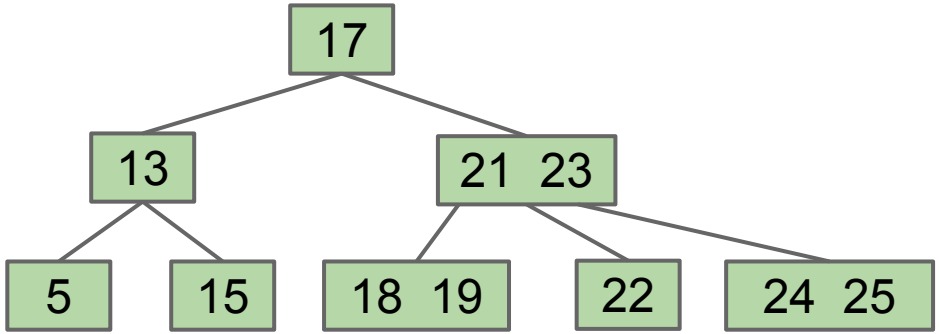
2-3 Tree Deletion (Extra)

This will not be covered in any homework, lab, or exam.

Deletion Operations

As with regular Binary Search Trees, deletion is a more complicated operation.

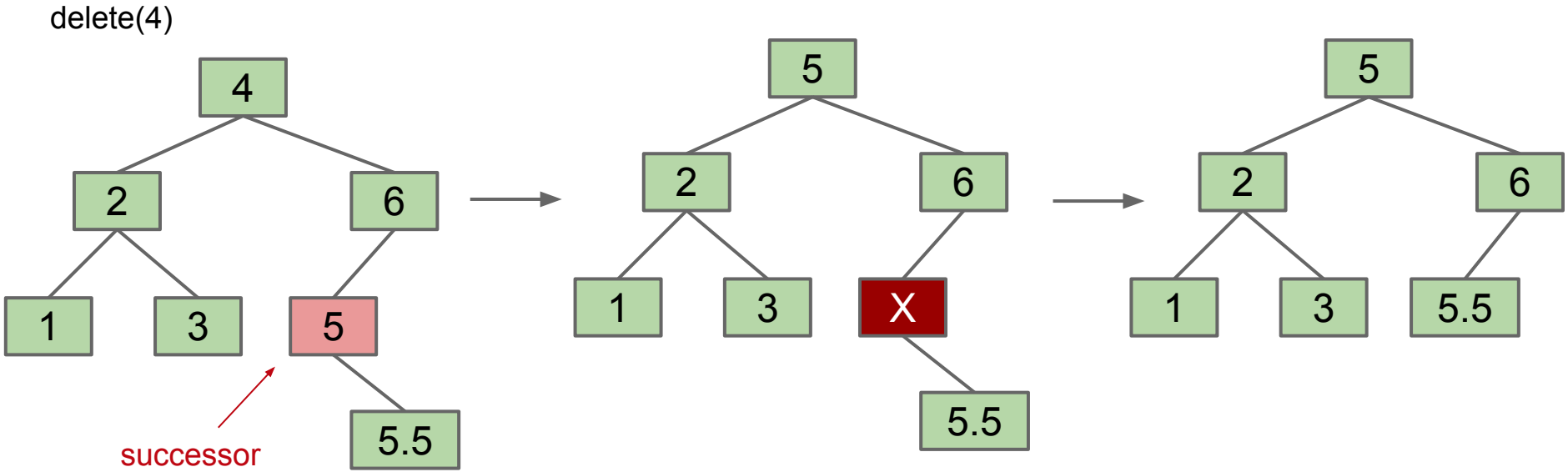
- Many possible deletion algorithms.
- We'll develop a deletion algorithm together.



Deletion from a Regular BST (Review)

In a regular BST, when we delete a value α with 2 children, we:

- Copy the value of the successor into α.
- Then we delete the successor.



