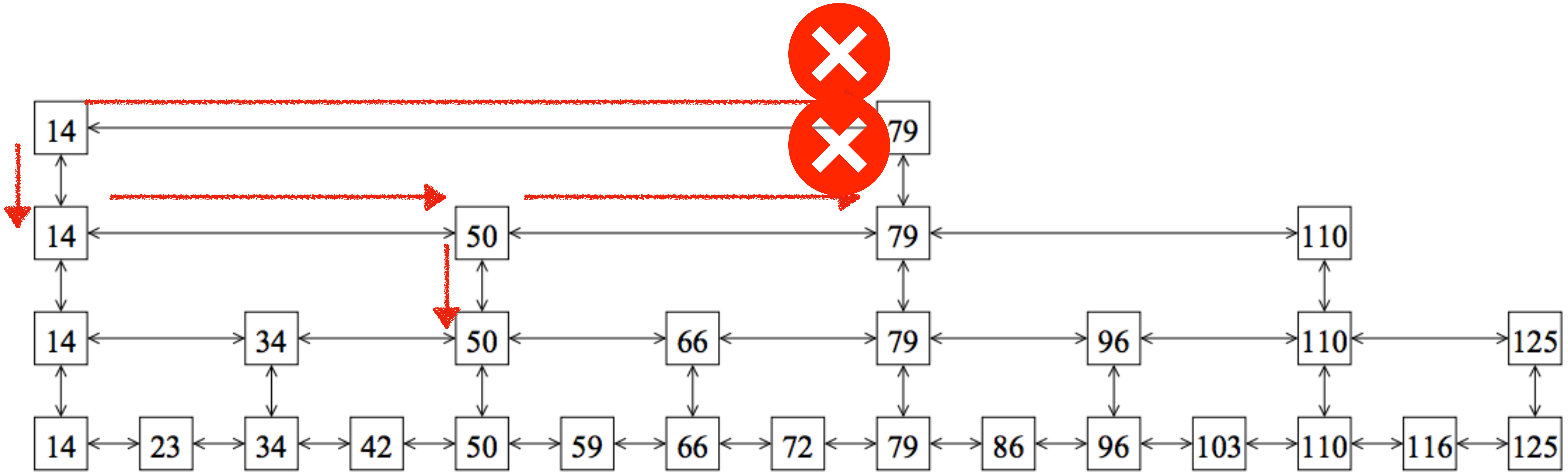
Skip List Search Example



– **Example:** Search for 72

* Level 1: 14 too small, 79 too big; go down 14

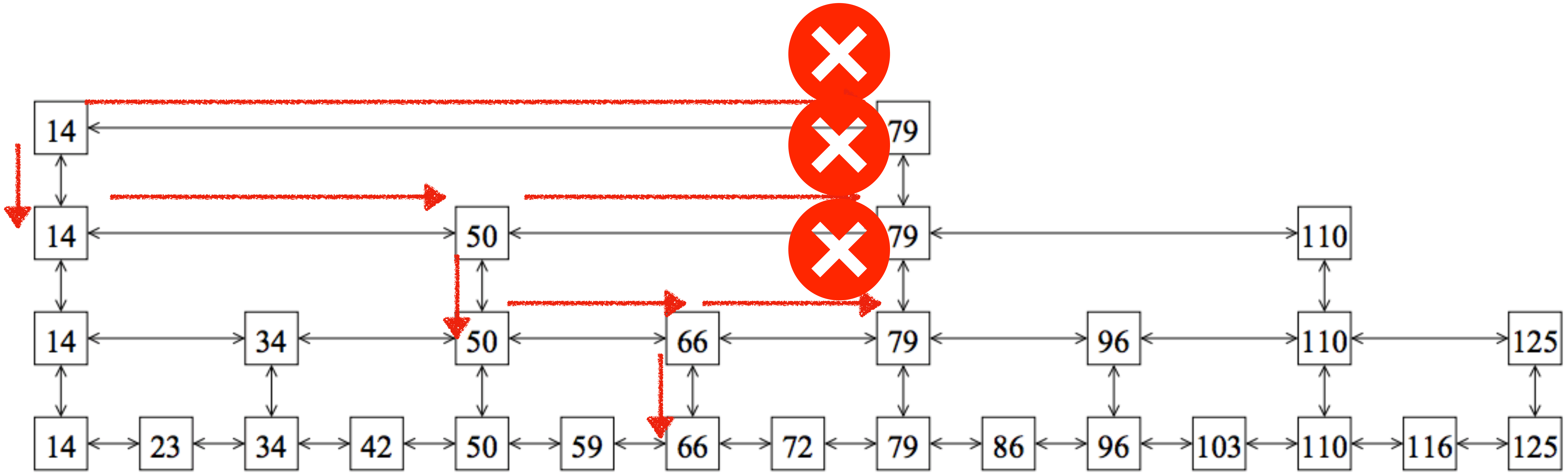
Skip List Search Example



– **Example:** Search for 72

* Level 1: 14 too small, 79 too big; go down 14
* Level 2: 14 too small, 50 too small, 79 too big; go down 50

