

# CS350: Data Structures

## Skip Lists

---

James Moscola

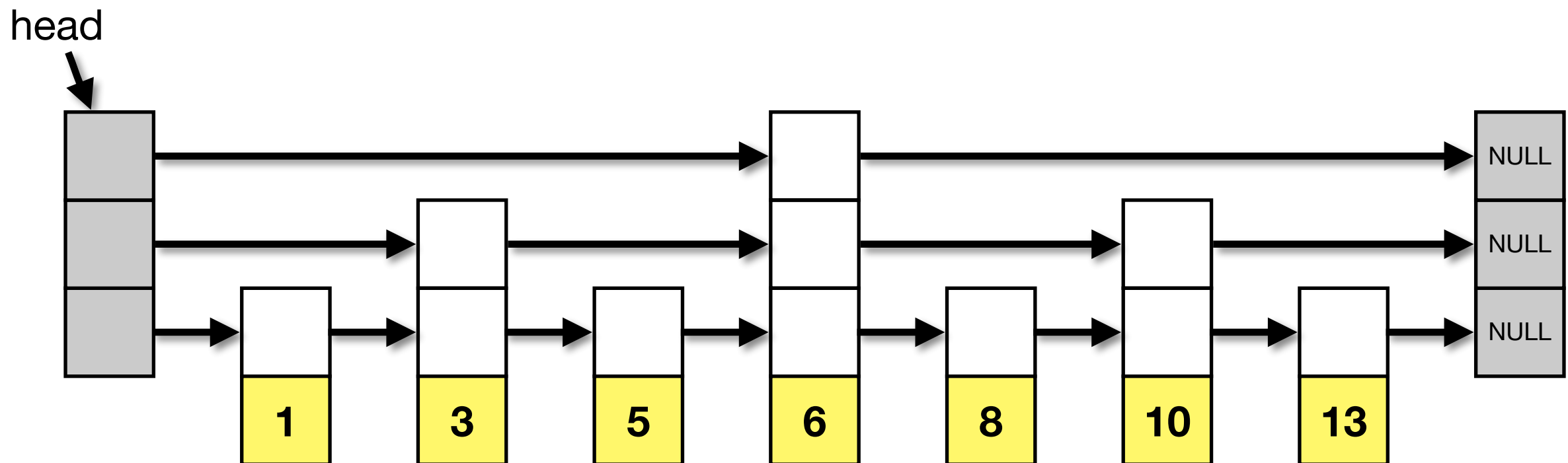
Department of Physical Sciences

York College of Pennsylvania



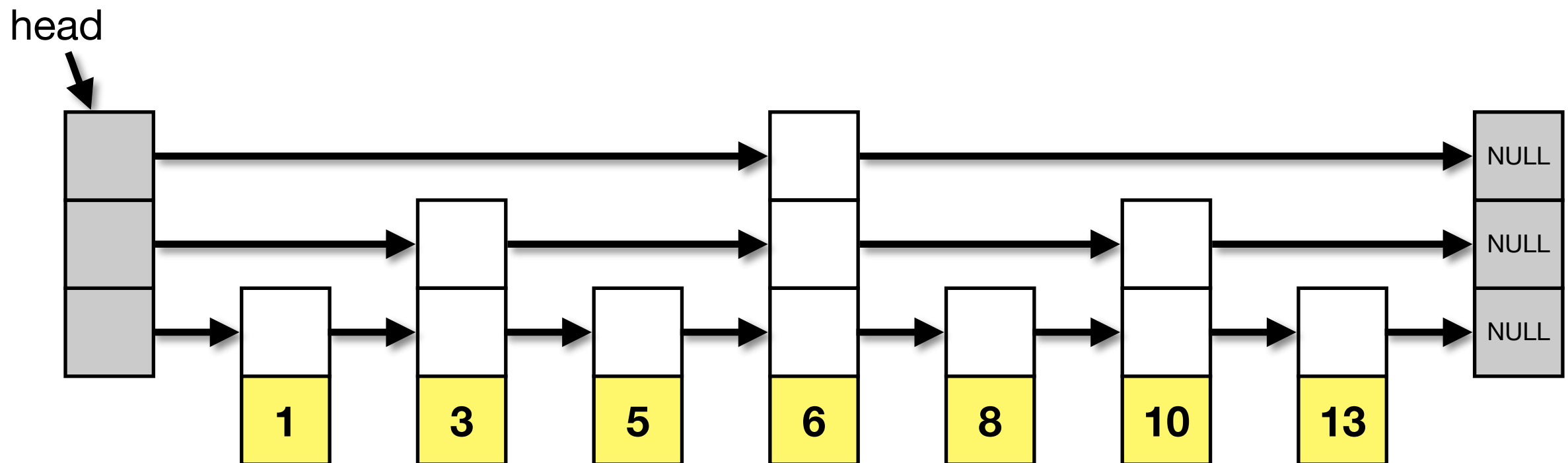
# Skip List Introduction

- A data structure used for storing a sorted list of elements using layers of linked lists
  - Bottom layer is a standard, ordered linked list
  - Upper layer linked lists create 'shortcuts' or 'fast lanes' from one location of the list to another
- Higher levels traverse greater portions of the list than lower levels



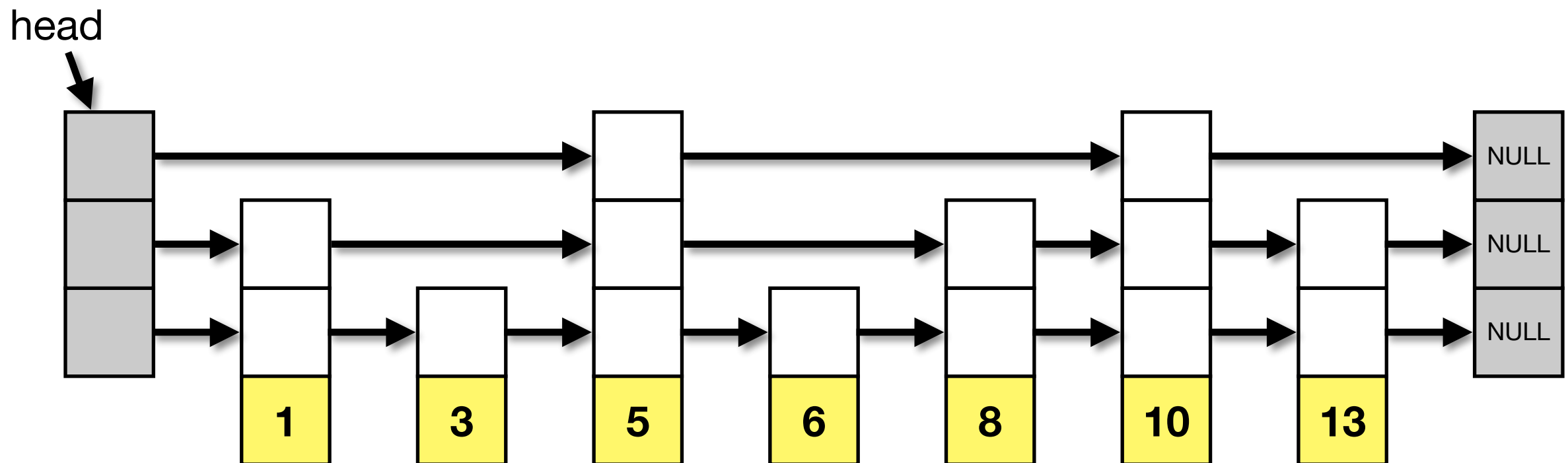
# Skip List Introduction

- The example skip list below shows the skip list divided perfectly in halves, quarters, etc.



