COL106
Data Structures and Algorithms

Subodh Sharma and Rahul Garg

Height Balanced Trees: Summary

- AVL trees
- Height of an AVL tree is O(log(N))
- Insert, find, remove operations take O(log(N)) time
- Can be used for sorting and indexing

2-4 Trees

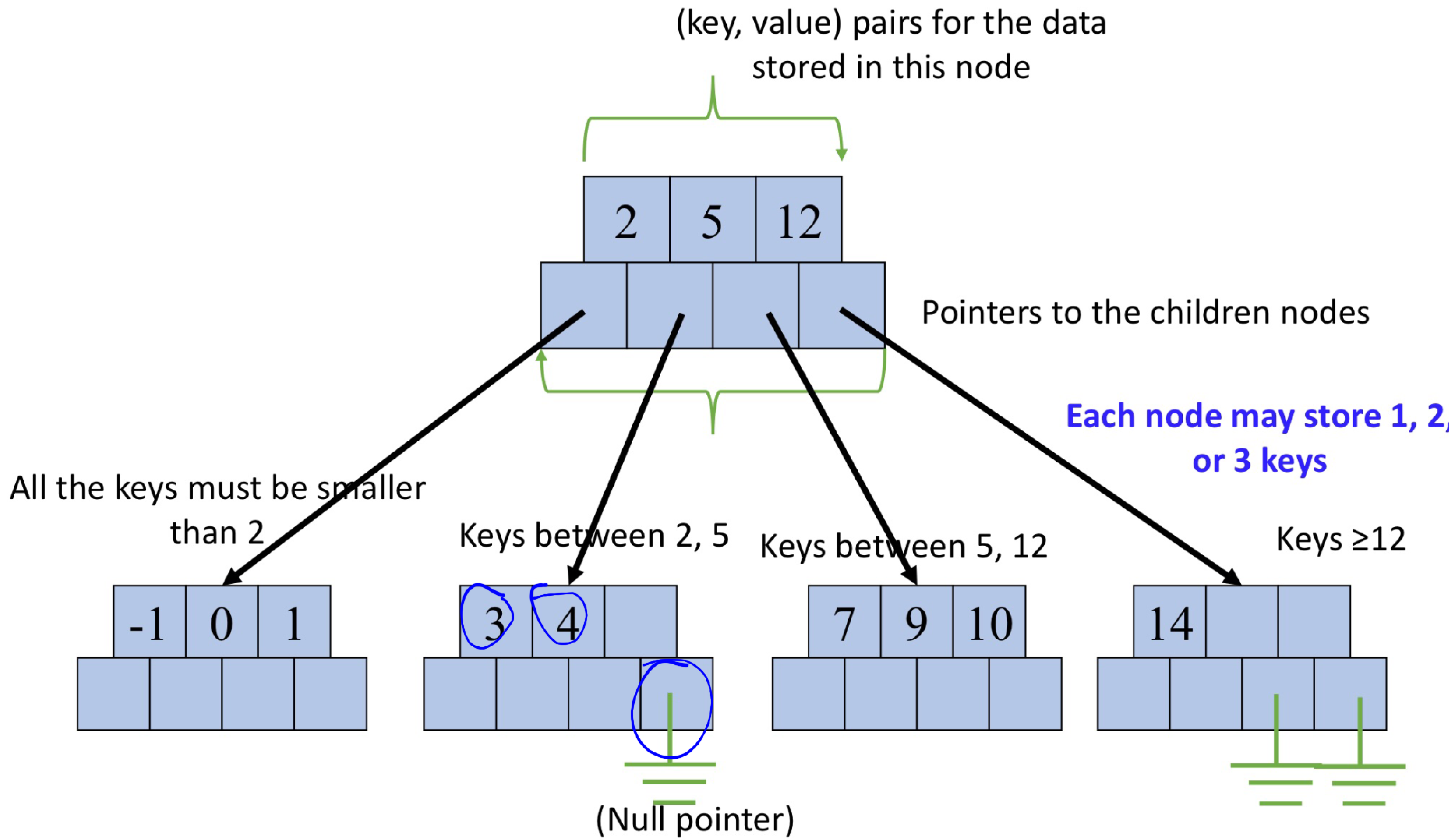
About 2-4 Trees

- Search trees
- Not binary
- Called 2-4 trees or 2-3-4 trees
- Balanced
- Each leaf node has the same depth

What are 2-4 Trees?