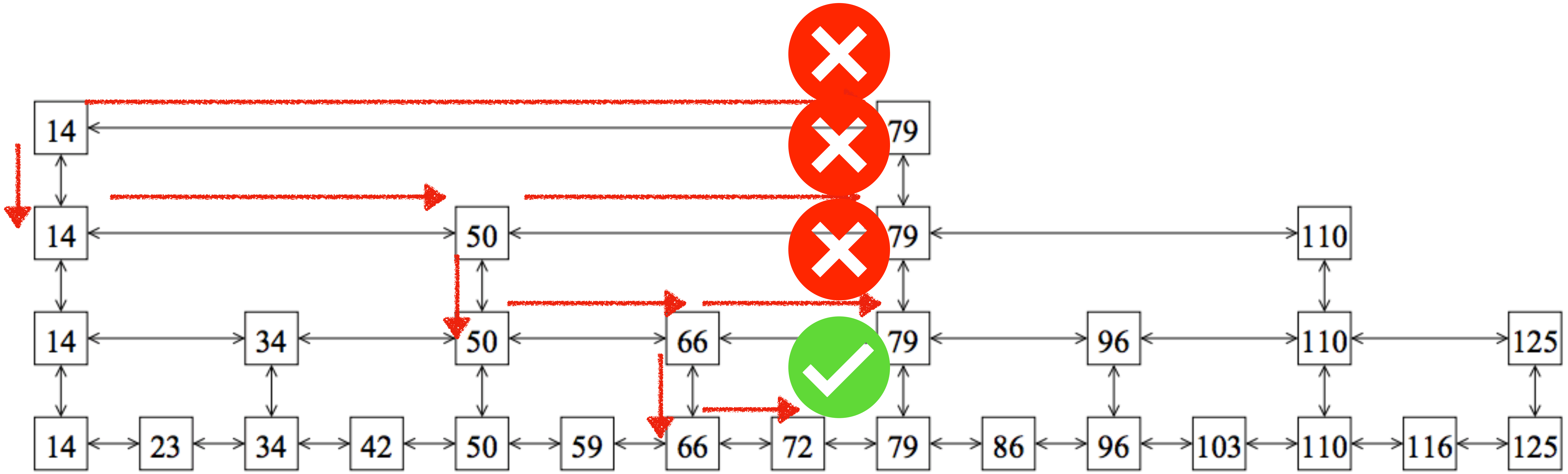
Skip List Search Example



– **Example:** Search for 72

* Level 1: 14 too small, 79 too big; go down 14
* Level 2: 14 too small, 50 too small, 79 too big; go down 50
* Level 3: 50 too small, 66 too small, 79 too big; go down 66

Skip List Search Example



– **Example:** Search for 72

* Level 1: 14 too small, 79 too big; go down 14
* Level 2: 14 too small, 50 too small, 79 too big; go down 50
* Level 3: 50 too small, 66 too small, 79 too big; go down 66
* Level 4: 66 too small, 72 spot on

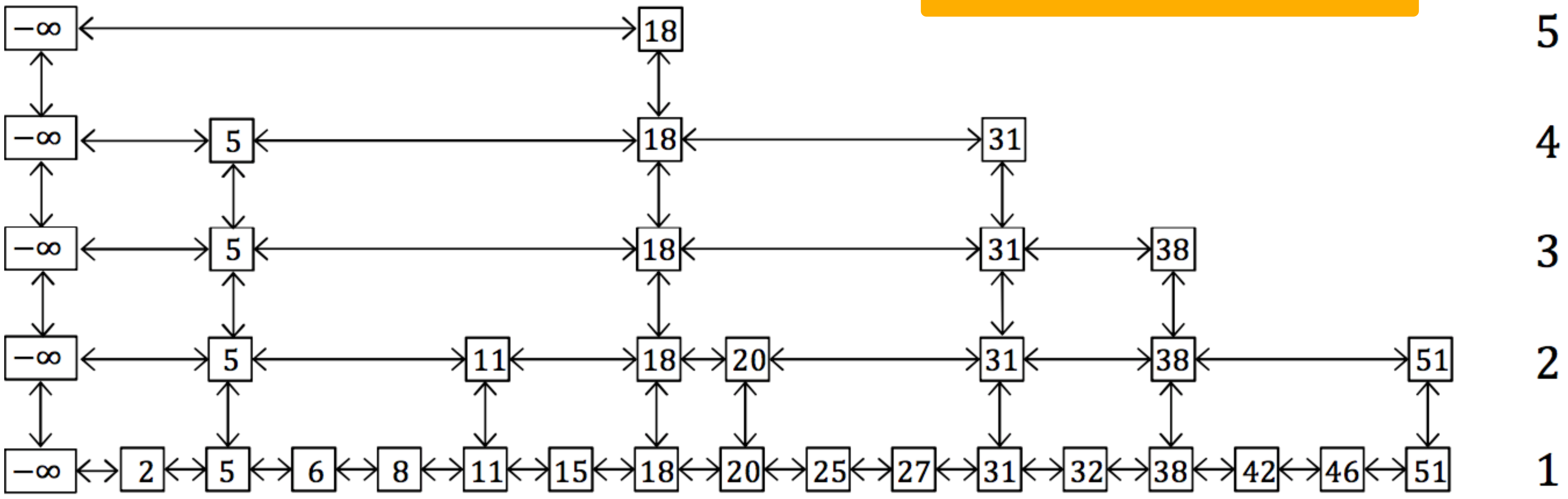
Skip List Analysis: Height

Let L_k be the set of all items in level $k \geq 1$.