Deletion from a 2-3 Tree

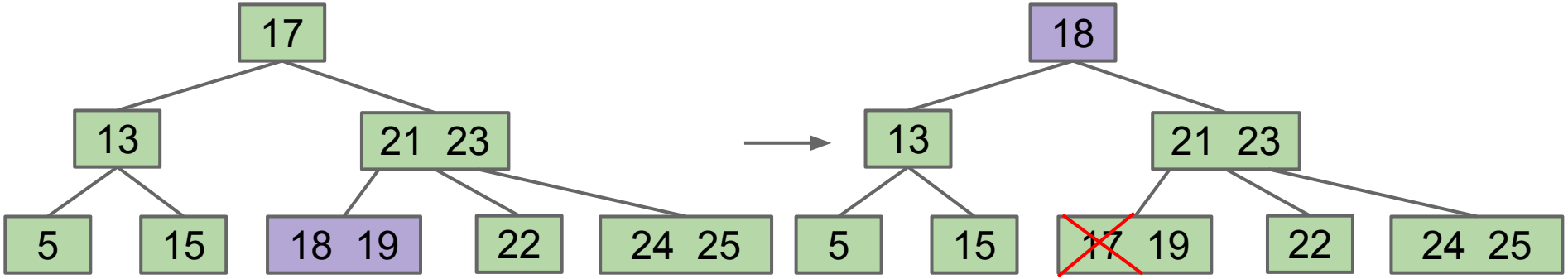
In a 2-3 Tree, when we delete α from a node with 2 or more children, we:

- Swap the value of the successor with α.
- Then we delete the successor value.

Note: Successor will always be in a leaf node!

Example: delete(17):

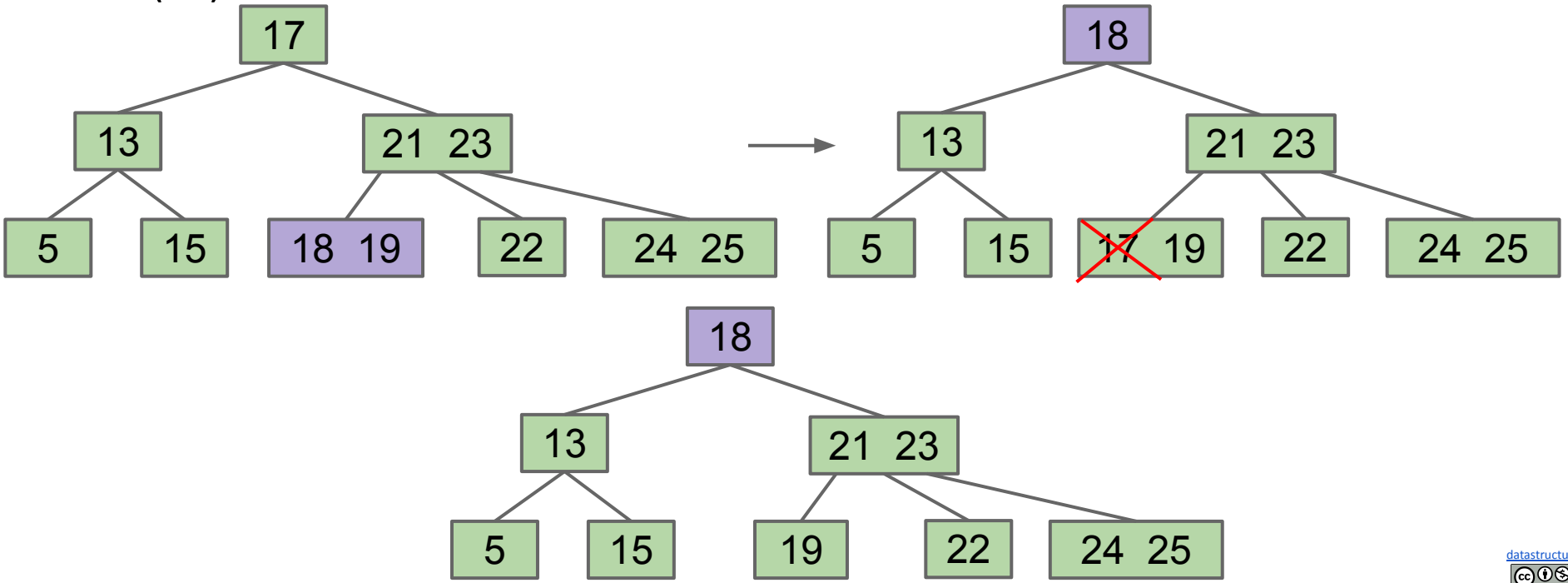
- Swap 17 with its successor 18, then delete 17.