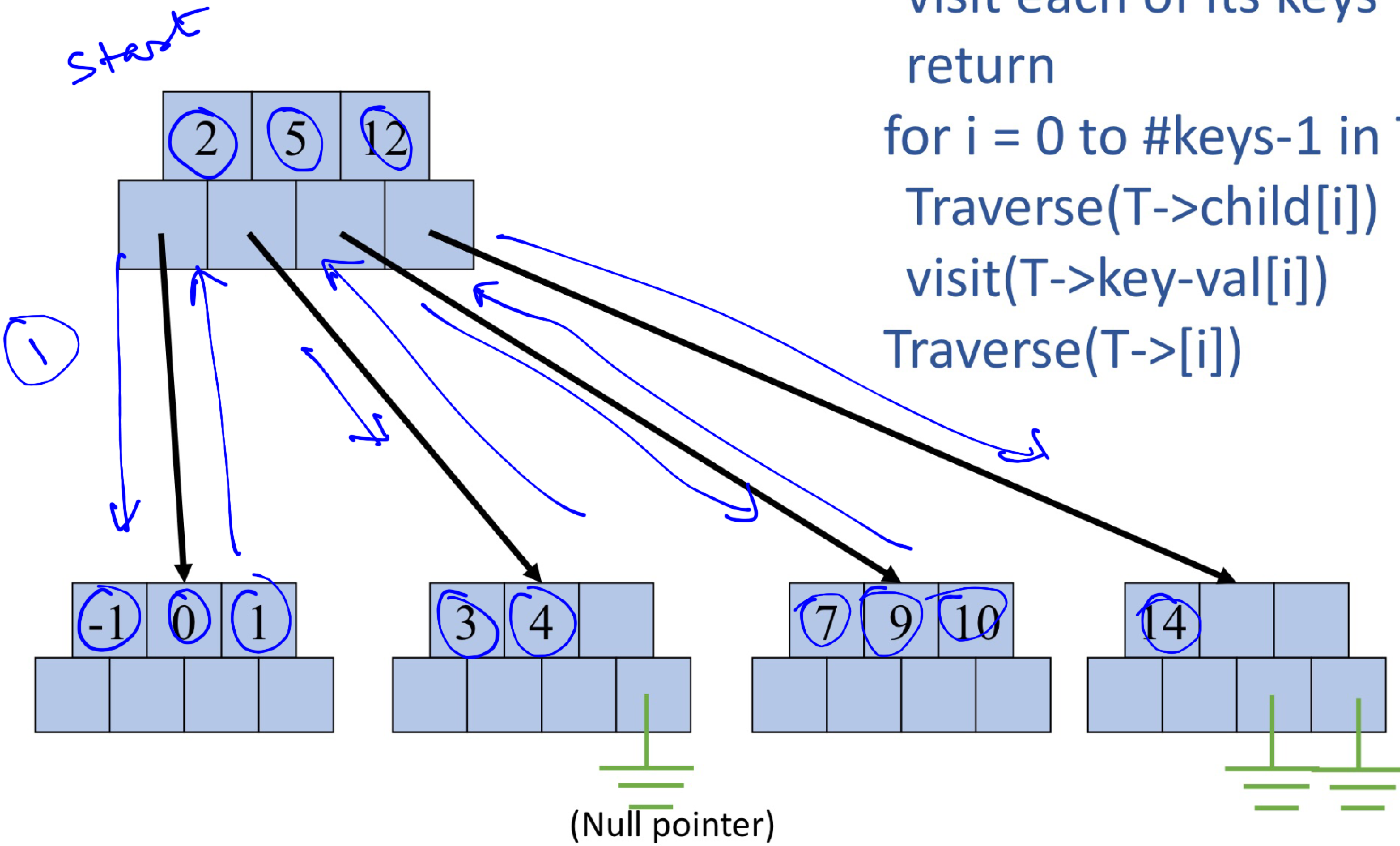
- Each internal node may have
  - 2, 3 or 4 children
  - Stores 1, 2 or 3 keys respectively
- The key-space is partitioned by the keys stored in the internal nodes
- Subtree only contain keys between the two keys stored in the parent
- Leaves have no children

2-4 Tree Node Structure

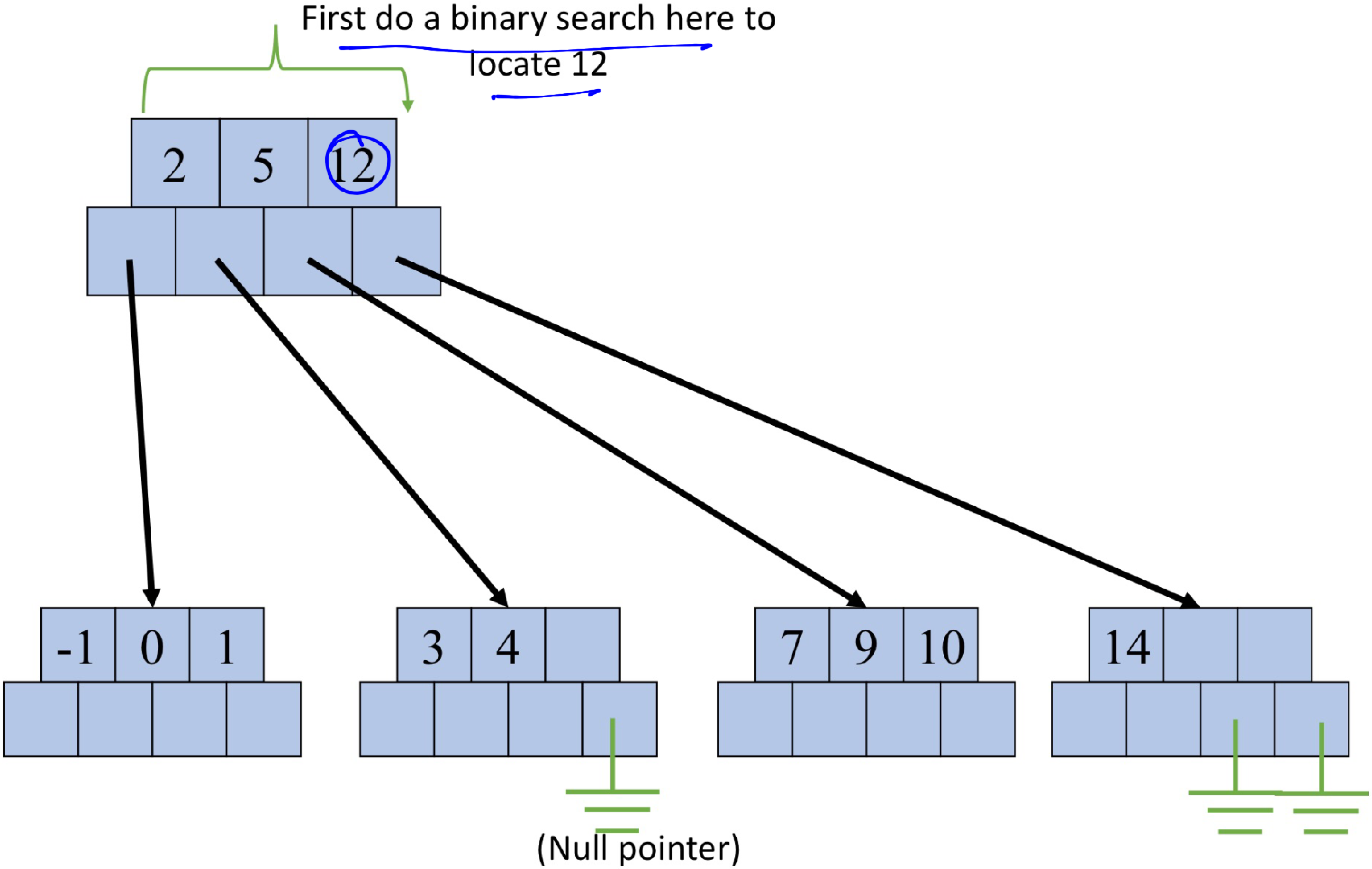


2-4 Tree Traversal (In-order)

