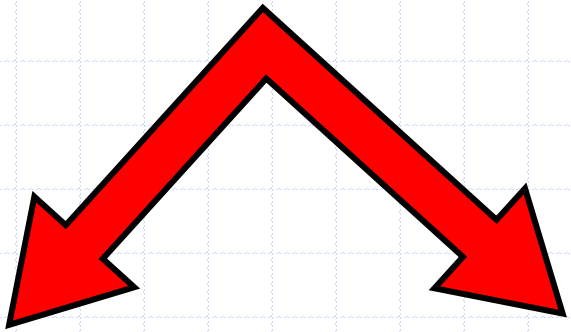
Advanced Data Structures and Algorithms

Associate Professor Dr. Raed Ibraheem Hamed

Computer Science Department
College of Science and Technology
University of Human Development

2015 – 2016

Randomized Data Structures

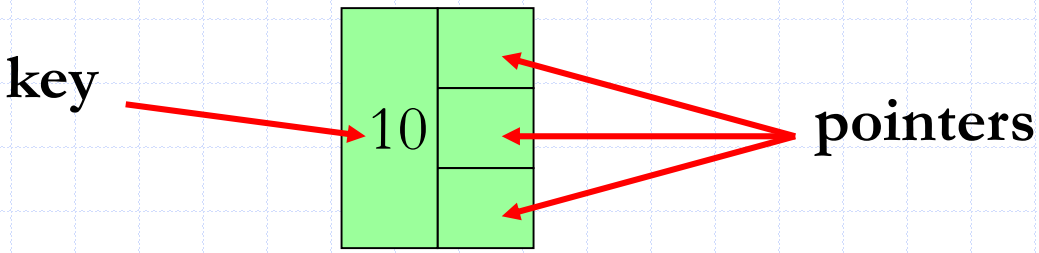


Treaps

Randomized skip lists

What is it?

In computer science, a **skip list** is a data structure that allows **fast search** within an ordered sequence of elements. Fast search is made possible by maintaining a linked hierarchy of subsequences, each skipping over fewer elements.