Multi-Key Leaves vs. Single-Key Leaves

If deleting from a leaf with multiple keys, the deletion is trivial. We simply remove the item from the leaf, and we are done.

delete(17):



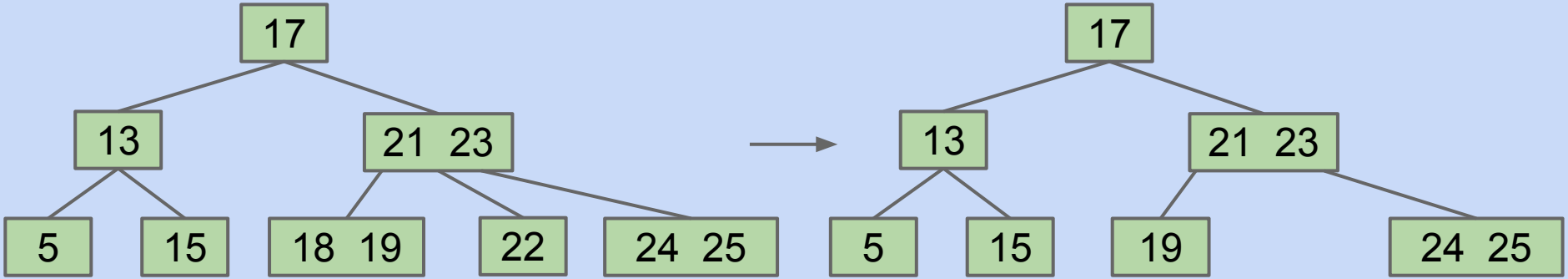
Multi-Key Leaves vs. Single-Key Leaves

If our leaf has multiple keys, the deletion is trivial. We simply remove the item.

If our leaf has a single key, we cannot simply remove the node entirely.

- Why?

delete(22):



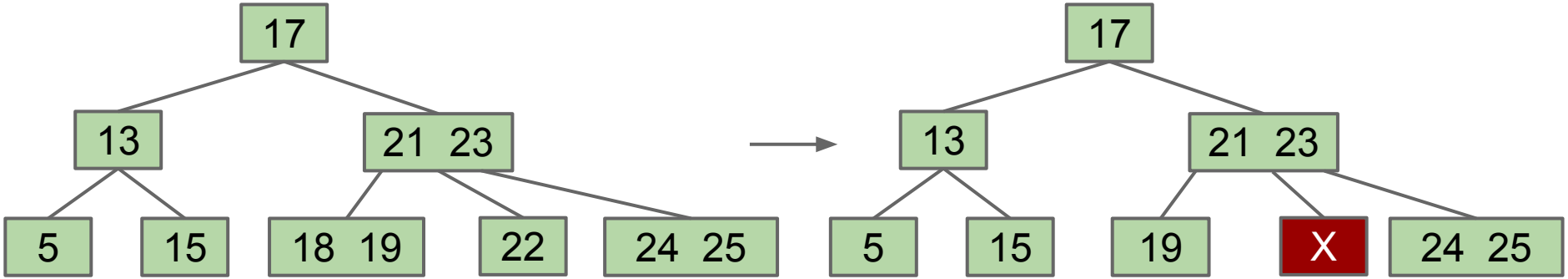
Multi-Key Leaves vs. Single-Key Leaves

If our leaf has multiple keys, the deletion is trivial. We simply remove the item.

If our leaf has a single key, we cannot simply remove the node entirely.

- Any node with k items must have k + 1 children!
- Instead, we'll leave an empty node, which must be filled.
- Filling the empty node is complex and has many cases (coming soon).