- Height of an element x: $\ell(x) = \max\{k \mid x \in L_k\}$
- Height of entire skip list L: $h(L) = \max\{\ell(x) \mid x \in L_0\}$

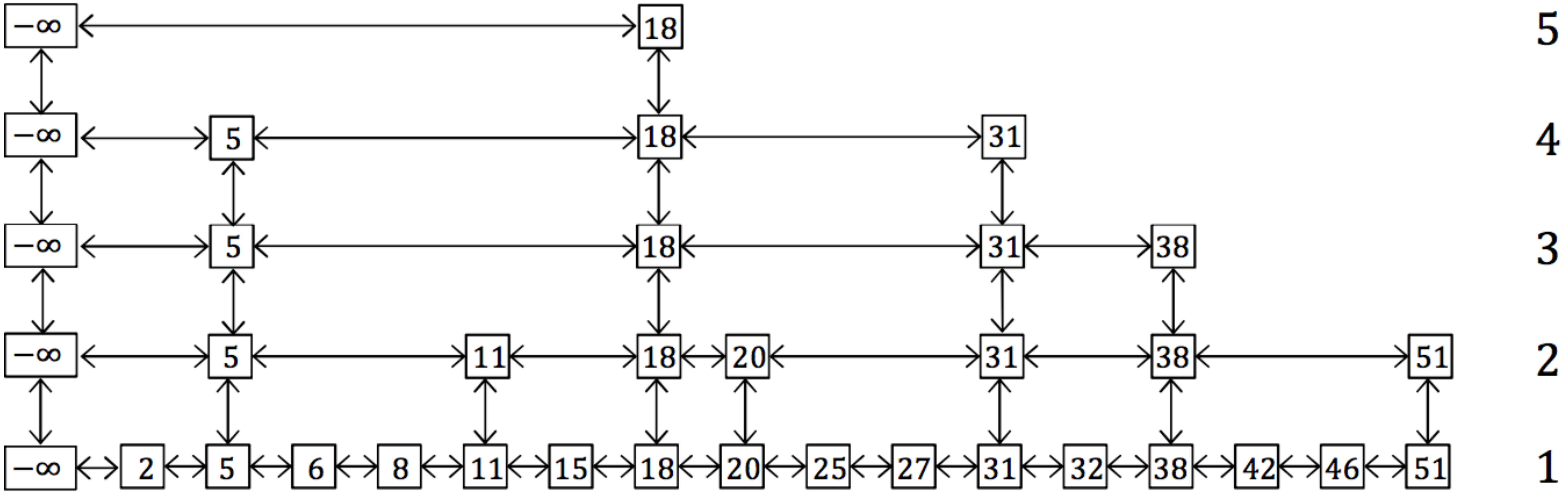
Maximum level that an element appears in

Maximum height among all elements in the list



Skip List Analysis: Height

- Expected height of a node?
    - **Question**: in an experiment with probability p of success, what is the expected number of trials until success?



Skip List Analysis: Height

Let X denote the random variable equal to the number of flips performed until we reach a tails (stopping condition for promotion). What is $E[X]$?

- For $i > 0$, we have $Pr[X = i] = (1-p)^{i-1}p$

$i - 1$ failures, then first success

- $E[X] = \sum_{i=0}^{\infty} i \cdot Pr[X = i] = \sum_{i=1}^{\infty} i(1-p)^{i-1}p \quad = \dfrac{p}{1-p} \sum_{i=1}^{\infty} i(1-p)^{i}$

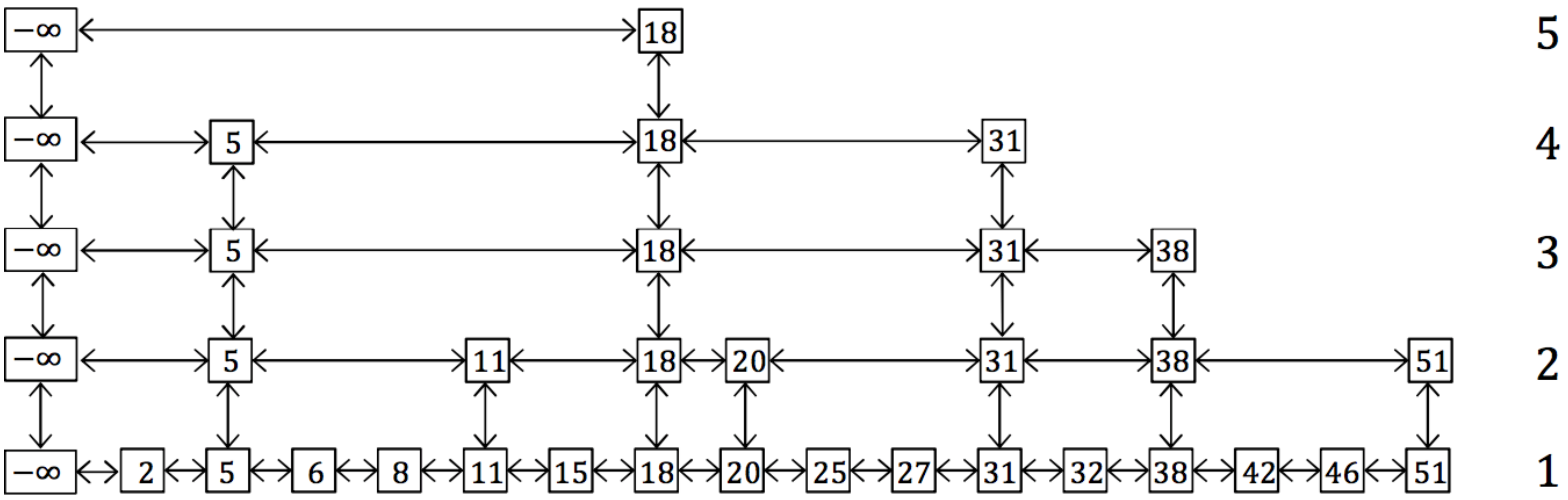
$$= \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