# Skip List Introduction

- **In practice, skip list node heights are distributed randomly throughout the skip list**
  - Helps us avoid the need to 'rebalance' the list when insertions or deletions occur



# Skip Lists

---

- Operations on **skip lists** are comparable in efficiency to balanced binary search trees (e.g. AVL trees, Red-Black trees)
  - Insertion, Deletion, and Search operations all run in  $O(\log N)$  time

# Skip List Search

# Skip List Searching

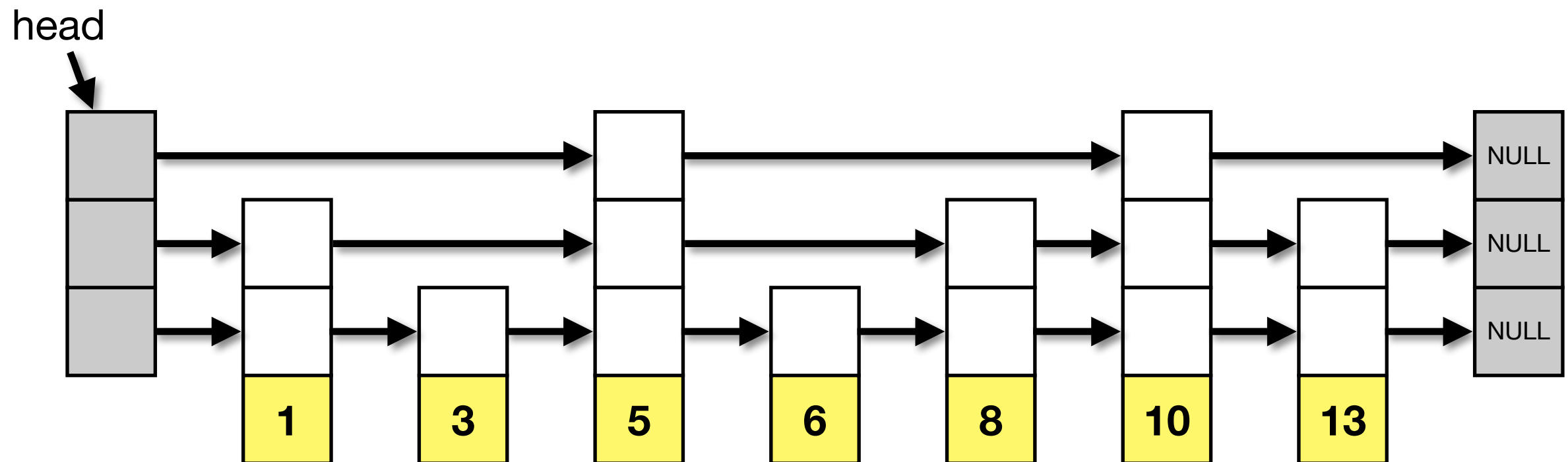
---

- **To search for an element in a skip list:**

- Start at the highest level at the head of the skip list and begin moving horizontally through the list
  - If the next node in the list has the desired data then done
  - If the data in the next node is less than the desired value, move to the next node and continue the search
  - If the data in the next node is greater than the desired value, drop down one level and continue the search