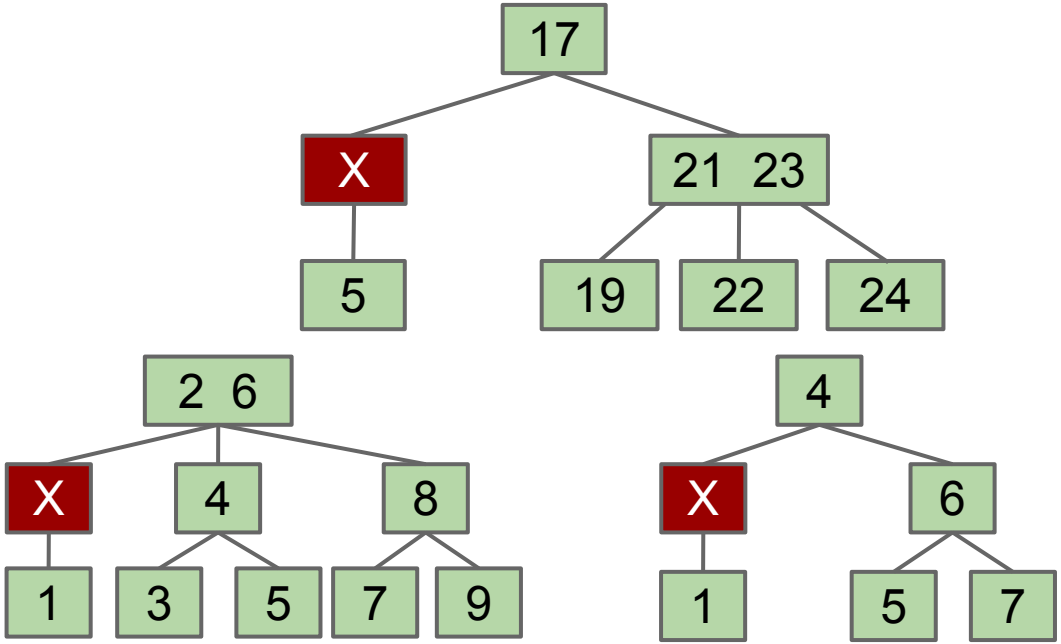
delete(22):



Filling in Empty Nodes (FIEN)

The hard part about deletion is filling empty nodes.

- There are three interesting cases to consider.
- For reasons that will become clear later, we will talk about how to fill empty boxes ANYWHERE in the tree, not just in the leaves.

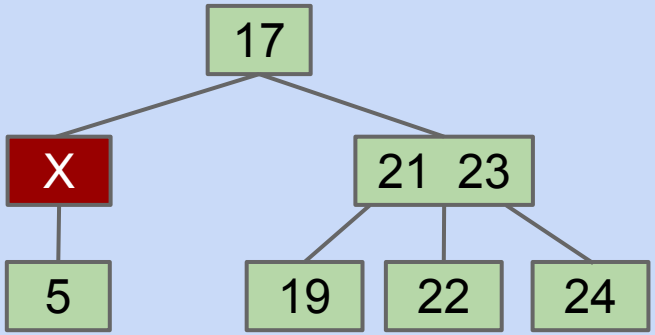


Since empty nodes contain k=0 items, non-leaf empty nodes have 1 child!

FIEN Case 1A: Multi-Key Sibling

In case 1A, the empty node's adjacent sibling has multiple keys.

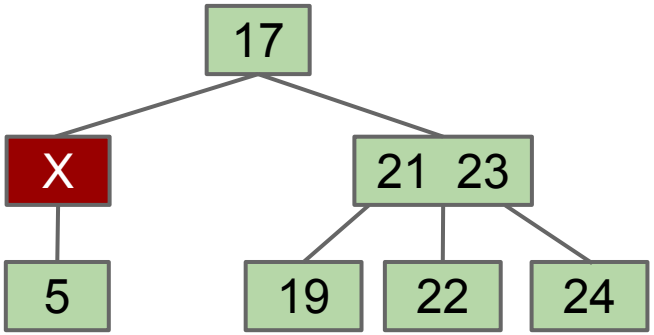
- Very hard optional challenge: Try to fill in X in the diagram below so that the result is a valid 2-3 tree.