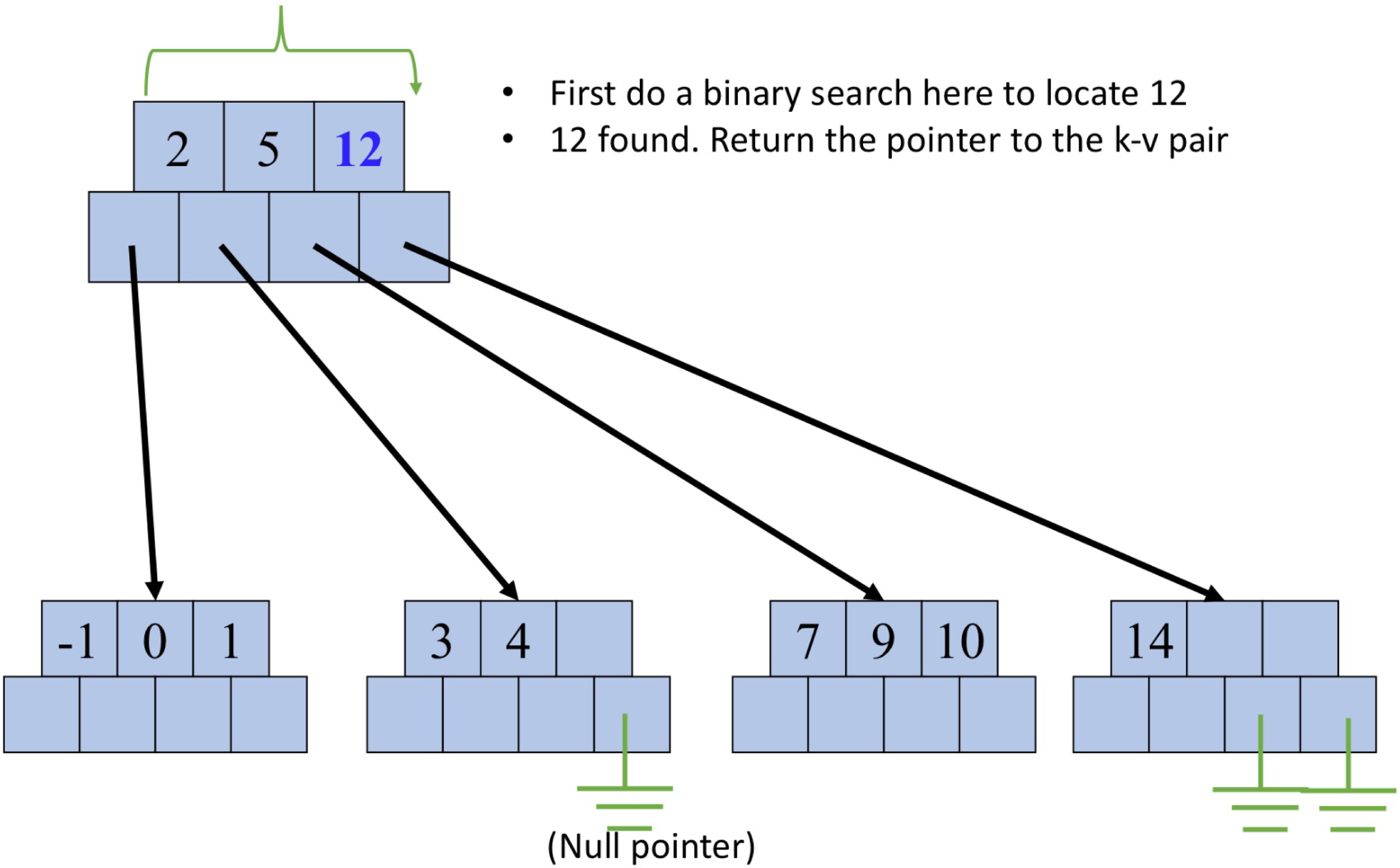
Traverse(T)
  If T is a leaf node
    visit each of its keys
    return
  for i = 0 to #keys-1 in T
    Traverse(T->child[i])
    visit(T->key-val[i])
  Traverse(T->[i])