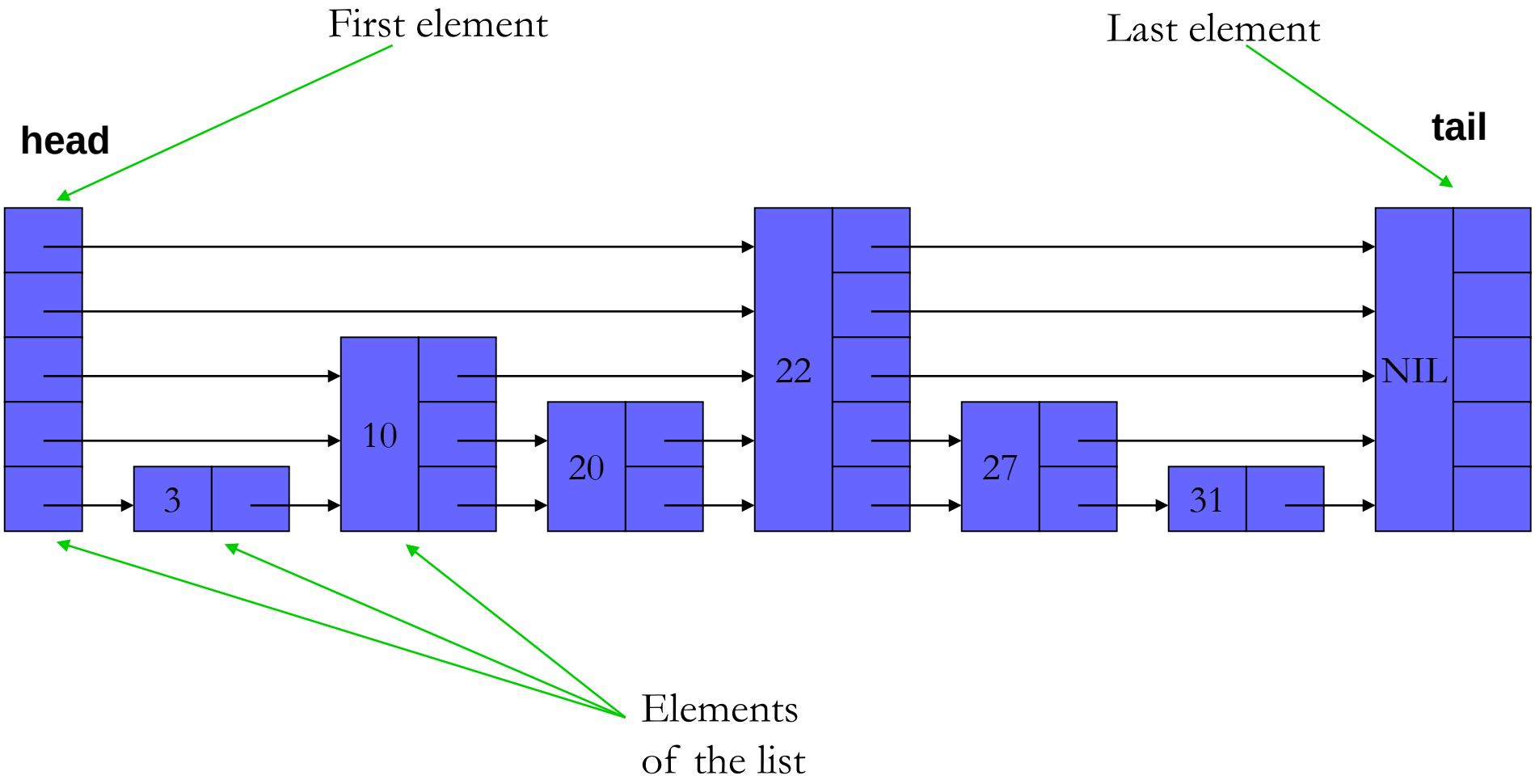
What is it?

- A Skip List is a Data Structure based on parallel linked lists "discovered" by William Pugh in 1990.
- Skip Lists support O(n) time for
  - Insertion
  - Deletion
  - Querying
- At a high level, a skip list is just a **sorted**, singly linked list with some "shortcuts" (additional pointers that allow to **travel faster**.

Skip lists

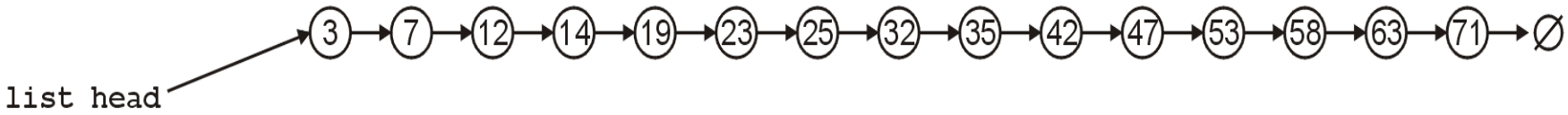
- A linked list has desirable insertion and deletion characteristics if we have a pointer to one of the nodes
  - Searching a linked list with n elements, however is O(n)
  - Can we achieve some of the characteristics of an **array** for searching a specific element ? To achieve this in a **linked list**, we require a **pointer** to the central element

What is it?