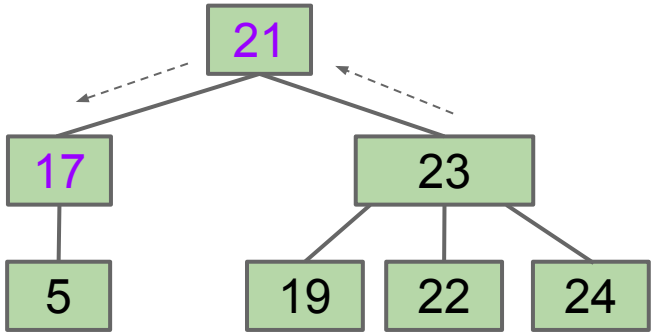


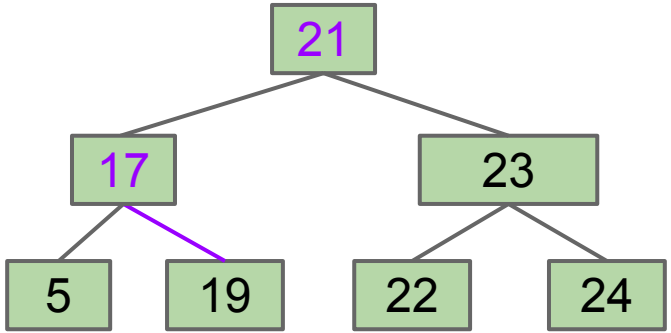
FIEN Case 1A: Multi-Key Sibling

In case 1A, the empty node's adjacent sibling has multiple keys.

- X steals parent's item. Parent steals sibling's item.
- If X was not a leaf node, X steals one of sibling's subtrees.







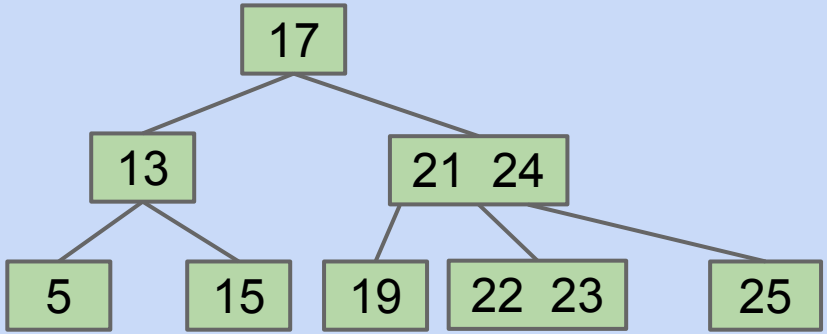
FIEN Case 1A: Multi-Key Sibling

In case 1A, the empty node's adjacent sibling has multiple keys.

- X steals parent's item. Parent steals sibling's item.
- If X was not a leaf node, X steals one of sibling's subtrees.

Try to delete 17 from the tree below.

- Hint: You'll end up in FIEN Case 1.



FIEN Case 1A: Multi-Key Sibling

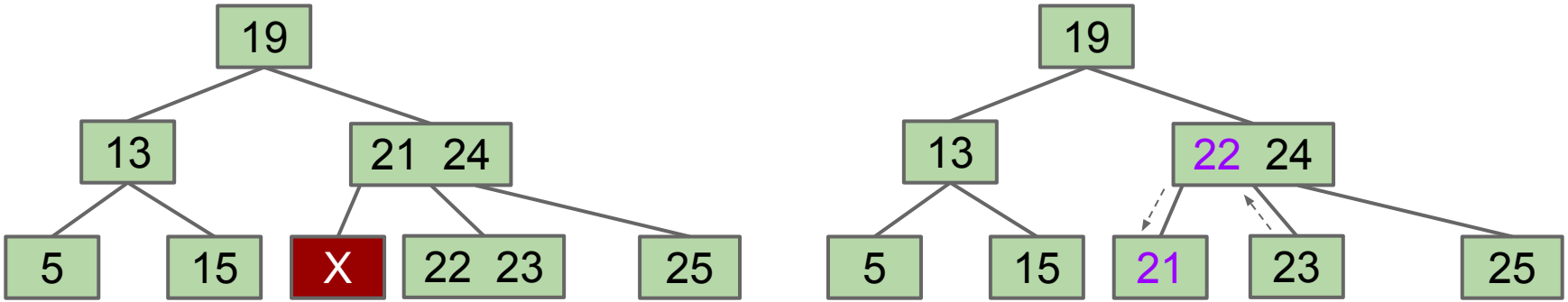
In case 1A, the empty node's adjacent sibling has multiple keys.