# Skip List Searching

- **Example -- find the node with a key value 6**



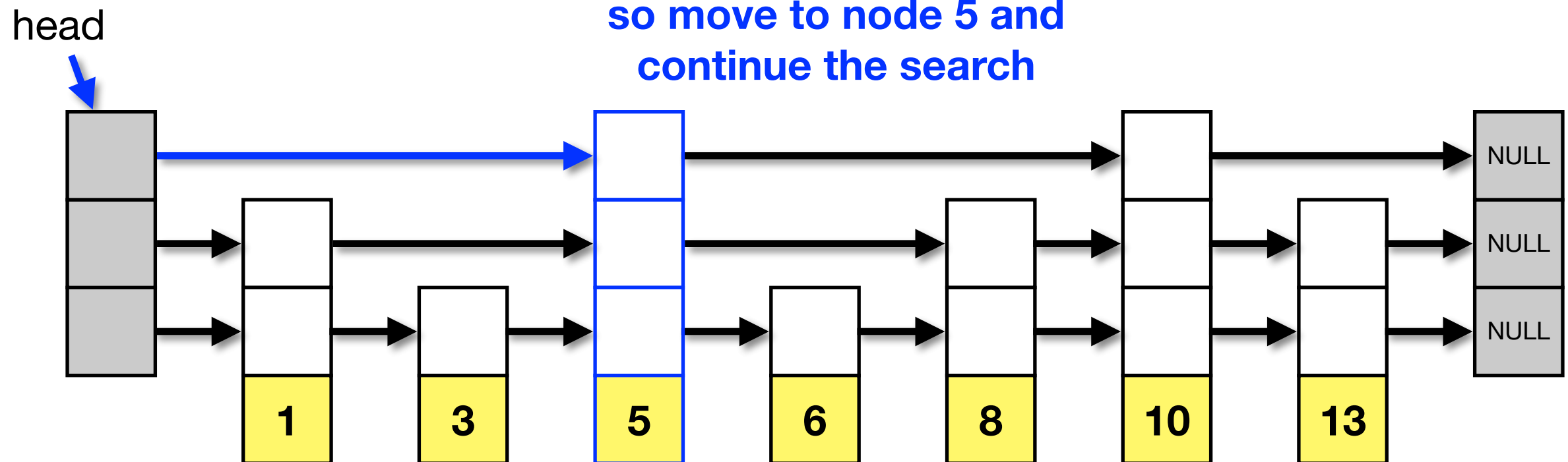


# Skip List Searching

- **Example -- find the node with a key value 6**

(1) Start at highest level of the head node and check data value at node.next

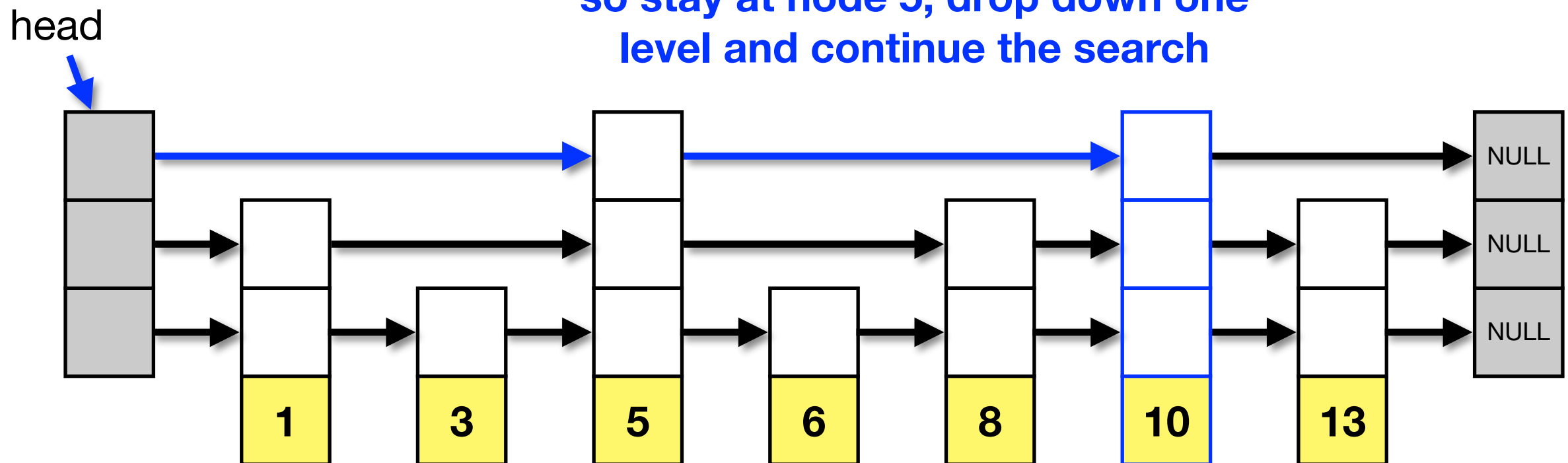
(2) Compare node.next.data to the desired value,  
 $5 < 6$   
so move to node 5 and continue the search



# Skip List Searching

- **Example -- find the node with a key value 6**

(3) Compare node.next.data to the desired value,  
 $10 > 6$   
so stay at node 5, drop down one level and continue the search