How to Find a Key? Find 12

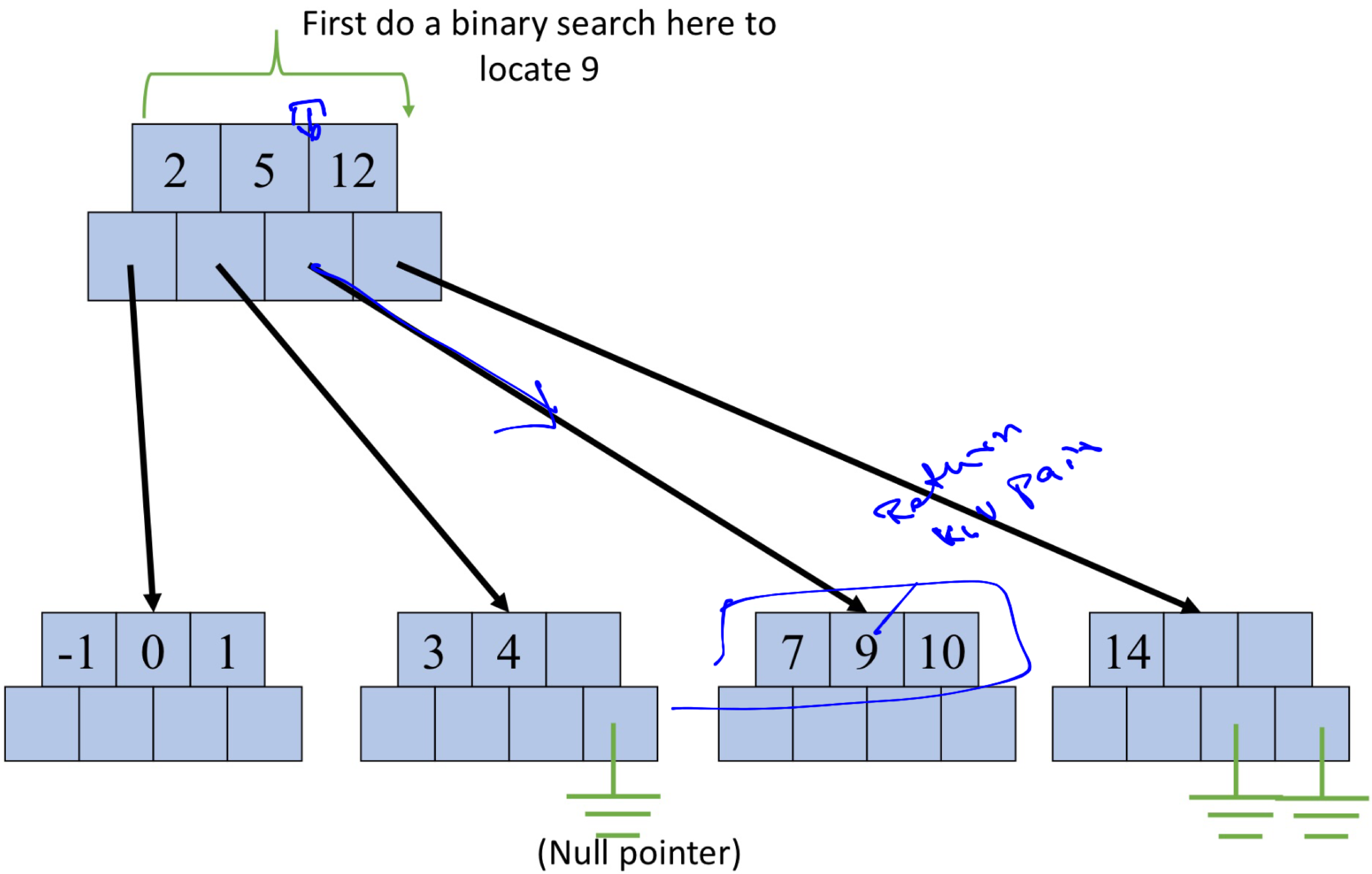


How to Find a Key? Find 12

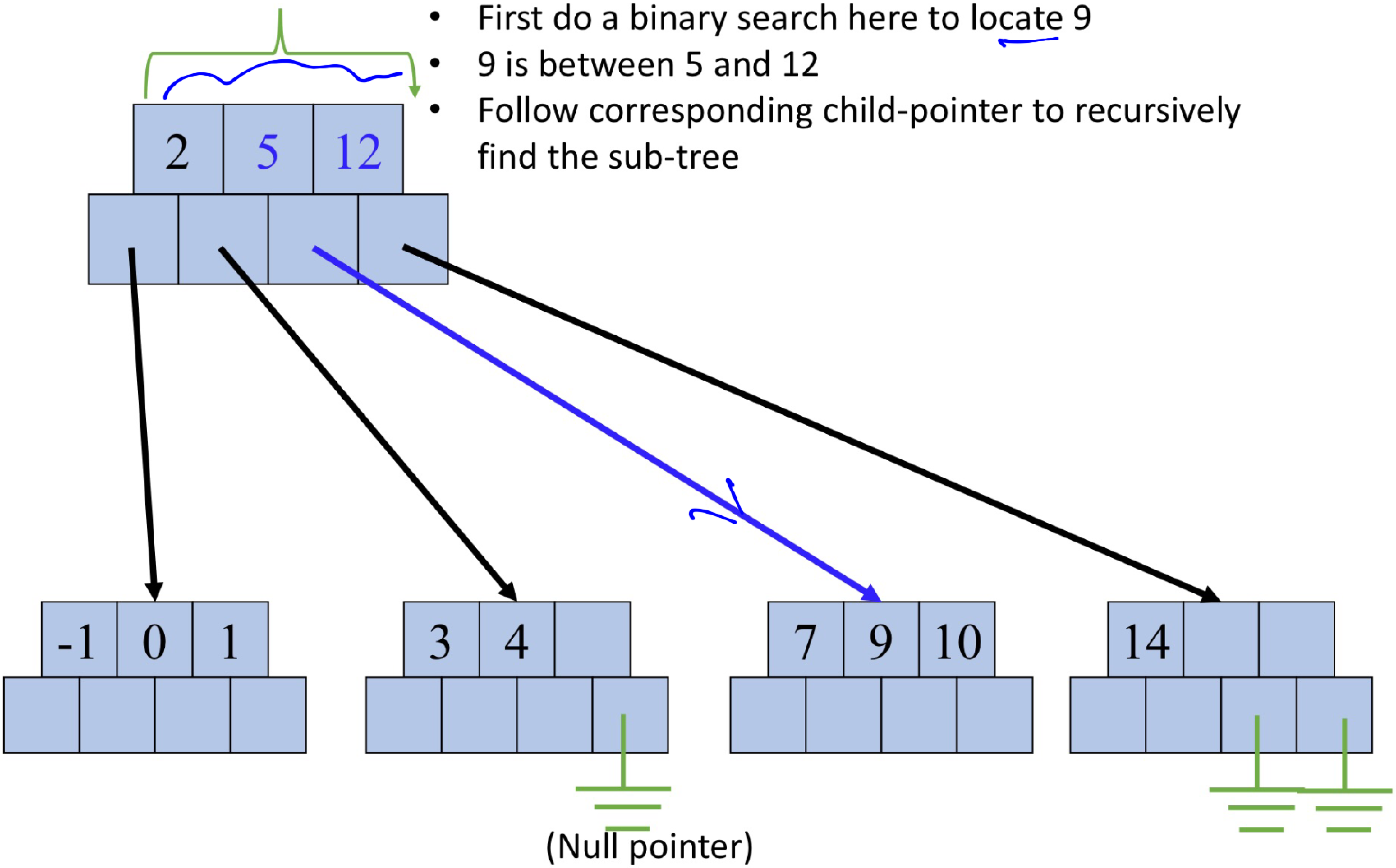


- First do a binary search here to locate 12
- 12 found. Return the pointer to the k-v pair