- X steals parent's item. Parent steals sibling's item.
- If X was not a leaf node, X steals one of sibling's subtrees.

Wasn't necessary since X was not a leaf node.

Try to delete 17 from the tree below.

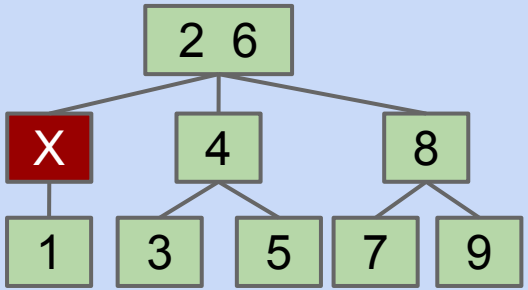
- Swap 17 with 19, then delete 17.



FIEN Case 2A: Multi-Key Parent

In case 2A, siblings on the right all have one key, but parent has two.

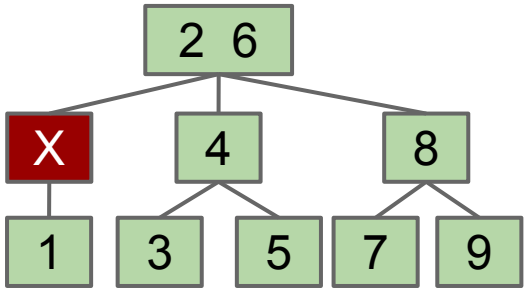
- Very hard optional challenge: Try to fill in X in the diagram below so that the result is a valid 2-3 tree.

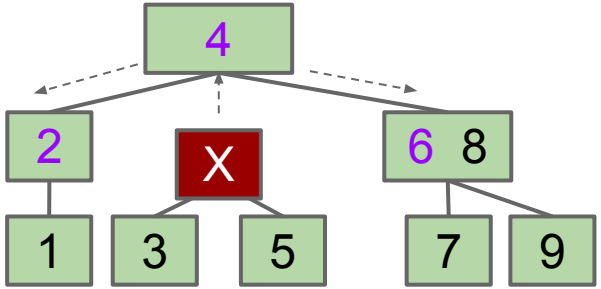


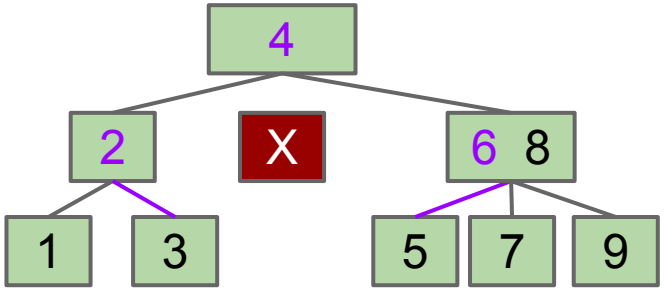
FIEN Case 2A: Multi-Key Parent

In case 2A, siblings on the right all have one key, but parent has two.

- X and right sibling steal parent's keys. Middle sibling moves into the parent.
- Subtrees are passed along so that every node has the correct children.





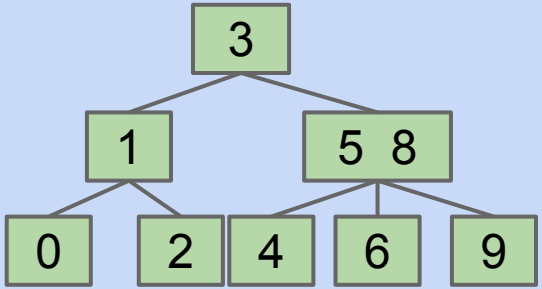


FIEN Case 2A Exercise

In case 2A, siblings on the right all have one key, but parent has two.

- X and right sibling steal parent's keys. Middle sibling moves into the parent.
- Subtrees are passed along so that every node has the correct children.

Try to delete 3 from the tree below.