First element
Last element
head
tail
3
10
20
22
27
31
NIL
Elements
of the list

Skip lists

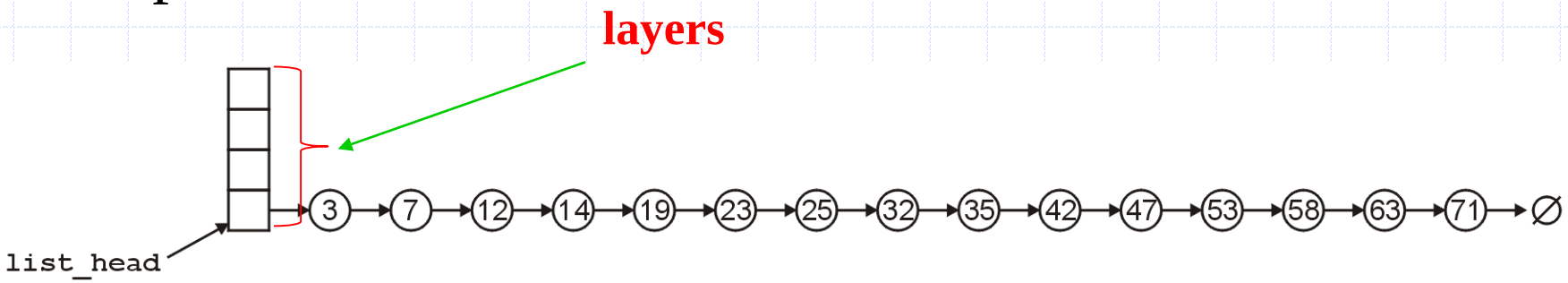
- What if we could skip over many elements at a time?
- Consider the following linked list...



Skip lists

- Start with a sorted linked list
  - Add another **layer** linking every other element
  - Repeat for that layer, etc
- Think of as a hierarchy of sorted linked lists

Suppose, however, if we had an array of head pointers

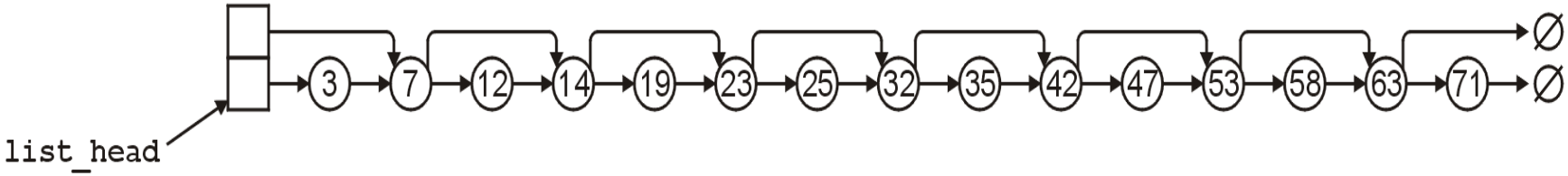


Skip list runtime

- How high does this stack (layer) go?
  - Level 1: n
  - Level 2: n/2
  - Level 3: n/4
- So search through the skip list will be O(n)
- But can we maintain this guarantee of efficiency?