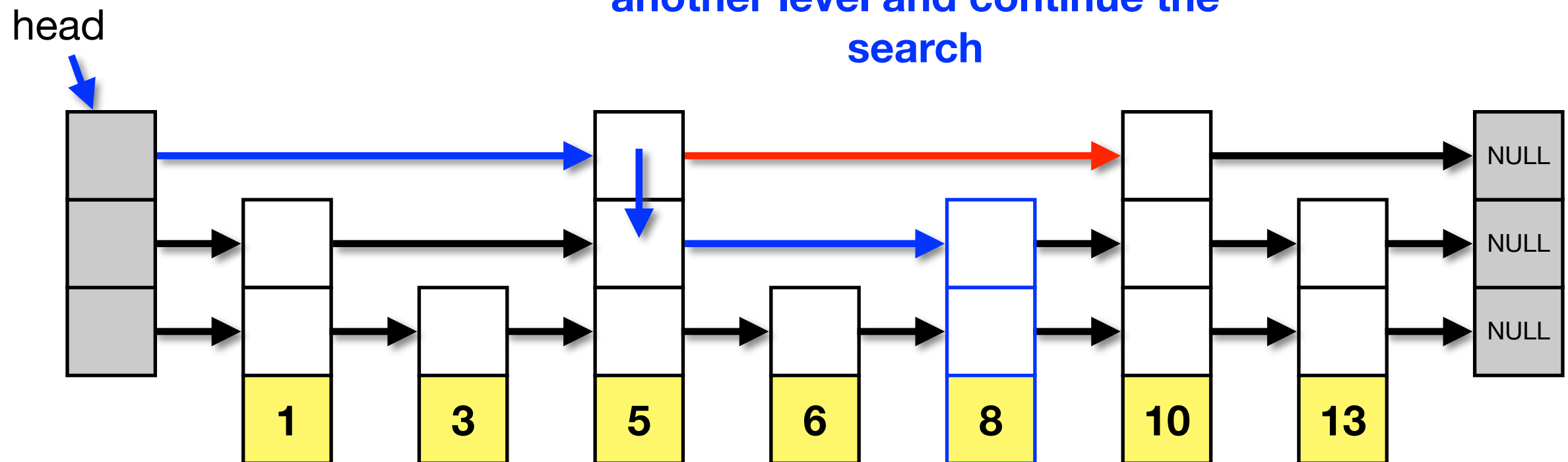
# Skip List Searching

- **Example -- find the node with a key value 6**

**(4) Compare node.next.data to the desired value,**

**8 > 6**

**so stay at node 5, drop down  
another level and continue the  
search**

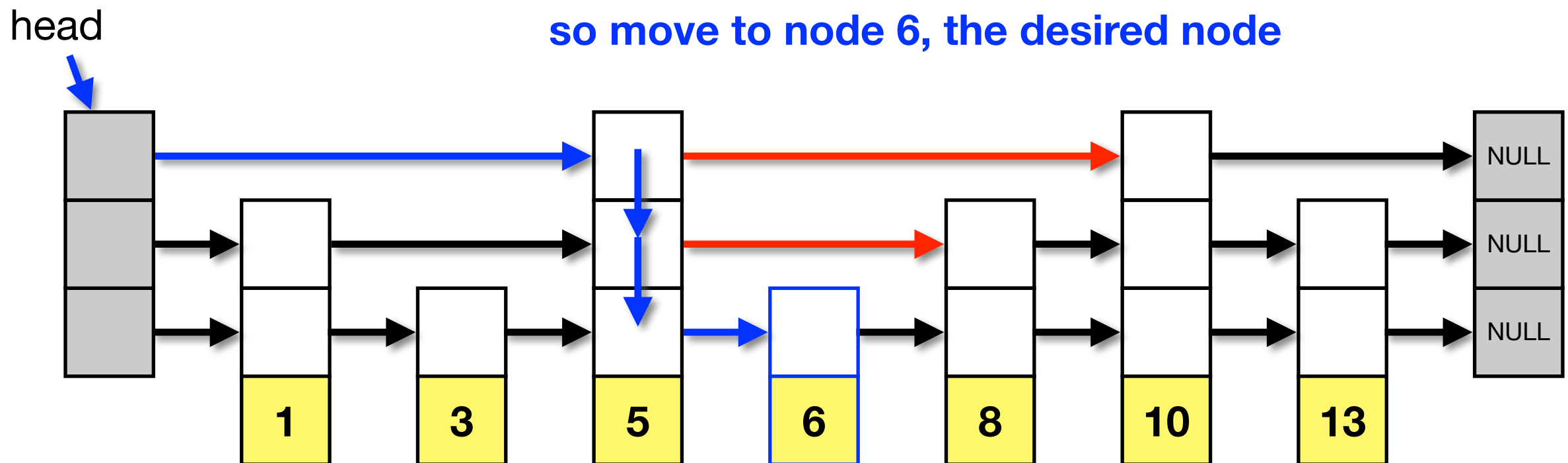


# Skip List Searching

- **Example -- find the node with a key value 6**

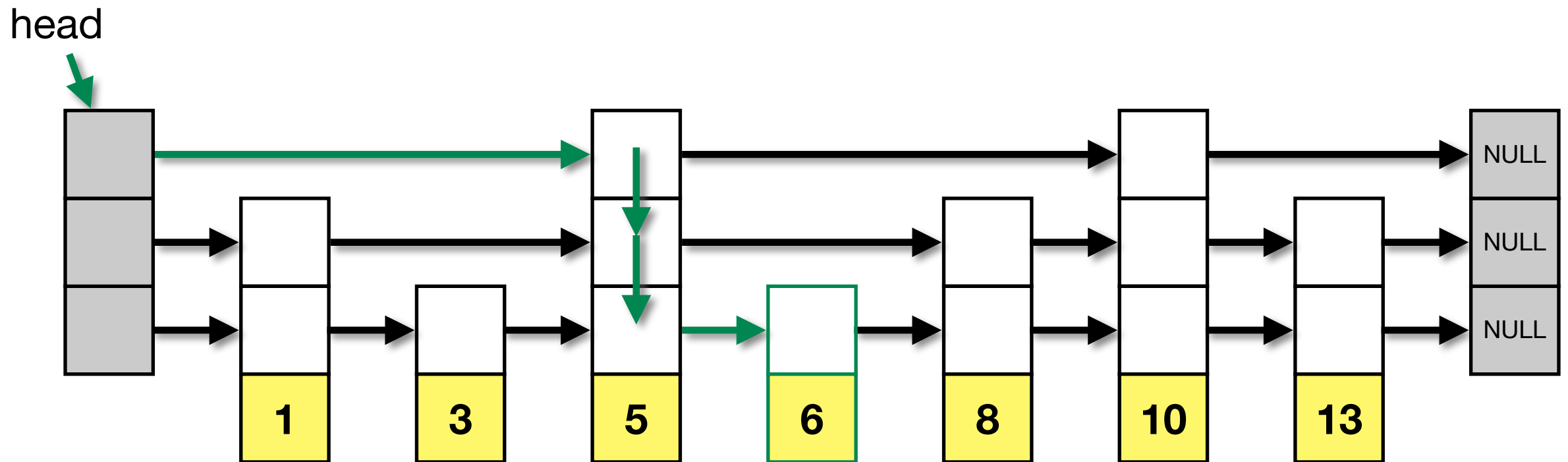
(5) Compare node.next.data to the  
desired value,  
 $6 == 6$

so move to node 6, the desired node



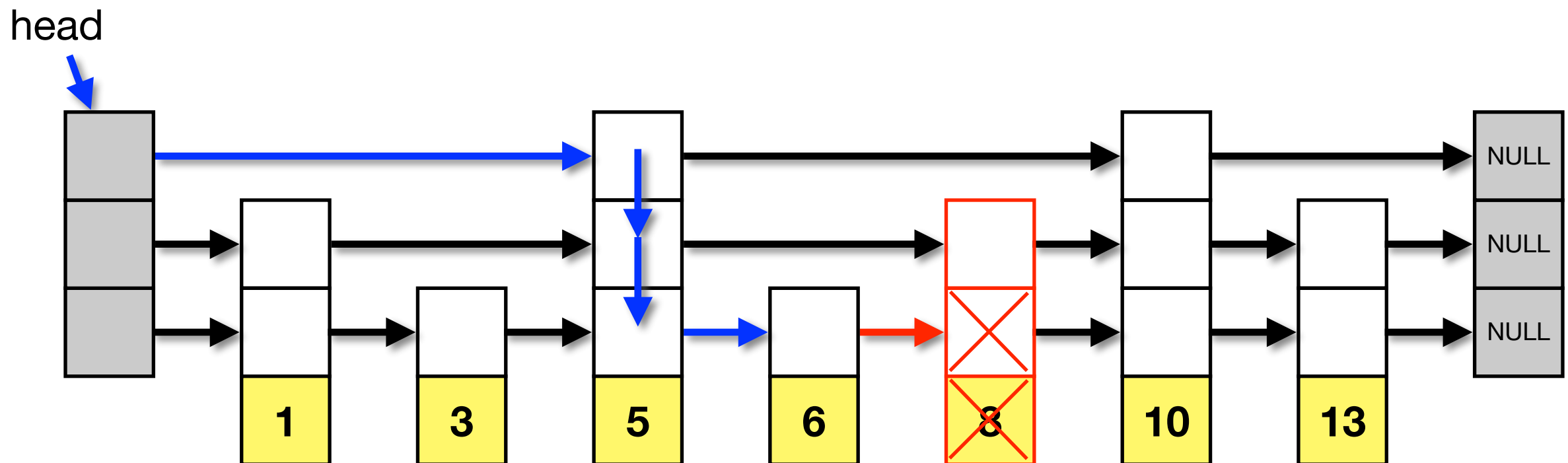
# Skip List Searching

- **Example -- find the node with a key value 6**



# Skip List Searching

- **Example -- find the node with a key value 7**
  - In this example, the desired element does not exist in the list
  - Search the list just as before, but when a value is found that exceed that desired value, AND there are no more levels to drop, then the desired element doesn't exist

