



# 3821 – Randomized Algorithms and Structures

## Skip Lists

# Skip Lists

- A relatively recent data structure (1990)
  - A randomized algorithm with benefits of AVL trees
    - $O(\lg n)$  **expected** time for Search, Insert
    - $O(1)$  time for Min, Max, Succ, Pred
  - *Much* easier to code than AVL trees!!

# Linked Lists

---

- Think about a sorted doubly linked list as a structure for dynamic sets. What is the running time of:
  - **Min( ) and Max( )?**
  - **Successor( )?**
  - **Search( )?**
  - **Insert( )?**

# Linked Lists

- Think about a sorted doubly linked list as a structure for dynamic sets. What is the running time of:

- **Min( ) and Max( )?**
- **Successor( )?**

}  $O(1)$

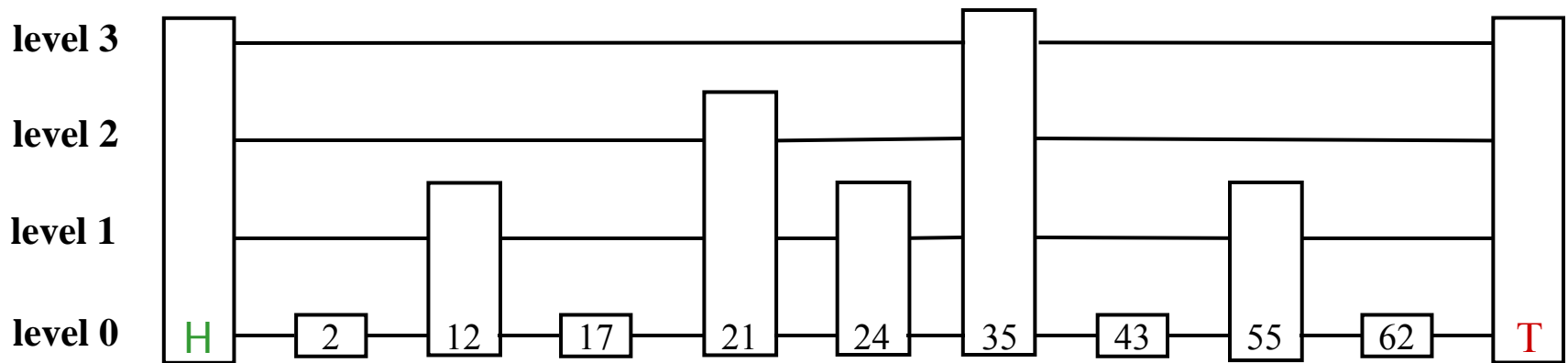
- **Search( )?**
- **Insert( )?**

}  $O(n)$

*Goal: make these  $O(\lg n)$  **expected time** in a linked-list setting!*

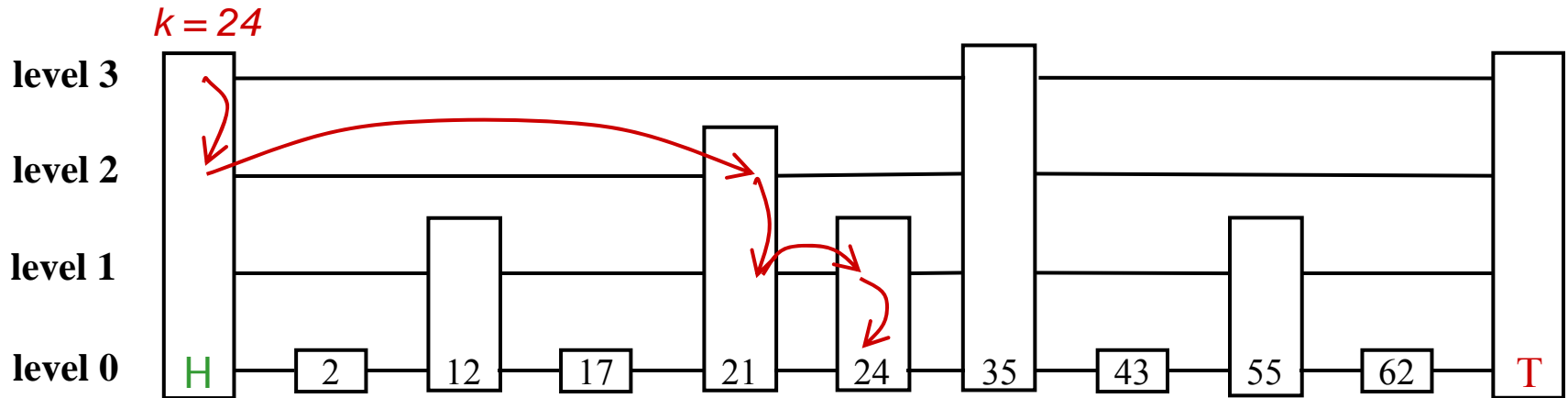
# Skip Lists

- The basic idea: create shortcuts!