- If $p = \dfrac{1}{2}$, then $E[X] = 2$

See page 720 in the text (Section 13.3)
Useful for homework!

Skip List Analysis: Height

- Expected height of a node?
  - Expected number of trials until success (tail): 2
- Worst-case height? $h(L) = \max\{\ell(x) \mid x \in L\}$



Skip List Analysis: Height

- **Claim.** A skip list with n elements has height $O(\log n)$ levels with high probability
- Goal: show that the probability that it has more than $d \log n$ levels is at most $1/n^c$, where the constants c, d usually depend on each other
- **Proof**. For any $x \in L$, $k \geq 1$, the probability that height of x is k
- What is Pr$[\ell(x) = k]$?

$$= (1 - p)^{k-1} p = (1 - \frac{1}{2})^{k-1} \frac{1}{2} = \frac{1}{2^k}$$

Skip List Analysis: Height

- **Claim.** A skip list with n elements has height $O(\log n)$ levels with high probability
- Goal: show that the probability that it has more than $d \log n$ levels is at most $1/n^c$, where the constants c, d usually depend on each other
- **Proof**. For any $x \in L$, $k \geq 1$, the probability that height of x is k
- What is $\Pr[\ell(x) = k] = \dfrac{1}{2^k}$
- $\Pr[\ell(x) > k]$ is probability $\ell(x)$ is $k+1$, $k+2$, ... the probability decreases by half each time, thus is at most $\dfrac{1}{2^k}$

Skip List Analysis: Height

- **Claim**. A skip list with n elements has height $O(\log n)$ levels w.h.p.
- **Proof**. For any $x \in L,\ k \geq 1$, the probability that height of x is k
- $\Pr[\ell(x) > k] = \sum_{k+1}^{\infty} \Pr[\ell(x) = i] = \sum_{i=k+1}^{\infty} \frac{1}{2^i} = \frac{1}{2^k}$
- $\Pr[h(L) > k] = \Pr[\cup_{x \in L} \ell(x) > k] \leq \sum_{x \in L} \Pr[\ell(x) > k] = \frac{n}{2^k}$

Union bound

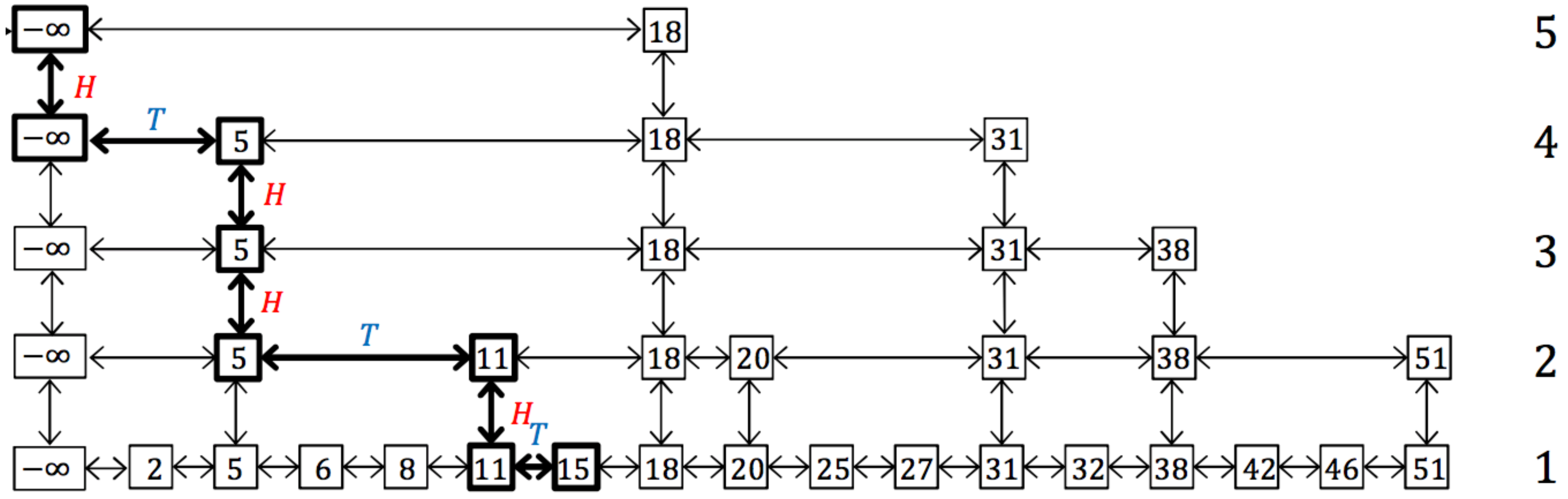
$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