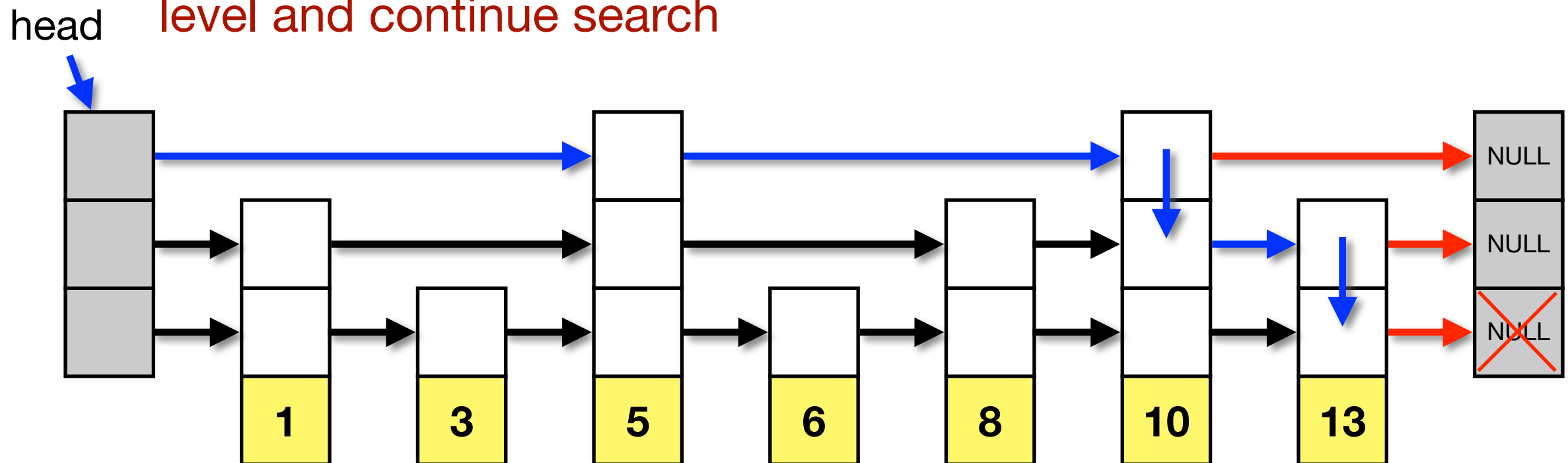


# Skip List Searching

- **Example -- find the node with a key value 15**

- Another example where the desired element does not exist in the list
- Start with a normal search, if search reaches bottom-most level and next node is null, then element does not exist in list

- If not at bottom-most level and `node.next == NULL`, then drop a level and continue search



# Skip List Insertion



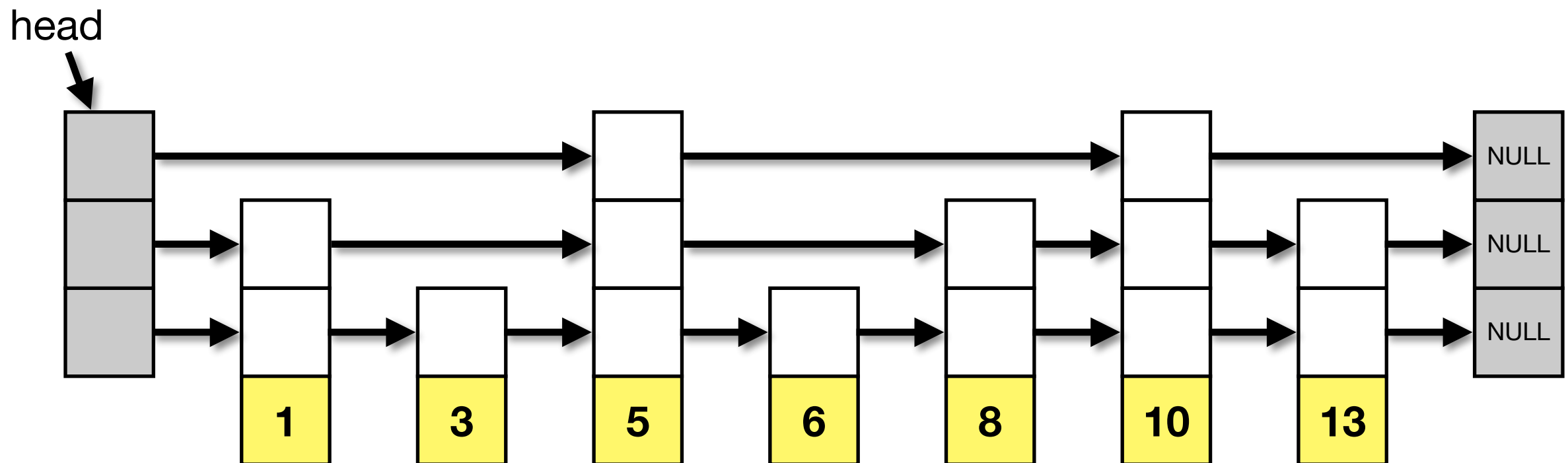
# Skip List Insertion

---

- **When inserting a node into a skip list, first perform a search to determine where the new node must be inserted**
- **The height of a newly inserted node is chosen at random**
  - Flip a coin, if heads, then add an additional level to the newly inserted node, if tails then don't add anymore levels
    - Probability of  $1/2$  that newly inserted node will only be at the bottom-most level
    - Probability of  $1/2$  that newly inserted node will have one additional level
    - Probability of  $1/4$  that newly inserted node will have two additional levels
    - Probability of  $1/8$  that newly inserted node will have three additional levels
    - etc.
- **During the insert procedure, maintain an array that remembers the rightmost node that is to the left of the current location for each level**

# Skip List Insertion

- **Example -- insert the node with a key value 7**
- **Start with a search to determine where the node should be inserted**