How to Find a Key? Find 9



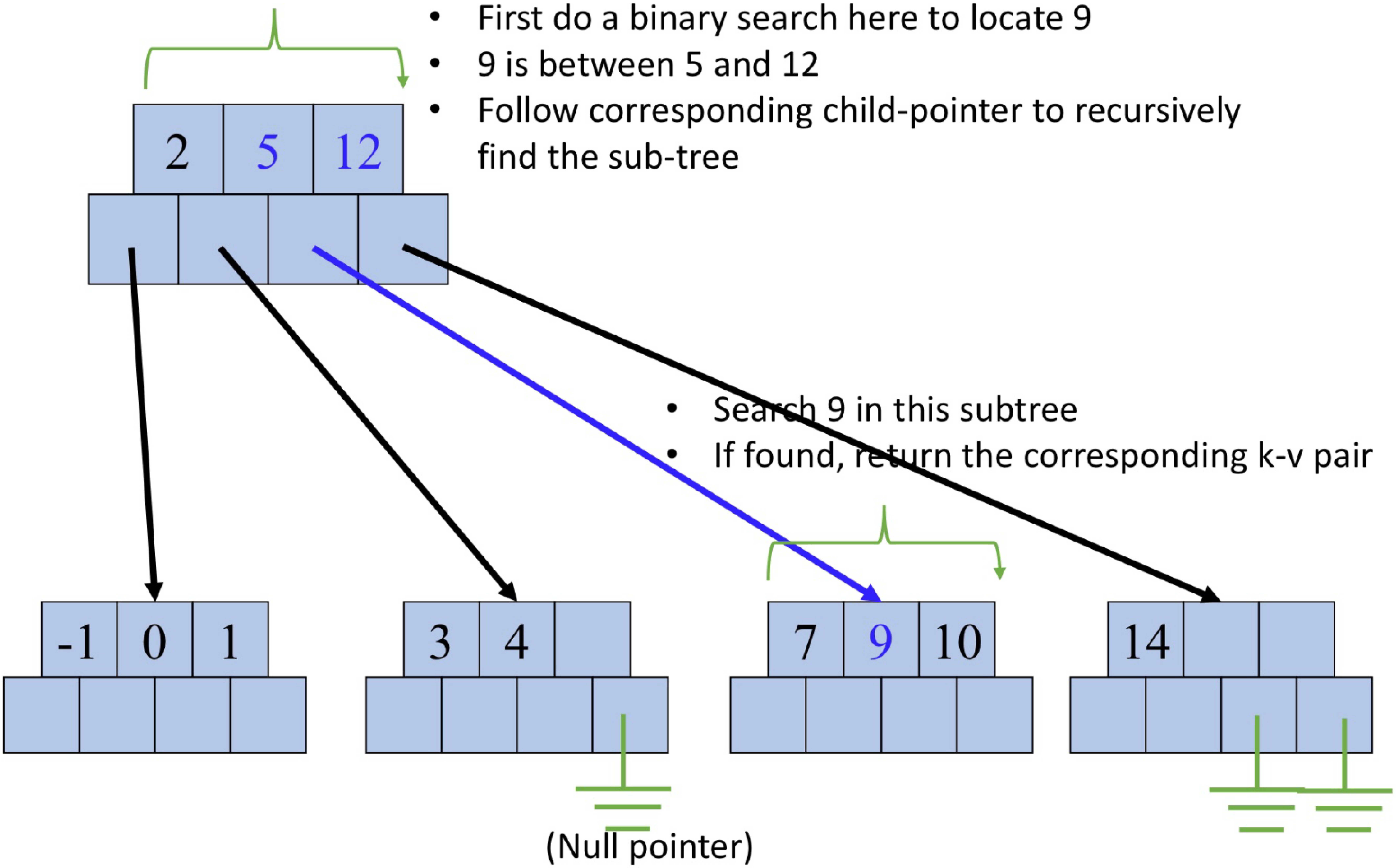
How to Find a Key? Find 9



- First do a binary search here to locate 9
- 9 is between 5 and 12
- Follow corresponding child-pointer to recursively find the sub-tree

How to Find a Key? Find 9

- First do a binary search here to locate 9
- 9 is between 5 and 12
- Follow corresponding child-pointer to recursively find the sub-tree



- Search 9 in this subtree
- If found, return the corresponding k-v pair

Find in 2-4 Trees

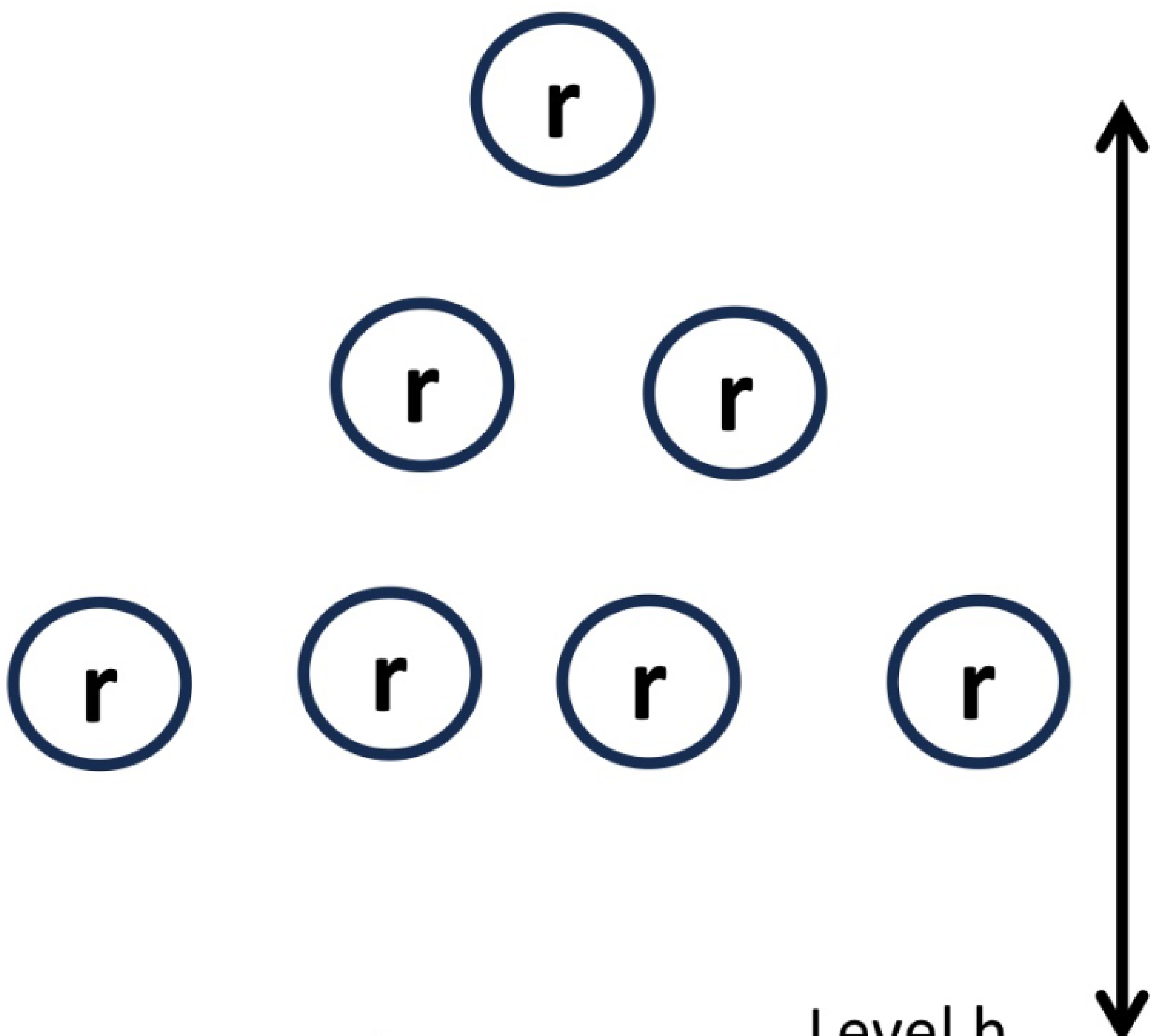
```
find(k, T)
  if k found in T return found
  if T is a leaf, return not-found
  if k < T->key[0]
    return find(k, T->child[0])
  if k > T->key[last]
    return find(k, T->child[last+1])
  find i such that:
    T->key[i] < k < T->key[i+1]
    return find(k, T->child[i])
```