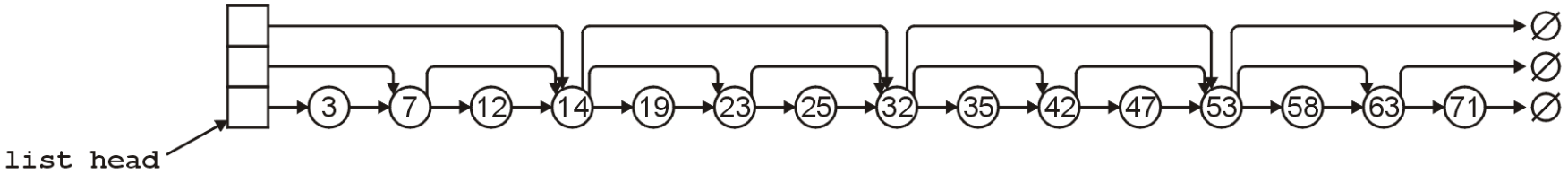
Skip lists

First, we could point to every second node



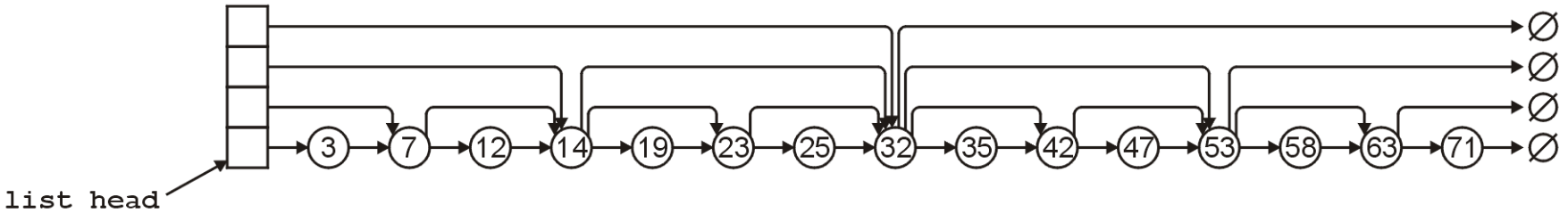
Skip lists

We continue by pointing to every 4^{th} entry



Skip lists

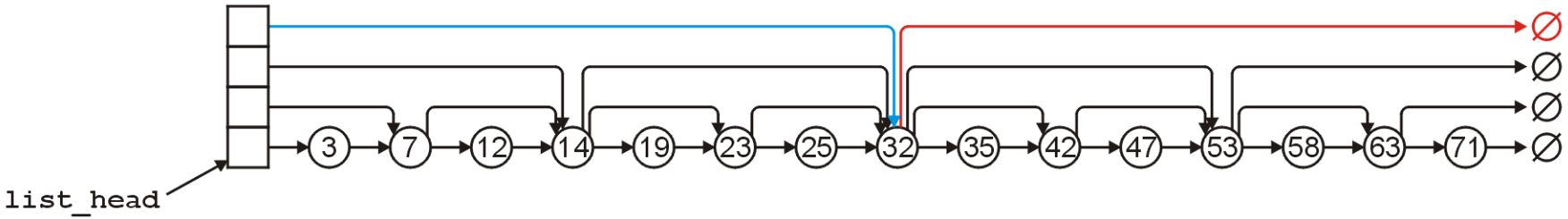
And then every 8th entry, and so on...



Skip lists

Suppose we search for 47

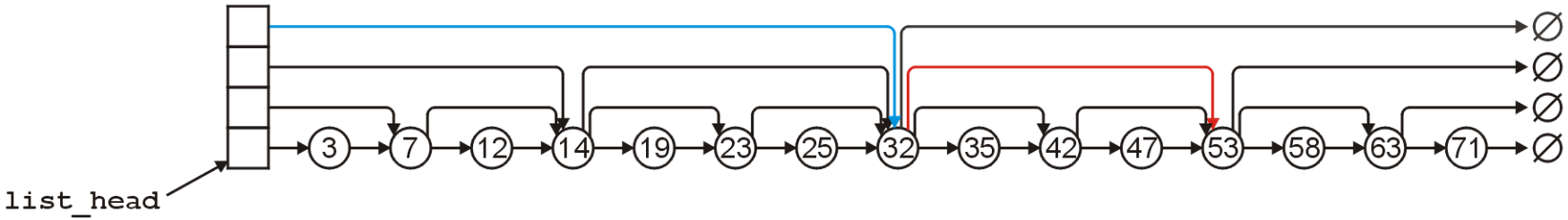
- Following the 4th-level pointer, first 32 < 47 but the next pointer is 0