$P(A \cup B) \leq P(A) + P(B)$

- $\Pr[h(L) > c \log n] \leq \frac{1}{n^{c-1}}$ [pick any $c > 2$ for w.h.p.]
- Thus, height of skip is $O(\log n)$ with high probability

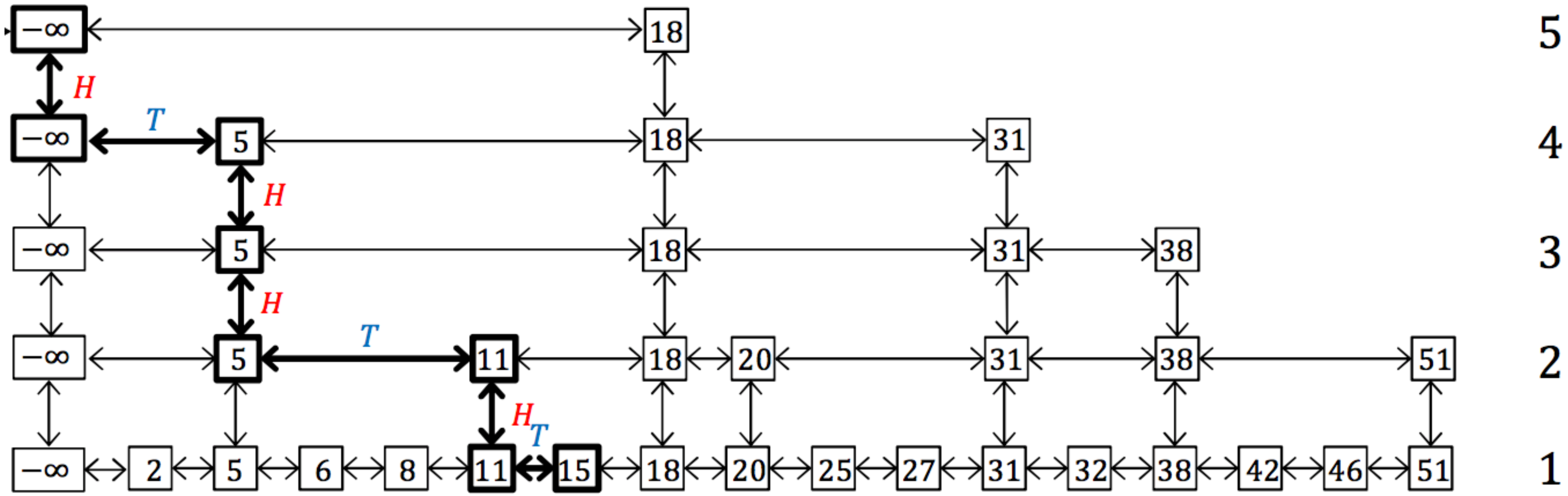
Skip List Search Cost

- **Claim.** Search cost in a skip list is $O(\log n)$ with high probability
- **Proof**. Idea think of the search path "backwards"
- Starting at the target element, going left or up until you reach root or sentinel node $(-\infty)$



Skip List Search Cost

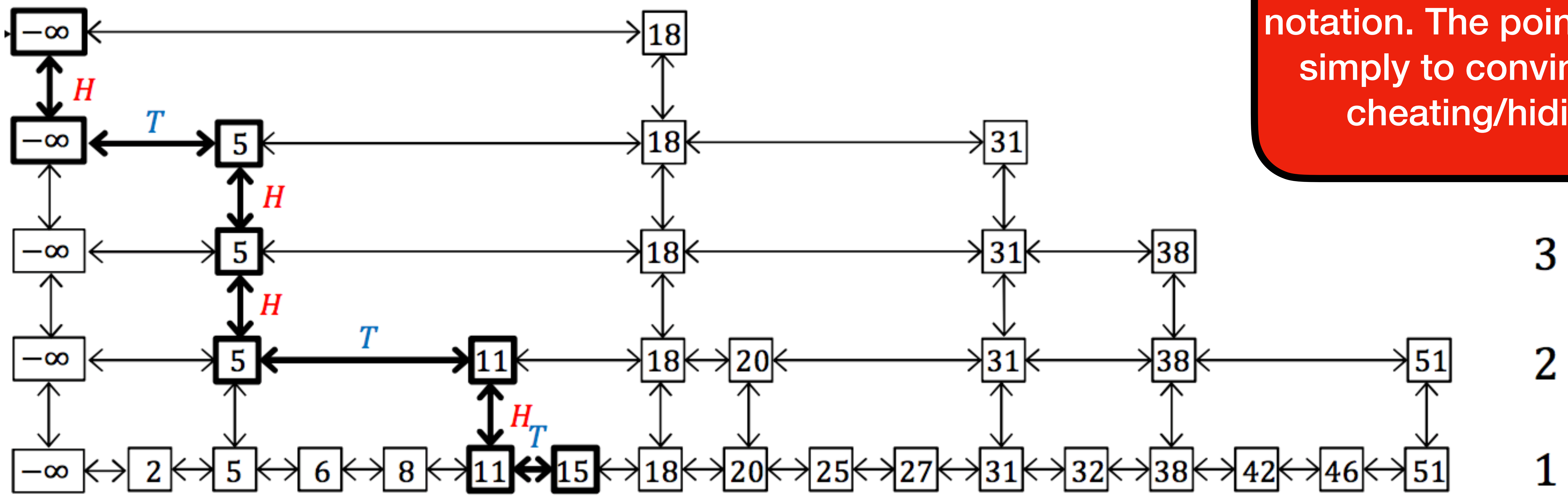
- Backwards search path, when do go up versus left?
- If node wasn't promoted (got tails here), then we go [came from] left
- If node was promoted (got heads here), then we go [came from] top