Running Time for Find?

- Consider a 2-4 tree of height h
- All the leaves are at depth h
- Each internal node has 2 or 3 or 4 children
- Can we bound the number of nodes: N(h)?

Bounding N(h) for 2-4 Trees

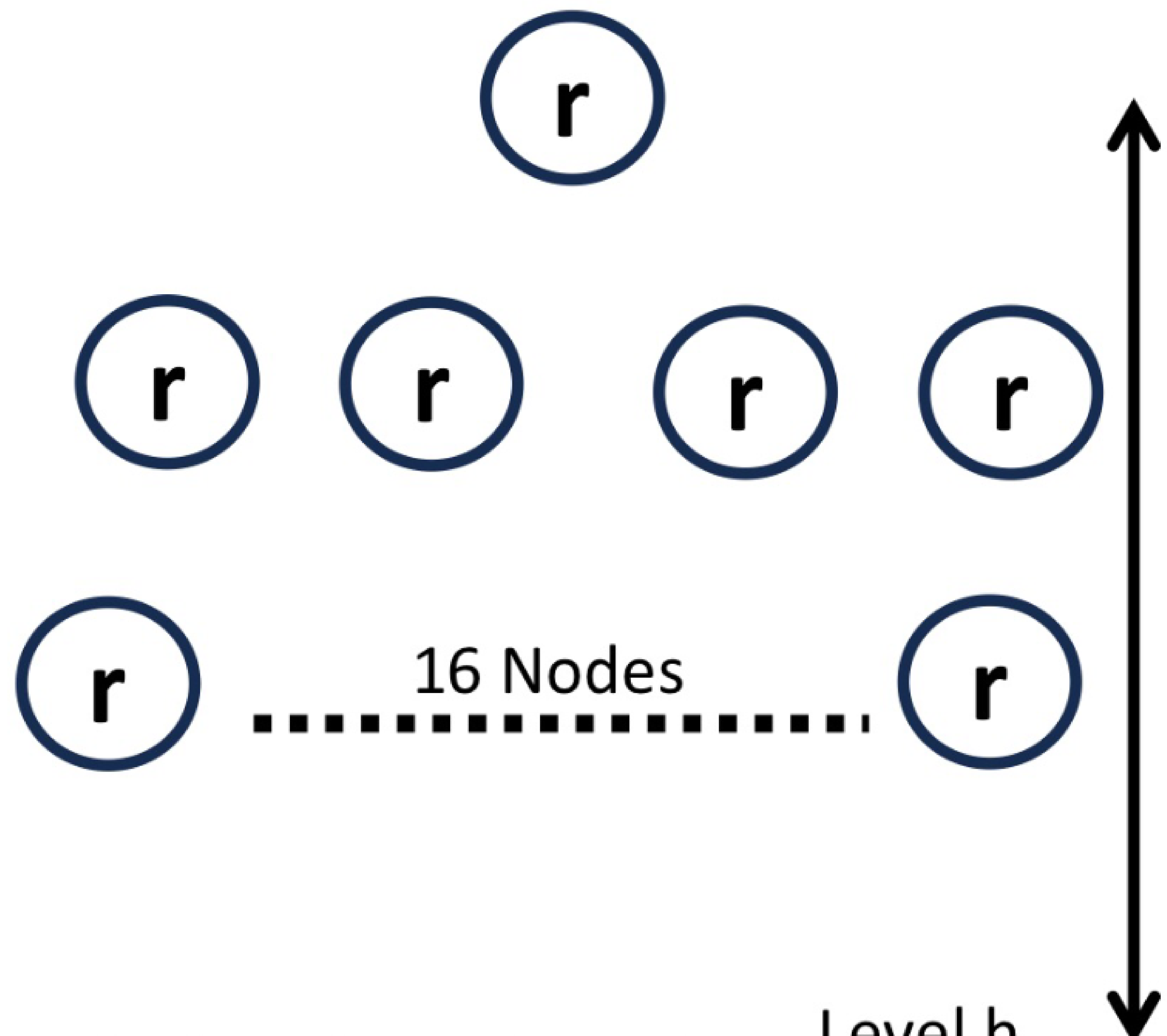
Smallest possible N(h)



$N(h) \geq 2^{h+1}-1$

Level h
At least 2^h leaves

Largest possible N(h)



$N(h) \leq (4^{h+1}-1)/3$

Level h
At least 4^h leaves

Running Time for Find?

- Consider a 2-4 tree of height h
- All the leaves are at depth h
- Each internal node has 2 or 3 or 4 children

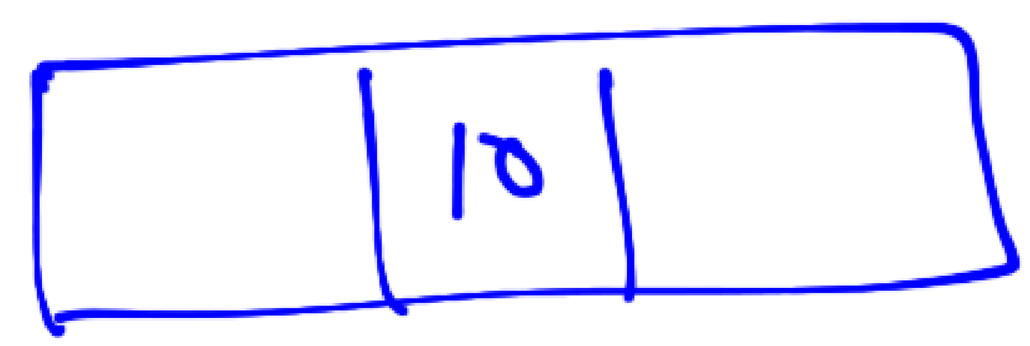
$$\mathbf{2^{h+1}-1 \le N(h) \le (4^{h+1}-1)/3}$$

$$\mathbf{\tfrac{1}{2} \log_2(3N+1) \le h \le \log_2(N+1)}$$

h = O(log(N))