Skip lists

Suppose we search for 47

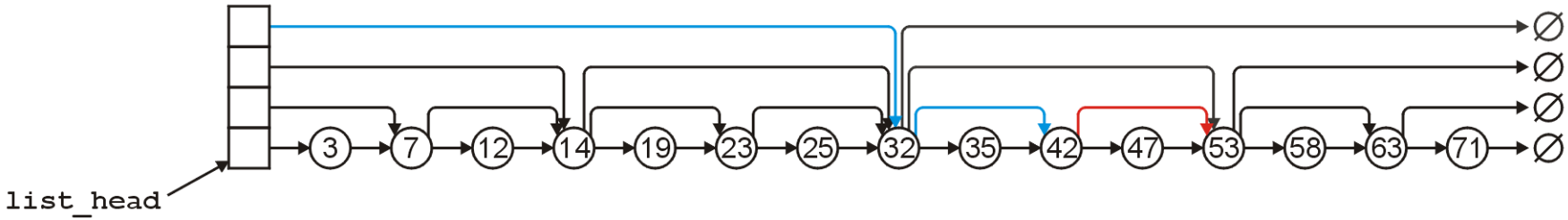
- Continuing with the 3rd-level pointer from 32, 53 > 47



Skip lists

Suppose we search for 47

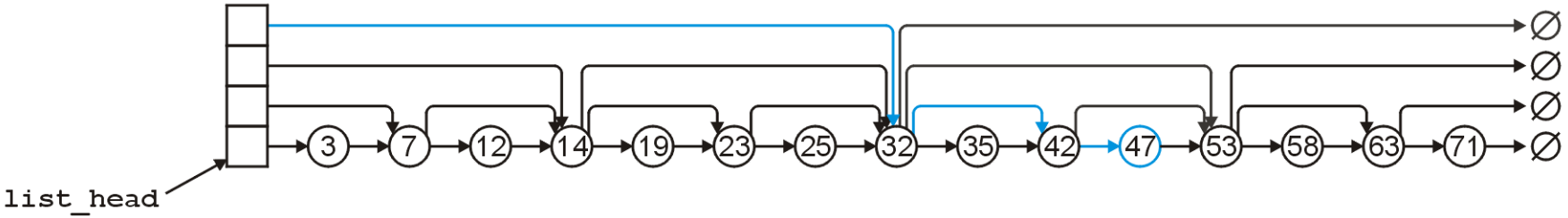
- Continuing with the 2nd-level pointer from 32, 47 > 42 but 53 < 47



Skip lists

Suppose we search for 47

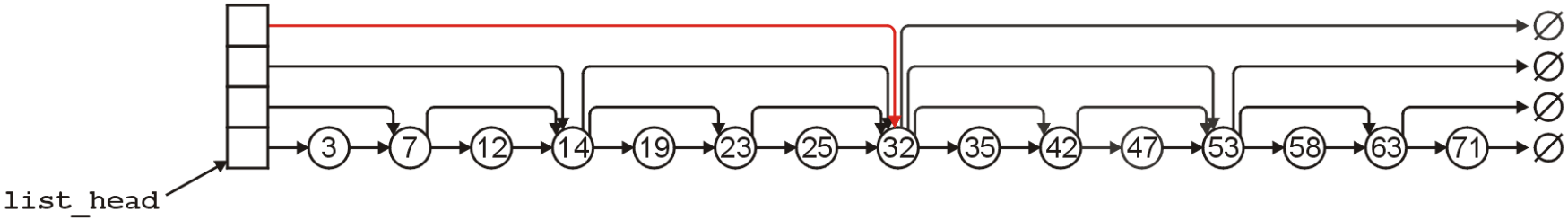
- Continuing with the 1st-level pointer from 42, we find **47**