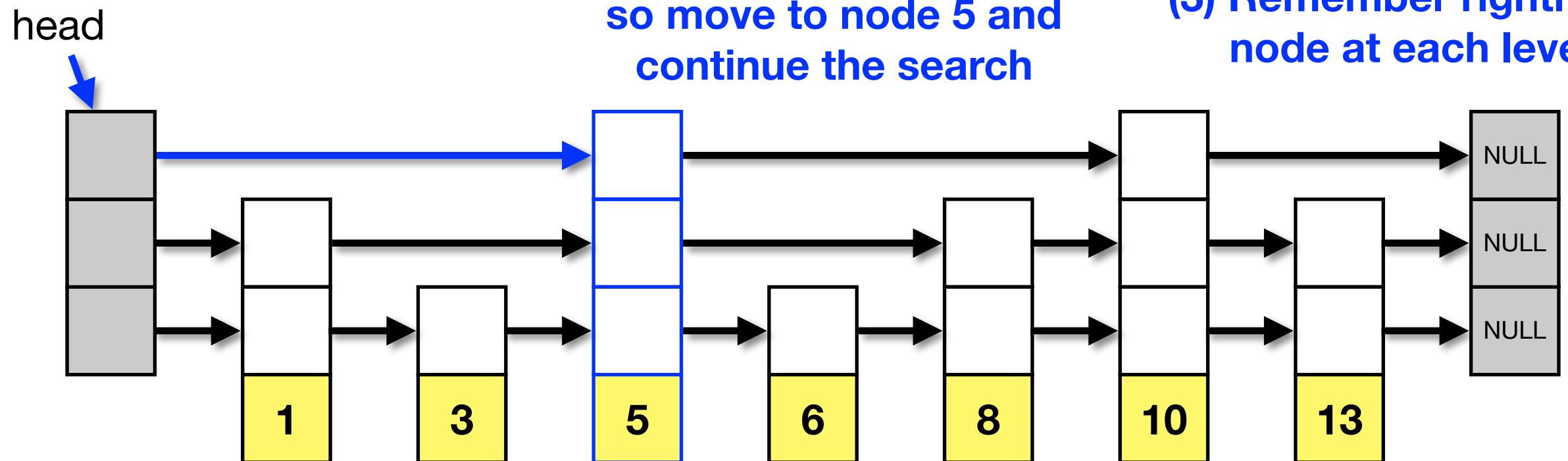
# Skip List Insertion

- **Example -- insert the node with a key value 7**

**(1) Start at highest level of the head node and check data value at node.next**

**(2) Compare node.next.data  
to the new value,  
 $5 < 7$   
so move to node 5 and  
continue the search**

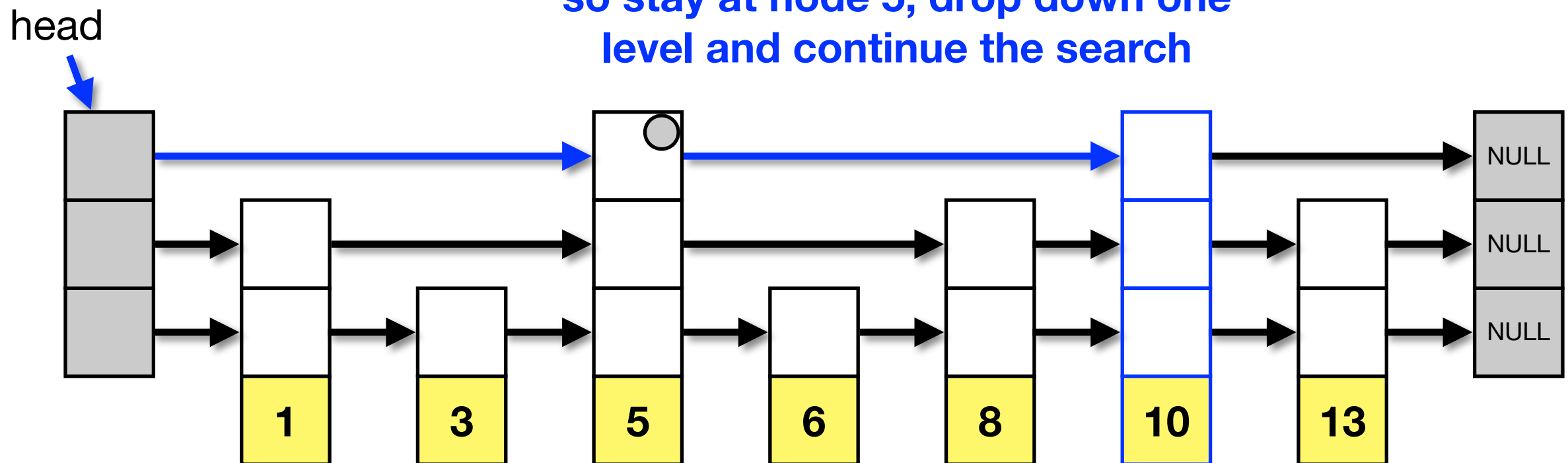
### (3) Remember rightmost node at each level



# Skip List Insertion

- **Example -- insert the node with a key value 7**

(4) Compare node.next.data to the new value,  
 $10 > 7$   
so stay at node 5, drop down one level and continue the search