Skip List Search Cost

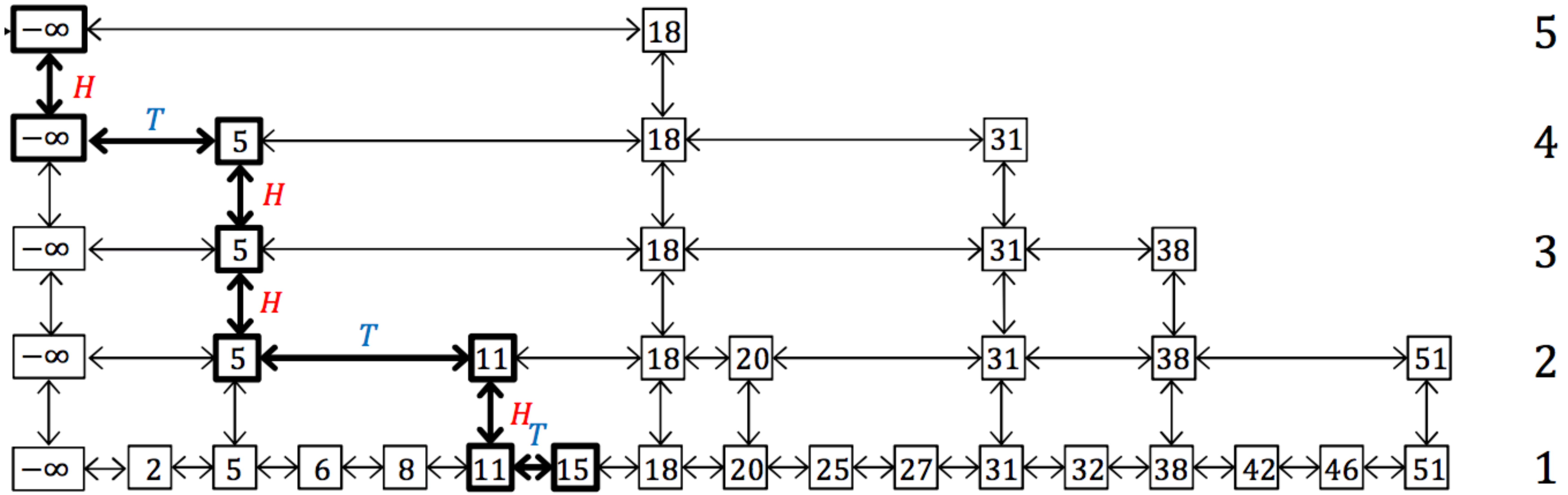
- How many consecutive tails in a row? (left moves on a level)
  - Same analysis as the height! $O(\log n)$
  - $O(\log^2 n)$ length overall—but we claimed $O(\log n)$ earlier

We are about to get very deep into notation. The point of showing this is simply to convince you we aren't cheating/hiding something.