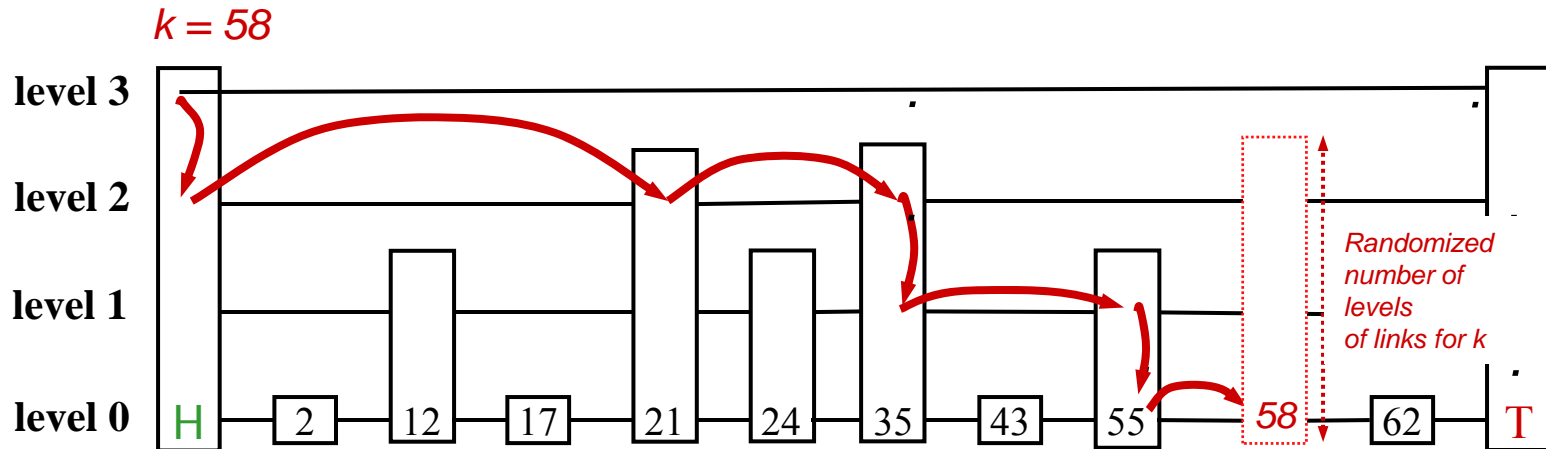
- Keep a linked list of elements with multiple links.

# Skip List Insert



- **To Search for  $k$**  : start from the head **H** and go as far right as you can, without exceeding  $k$ , using the highest possible level of links; then drop one level down and repeat the procedure using lower level links.
- This somewhat like elevators in skyscrapers, except that elevator stops are randomized!

# Skip List Insert and Delete



- **To Insert  $k$** : first search to find the right place. Then toss a coin until you get a head, and count the number of tails that you got. You insert  $k$  and link it at this many levels! (deciding what elevators stop there!)
- **To Delete  $k$** : just as in a standard linked list, but taking care of pointers at all levels affected.

# Skip List probabilistic analysis

- The probability of getting  $i$  consecutive heads when flipping a coin  $i$  times is  $1/2^i$ . Thus, each element has links on level  $i$  with probability  $1/2^i$
- If  $n$  elements belong to a set each with probability  $p$ , the expected size of the set is  $n p$ . Thus, an  $n$  element skip list has about  $n/2^i$  elements with links of level  $i$ .
- Since an element has links up to the order of exactly  $i$  with probability  $1/2^{i+1}$  the total *expected* number of links per element is

$$O\left(\sum_{i=1}^{\infty} \frac{i}{2^{i+1}}\right) = O(1)$$

# Skip List probabilistic analysis

- Since the expected number of elements with links on level  $i$  is  $n/2^i$  and since  $n/2^i > 1$  only for  $i < \lg n$ , only  $i = \lg n$  levels are likely to have at least one element.
- Thus, the expected number of levels of a Skip List with  $n$  elements is  $O(\lg n)$ .
- Moreover, the probability of having an element on level  $i$  is smaller than  $n/2^i$ ; thus, the probability to have an element on level  $2 \lg n$  is less than  $n/2^{2 \lg n} = n/n^2 = 1/n$ .

# Skip List probabilistic analysis

- If **searching**, and at level  $i$  at the moment, then the probability that we cannot move forward on the same level, yet there are many elements on level  $i-1$  to traverse is very small.
- This is because for each element with link on level  $i$  the probability of also having a link at level  $i+1$  is  $1/2$ ; thus the expected number of elements with links at level  $i$  but without links on level  $i+1$  is equal to the number of consecutive coin tosses to get a head, which is 2:

$$1/2 + 2/2^2 + 3/2^3 + \dots = \sum_{i=1}^{\infty} i/2^i = 2$$

- Thus, there are  $\lg n$  levels to go down, with expected two links per level to go to the right; altogether  $O(\lg n)$  many steps to complete the search. Similar estimate holds for deletions.

# Skip List summary

- A skip list is an efficient randomized data structure that is an alternative to binary search trees (BST);
- In a skip list with  $n$  entries
  - The expected space used is  $O(n)$
  - The expected search, insertion and deletion time is  $O(\log n)$  with substantially smaller constants than for BST;
  - The worst case search, insertion and deletion time is  $O(n)$ , but with rapidly decreasing probability ( $< 1/n^k$ ) that such time is larger than  $k \lg n$ .
- Thus, unless we must provide absolute worst case performance guarantee, the skip lists are simpler and on average faster data structure than self balanced binary search trees.