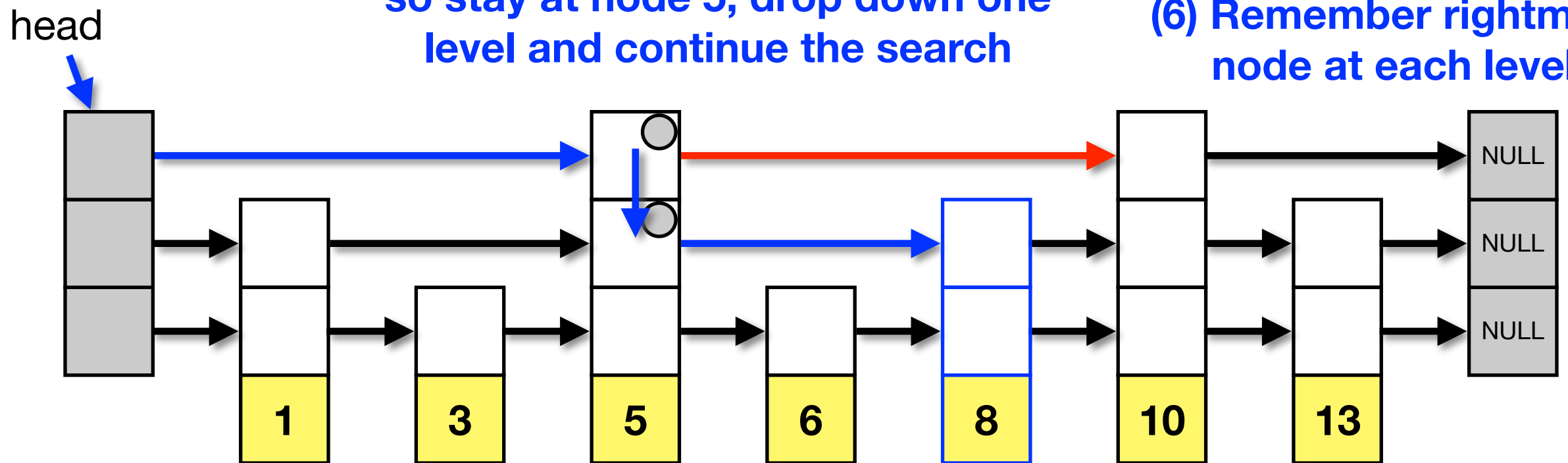
# Skip List Insertion

- **Example -- insert the node with a key value 7**

**(5) Compare node.next.data to the new value,  
8 > 7**

**so stay at node 5, drop down one level and continue the search**

## (6) Remember rightmost node at each level



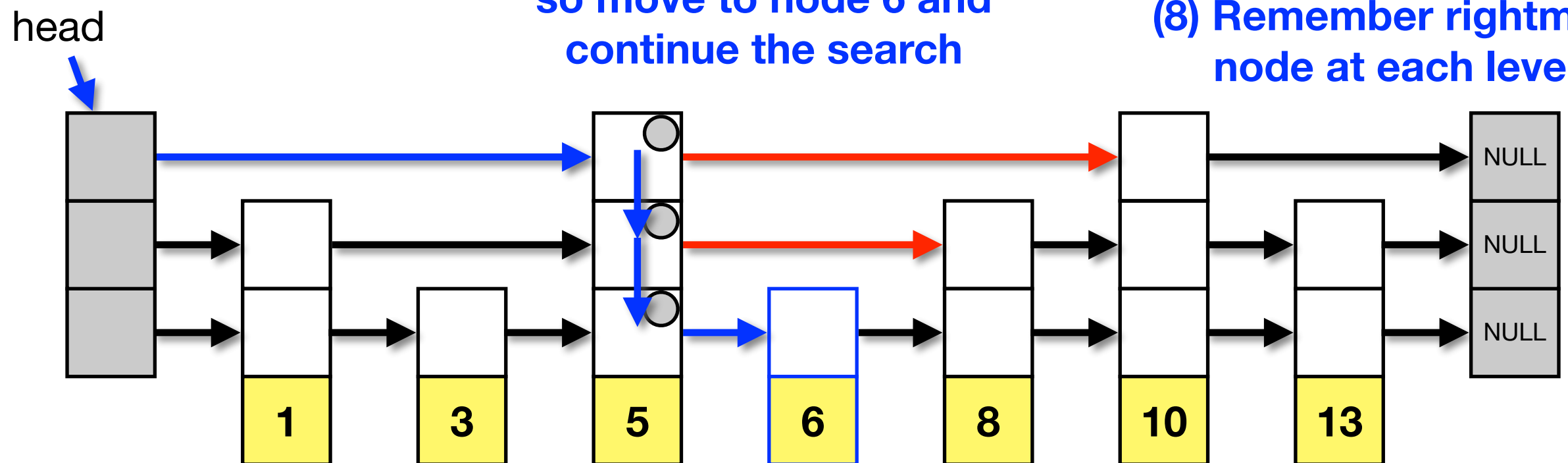
# Skip List Insertion

- **Example -- insert the node with a key value 7**

(7) Compare node.next.data  
to the new value,  
 $6 < 7$

so move to node 6 and  
continue the search

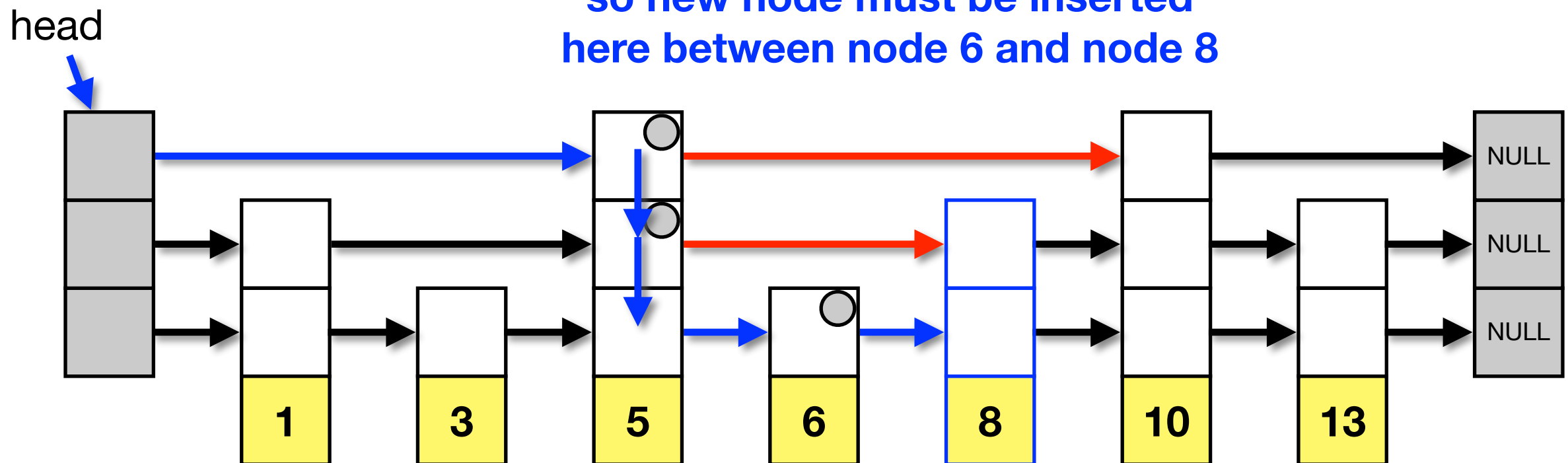
(8) Remember rightmost  
node at each level



# Skip List Insertion

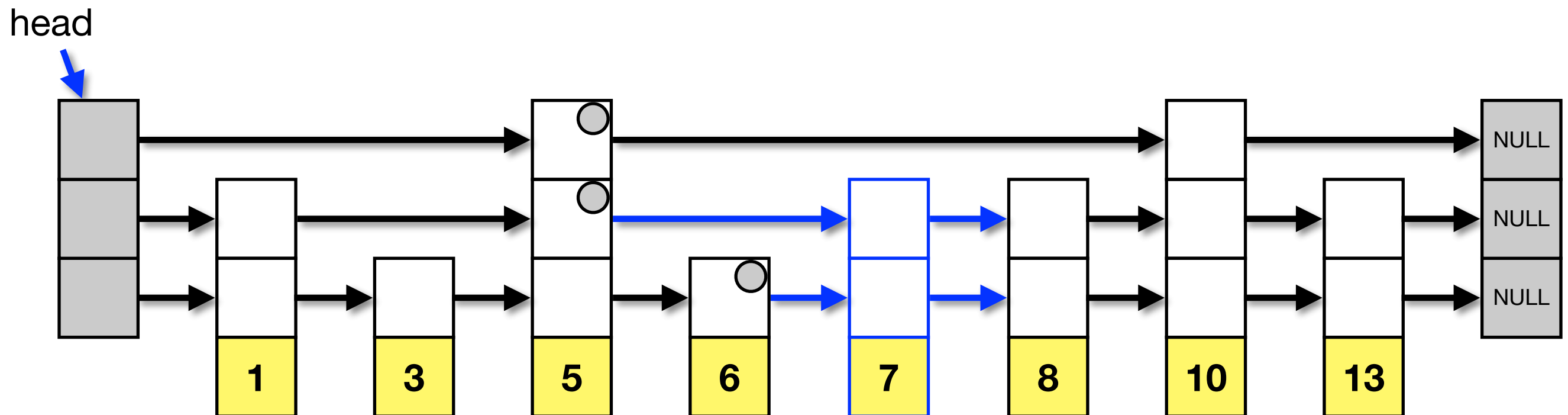
- **Example -- insert the node with a key value 7**

(9) Compare node.next.data to the new value,  
 $8 > 7$   
so new node must be inserted here between node 6 and node 8



# Skip List Insertion

- **Example -- insert the node with a key value 7**
  - To insert node 7:
    - Randomly pick height for node 7
    - Markers were left behind to remember the rightmost node at each level to the left of the insertion location ... use them to update next pointers