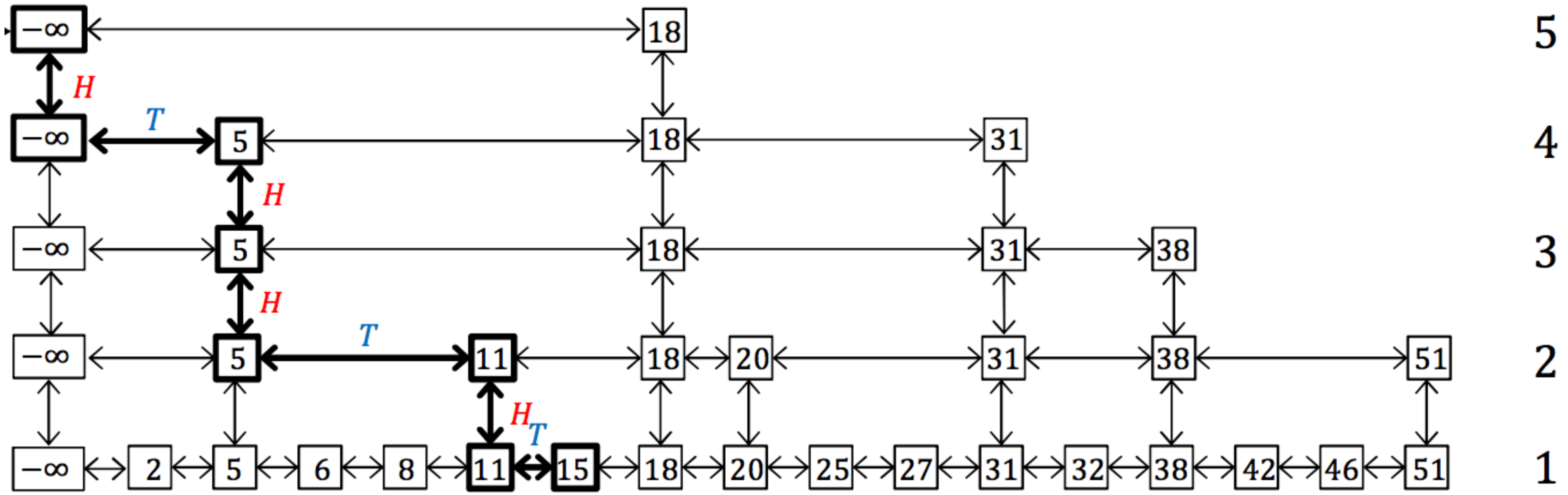
Skip List Search Cost

- Search path is a sequence of $HHHTTTHHTT\ldots$
- How many "up" moves (H) before we are done?
  - Height: $c \log n$ with high probability



Skip List Search Cost

- Search ends when we reach top list: have seen at least $c \log n$ heads
- **Search cost**: Can we bound the number of times do we need to flip a coin until we see $c \log n$ heads with high probability?



Coin Flipping

- **Claim.** Number of flips until $c \log n$ heads is $\Theta(\log n)$ with high probability, that is, with probability $1 - 1/n^c$
  - Note. Constant in $\Theta(\log n)$ will depend on c
- **Proof**. Say we flip $10c \log n$ coins
  - Pr[exactly $c \log n$ heads]

$$= \binom{10c \log n}{c \log n} \cdot \left(\frac{1}{2}\right)^{c \log n} \cdot \left(\frac{1}{2}\right)^{9c \log n}$$

  - Pr[at most $c \log n$ heads] $\leq \binom{10c \log n}{c \log n} \cdot \left(\frac{1}{2}\right)^{9c \log n}$

Coin Flipping

- **Claim.** Number of flips until $c \log n$ heads is $\Theta(\log n)$ with high probability, that is, with probability $1 - 1/n^c$

Applied "Deathbed formula"

- **Proof**. $\Pr[\text{at most } c \log n \text{ heads}] \leq \left(\dfrac{e \cdot 10c \log n}{c \log n}\right)^{c \log n} \cdot \left(\dfrac{1}{2}\right)^{9c \log n}$

$$= (10e)^{c \log n} \cdot \left(\frac{1}{2}\right)^{9c \log n}$$

Coin Flipping

- **Claim.** Number of flips until $c \log n$ heads is $\Theta(\log n)$ with high probability, that is, with probability $1 - 1/n^c$

- **Proof**. $\Pr[\text{at most } c \log n \text{ heads}] \leq \left(\frac{e \cdot 10c \log n}{c \log n}\right)^{c \log n} \cdot \left(\frac{1}{2}\right)^{9c \log n}$