Skip lists

Suppose we search for 24

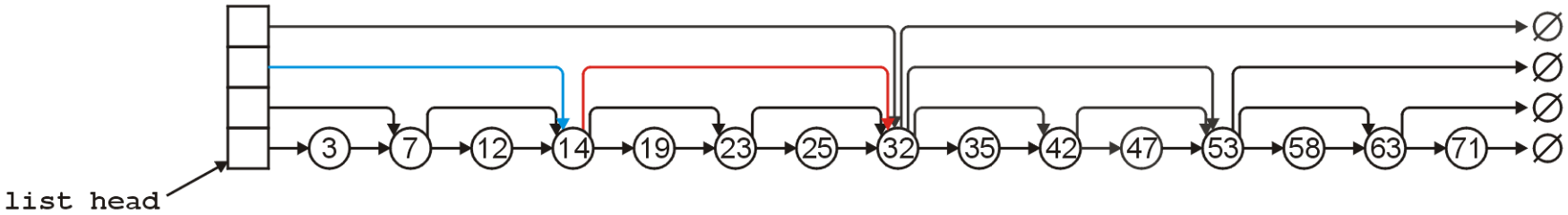
- With the 4th-level pointer, 32 > 24



Skip lists

Suppose we search for 24

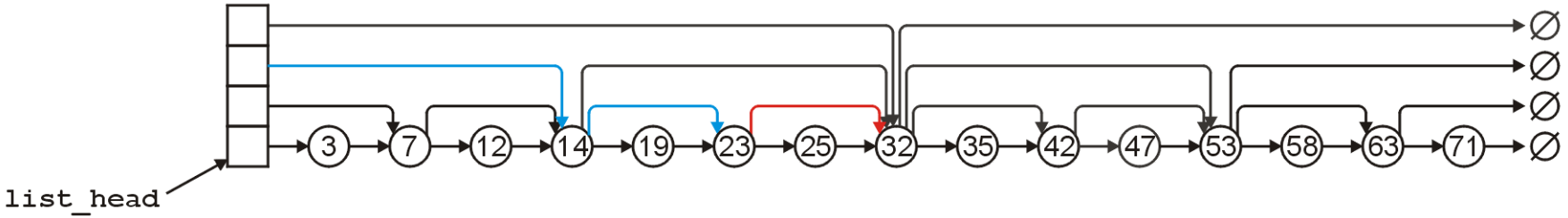
- Following the 3rd-level pointers, 14 < 24 but 32 > 24



Skip lists

Suppose we search for 24

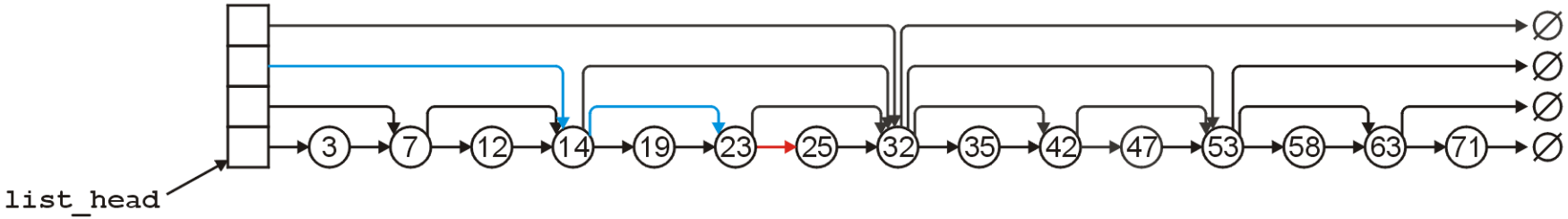
- Following the 2nd-level pointer from 14, 23 < 24 but 32 > 24