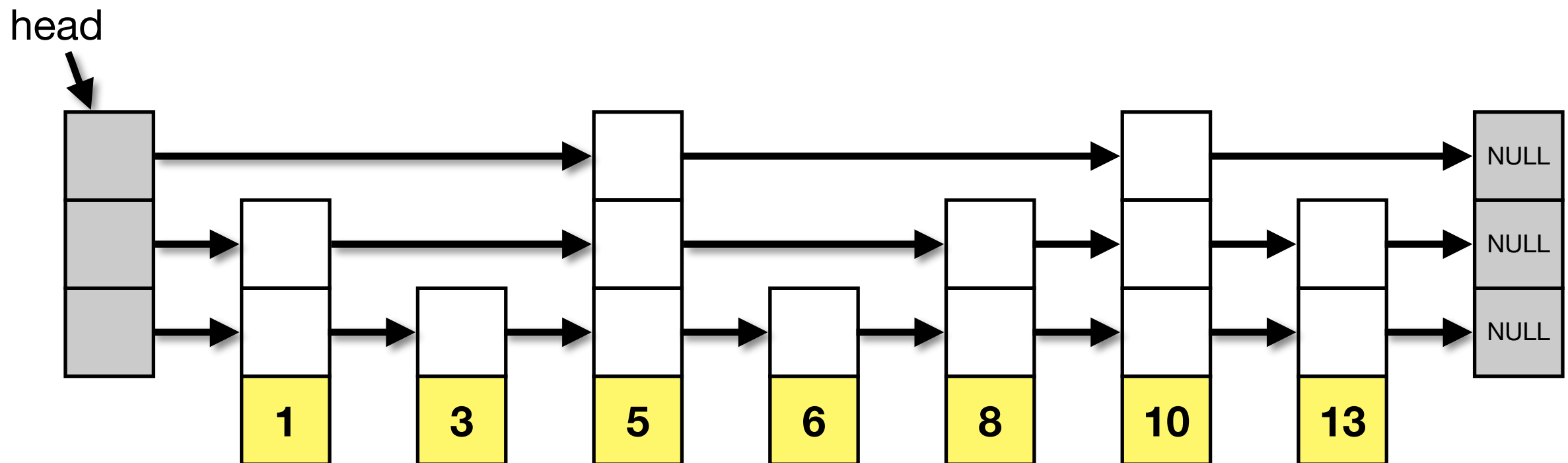




# Skip List Deletion

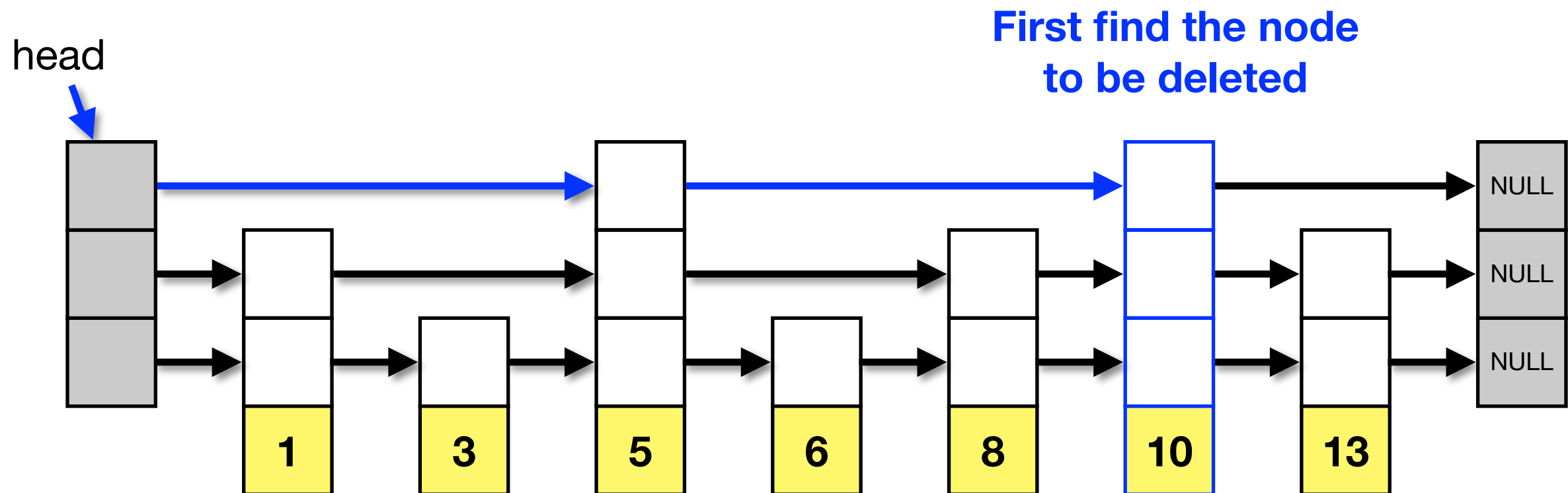
# Skip List Deletion

- For each level of which a skip list node is a part, splice the node out in the same fashion as a standard linked list
- Example -- delete the node with a key value 10



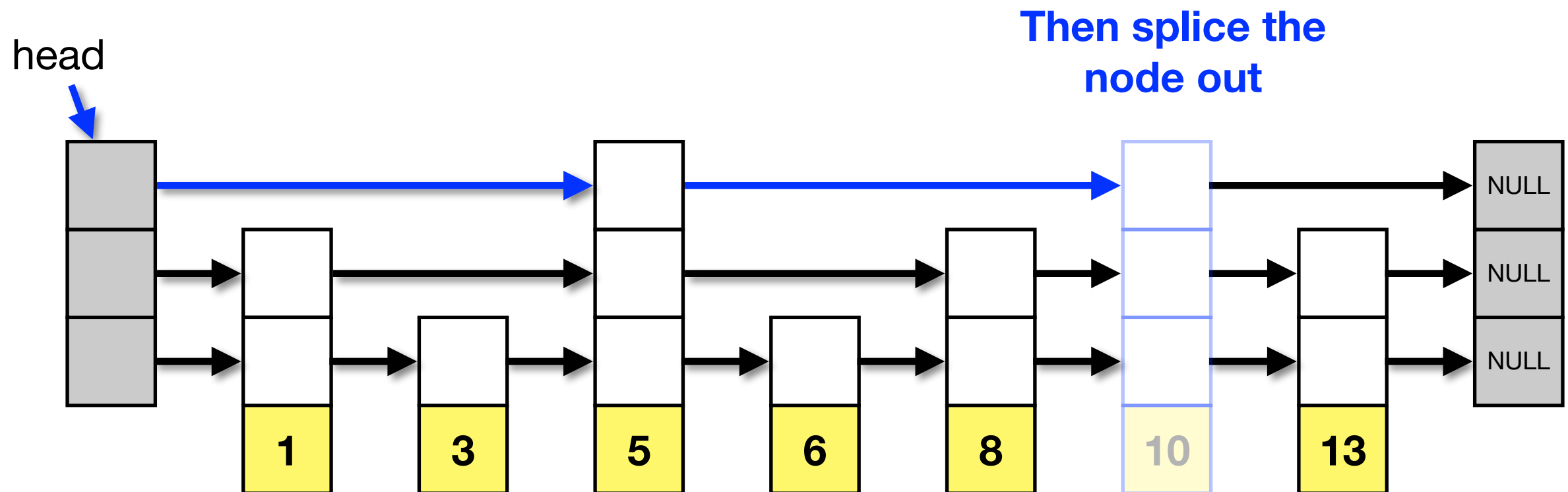
# Skip List Deletion

- For each level of which a skip list node is a part, splice the node out in the same fashion as a standard linked list
- Example -- delete the node with a key value 10



# Skip List Deletion

- For each level of which a skip list node is a part, splice the node out in the same fashion as a standard linked list
- Example -- delete the node with a key value 10



# Skip List Deletion

- For each level of which a skip list node is a part, splice the node out in the same fashion as a standard linked list
- Example -- delete the node with a key value 10