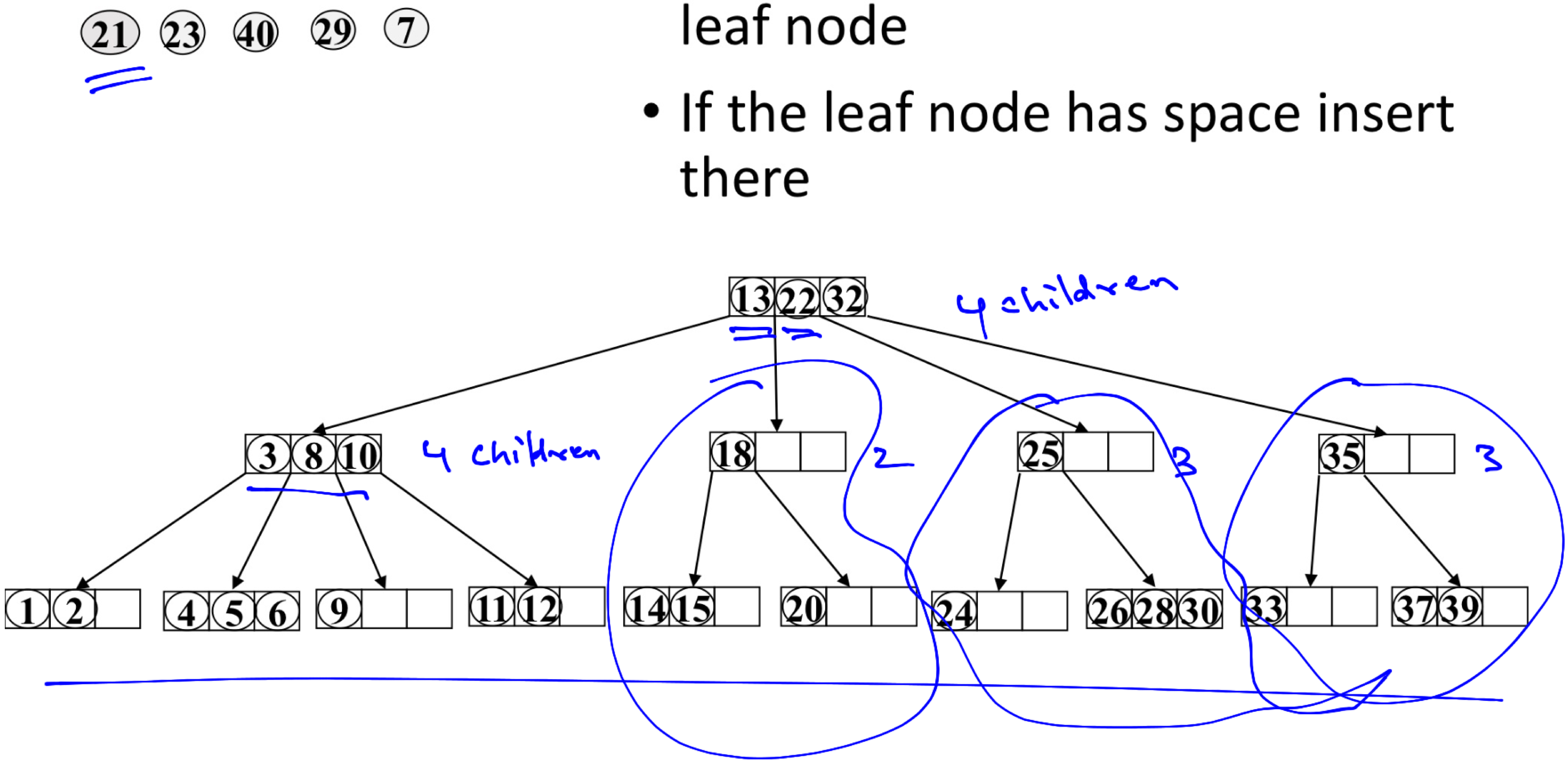
Find runtime is O(log(N))

How to Insert in a 2-4 Tree?



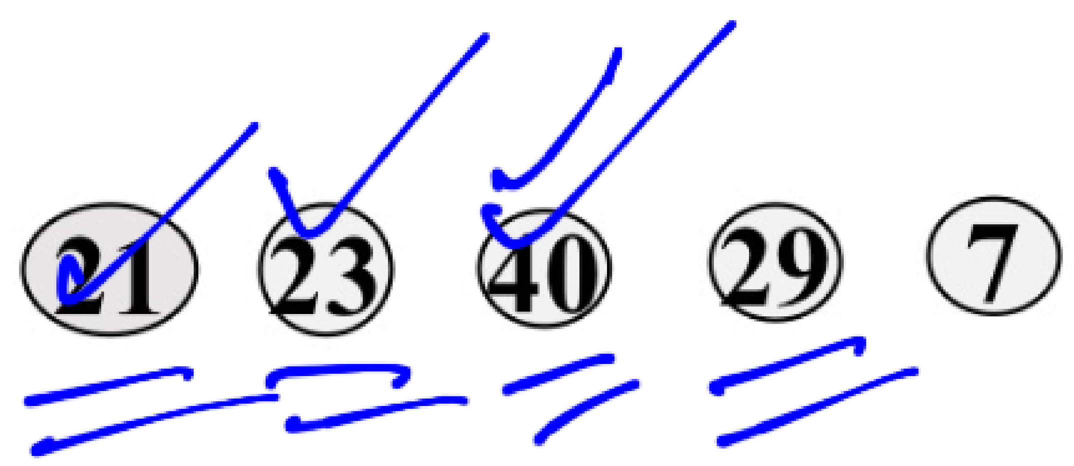
Insertion

- Insertion always traverses to a leaf node
- If the leaf node has space insert there