Skip lists

Suppose we search for 24

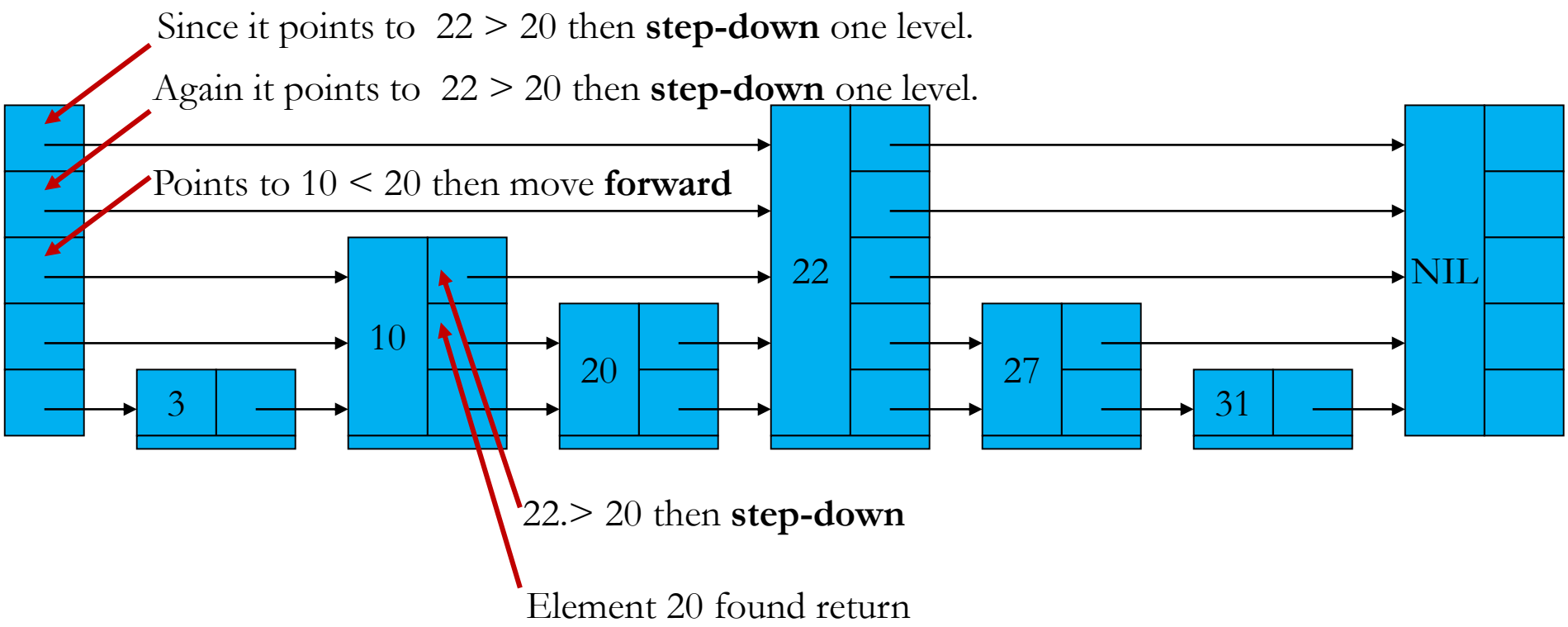
- Following the 1st-level pointer from 23, 25 > 24
- **Thus, 24 is not in the skip list**



SEARCH

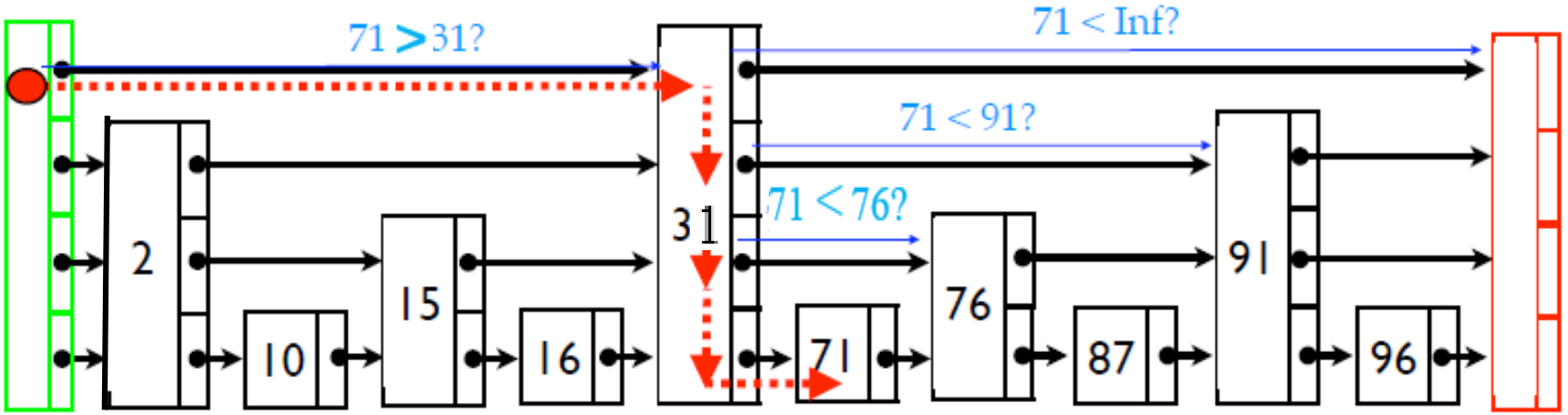
Example: search the object which key is 20