$$= (10e)^{c \log n} \cdot \left(\frac{1}{2}\right)^{9c \log n}$$

$$= 2^{\log(10e) \cdot c \log n} \cdot \left(\frac{1}{2}\right)^{9c \log n}$$

$$= 2^{(\log(10e) - 9) \cdot c \log n} = 2^{-d \log n}$$

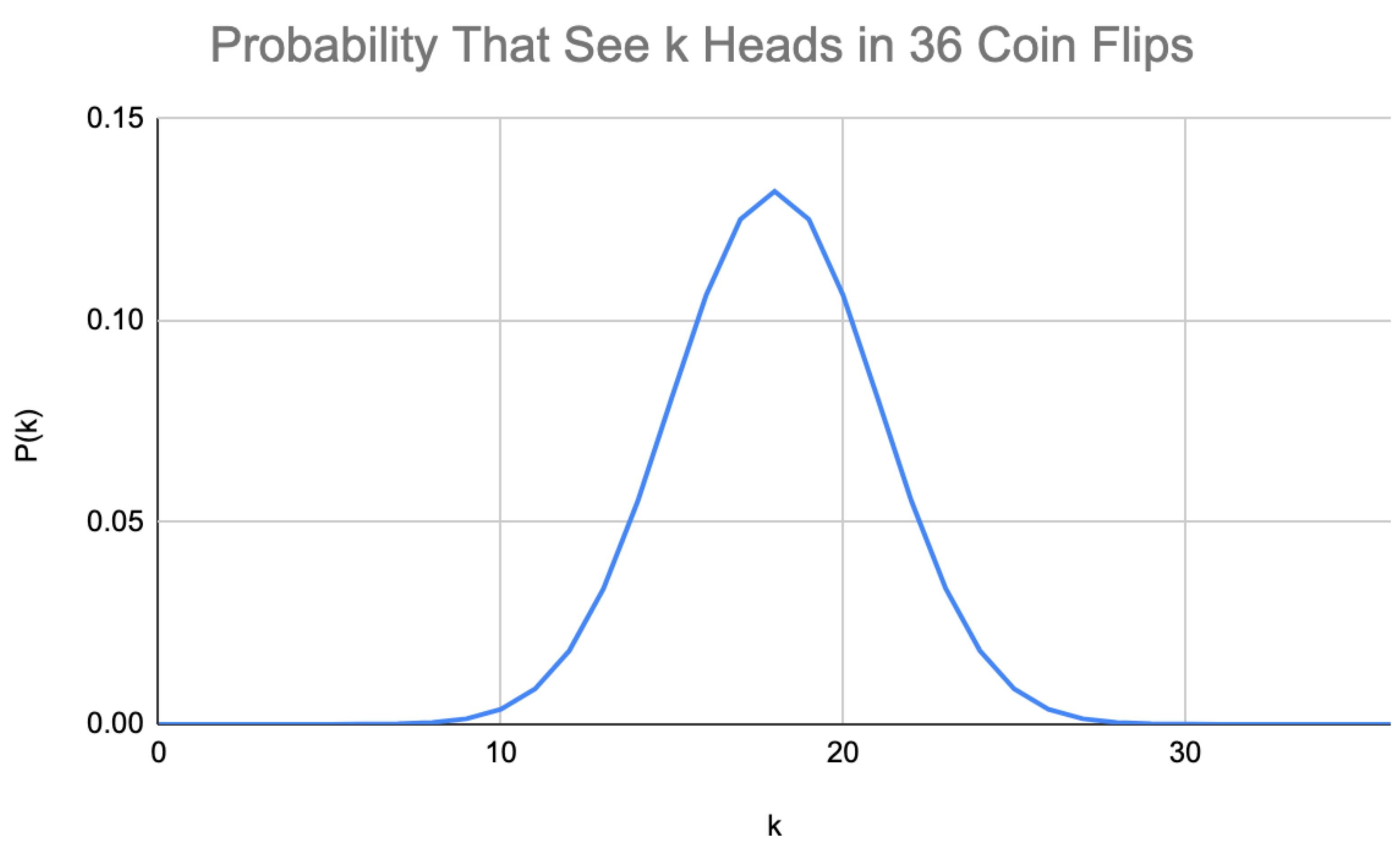
$$= 1/n^d$$

Aside: Coin Flipping and CLT

Let n be the number of coin flips we make, with $p = 1/2$ being the probability of success, and $q = 1/2$ being the probability of failure. Then the:

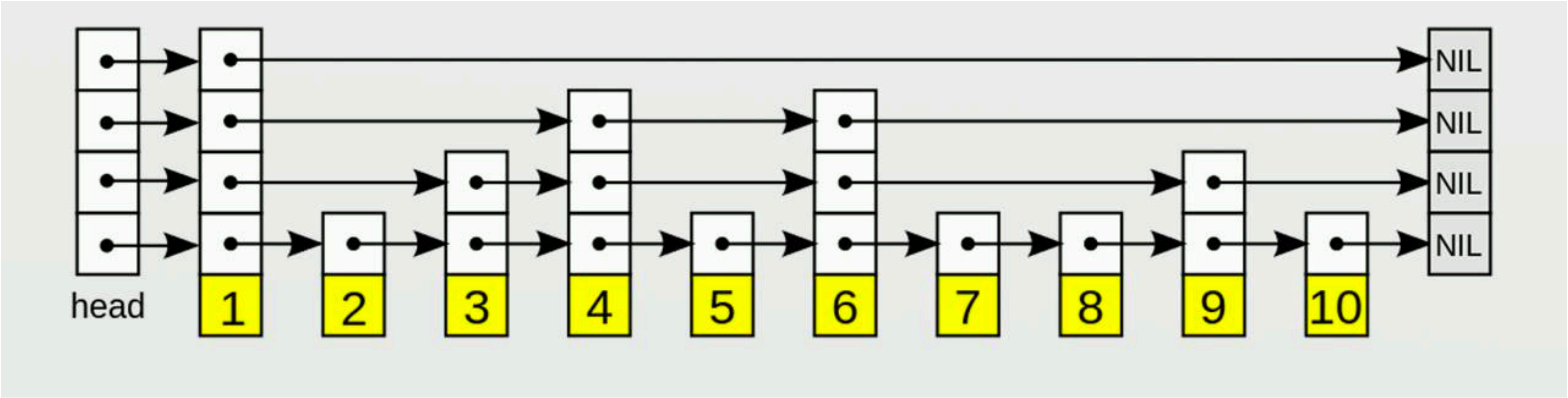
- mean $\mu = np = n/2$, and variance $\sigma^2 = npq = n/4$

- The central limit theorem says that, for a sequence of independent and identically distributed random variables drawn from a distribution with expected value μ and a finite variance σ^2, the sample averages converge to μ as $n \to \infty$.



- Although not a proof, hopefully this helps to further illustrate the unlikelihood of a very tall skiplist!

Skip Lists



- Using $O(\log n)$ linked lists, achieve same performance as binary search tree
- No stored information about balance, no tricky balancing rules!
- Just flip coins when inserting new elements to decide which lists they reside in

Source: Wikipedia

Summary: Skip Lists (Randomized Search Trees)

- Invented around 1990 by Bill Pugh
- Motivation: binary search trees are a pain to implement
- Skip lists balance randomly; no rules to remember, no rebalancing
- Build out of simple structure: sorted linked lists
- Inserts, deletes, search, predecessor, successor are all $O(\log n)$ with high probability
- No rebalancing makes them useful in concurrent programming
    - E.g, lock-free data structures

Acknowledgments

- Some of the material in these slides are taken from
  - Shikha Singh
  - MIT slides: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-12-skip-lists/lec12.pdf
  - Eric Demaine handout: https://courses.csail.mit.edu/6.046/spring04/handouts/skiplists.pdf