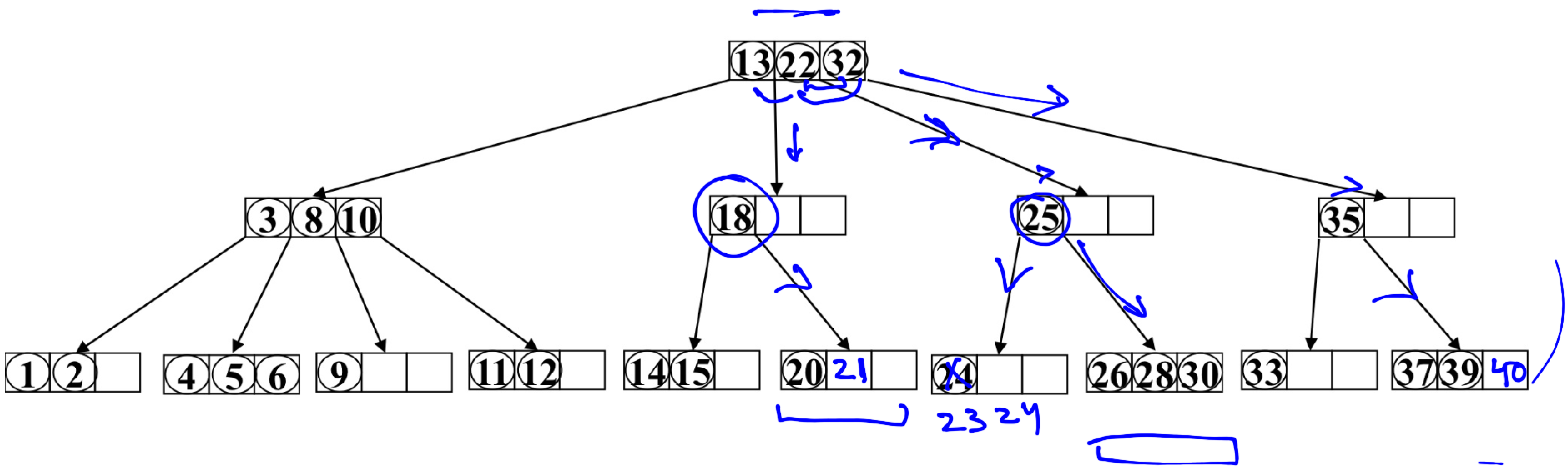


Slide adapted from Naveen Garg, IIT Delhi

Insertion



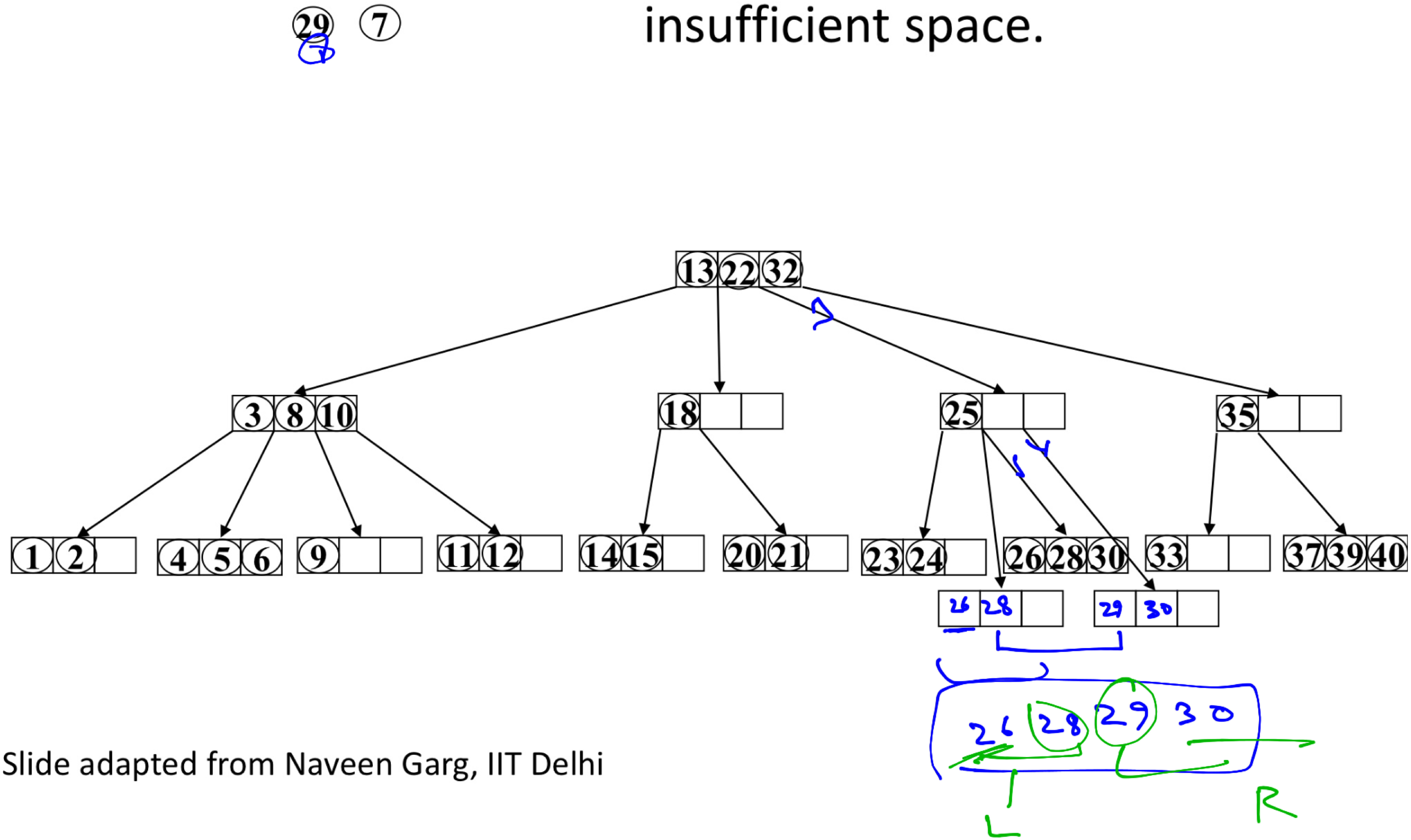
- Insertion always traverses to a leaf node
- If the leaf node has space insert there



Slide adapted from Naveen Garg, IIT Delhi

Insertion

- Nodes get split if there is insufficient space.



Slide adapted from Naveen Garg, IIT Delhi