Start with the top pointer of the head



SEARCH

Find 71



When search for k:

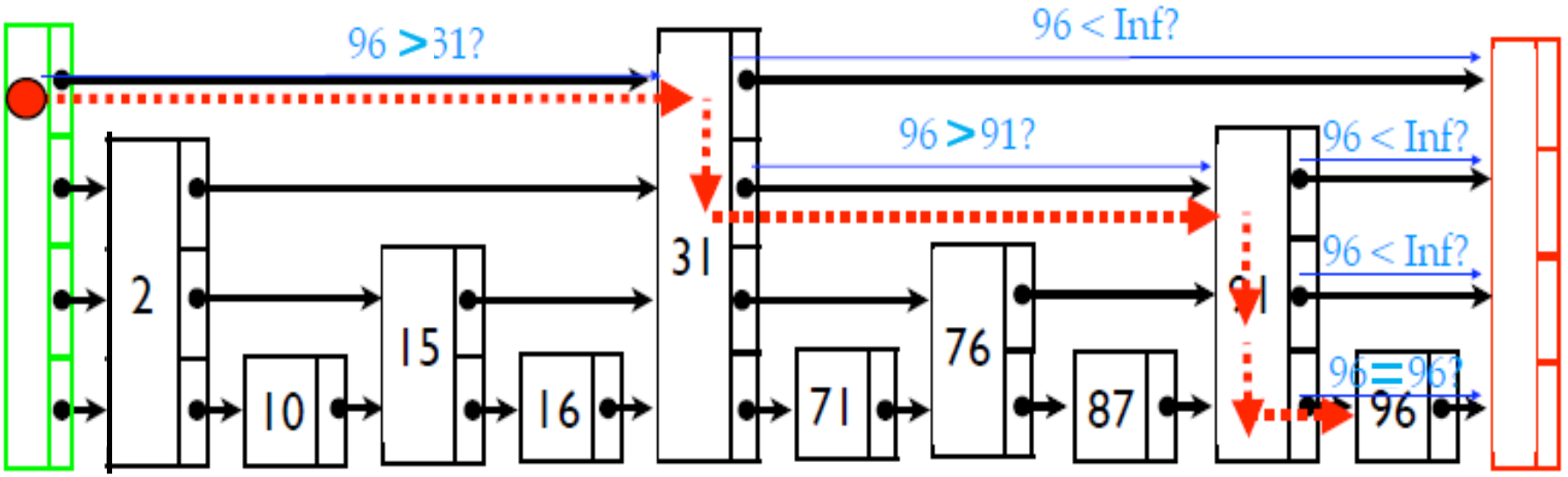
If k = key, done!

If k < next key, go down a level

If k ≥ next key, go right

SEARCH

Find 96



When search for k:

If k = key, done!

If k < next key, go down a level

If k ≥ next key, go right

UHD
UNIVERSITY OF HUMAN DEVELOPMENT

Thank you ????