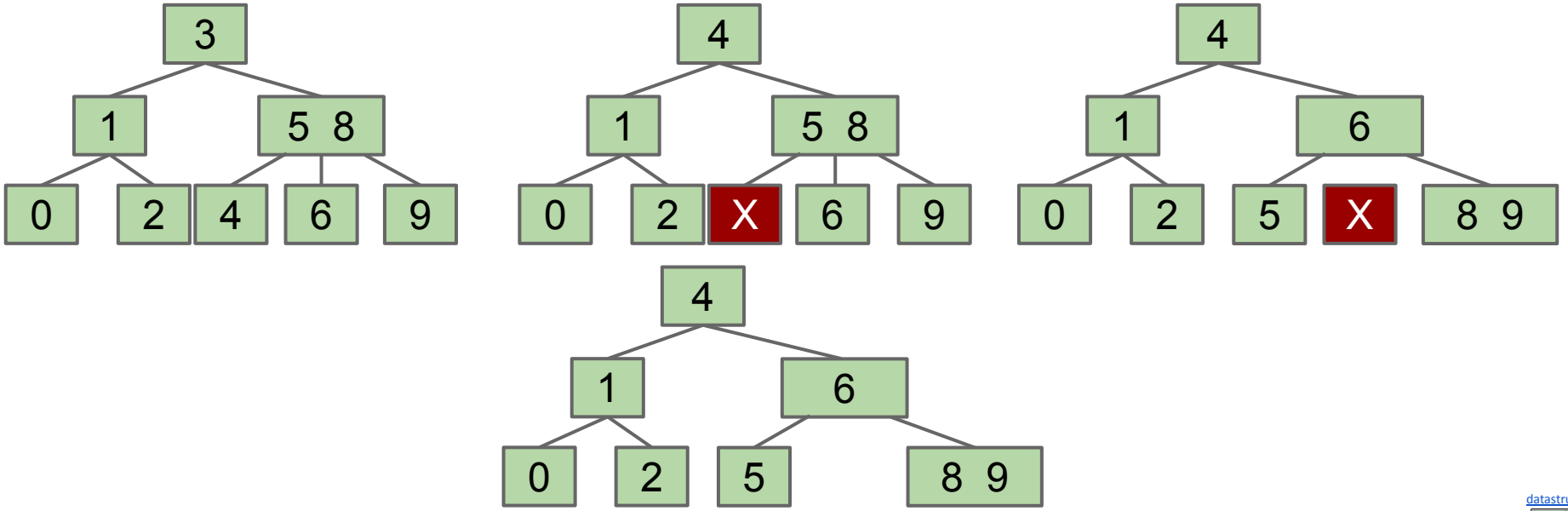


FIEN Case 2A Exercise

In case 2A, siblings on the right all have one key, but parent has two.

- X and right sibling steal parent's keys. Middle sibling moves into the parent.
- Subtrees are passed along so that every node has the correct children.

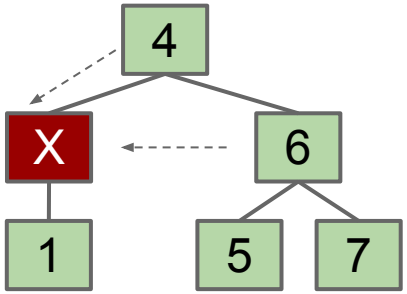
delete(3)

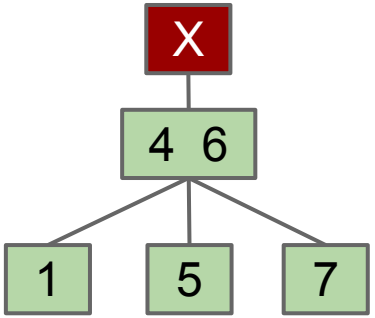


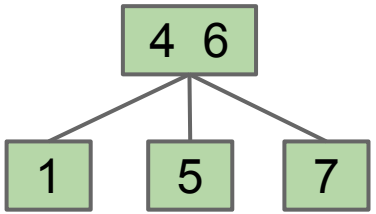
FIEN Case 3: Single-Key Parent and Sibling

In FIEN Case 3: The parent and all siblings have only one item.

- Combine 1 sibling and parent into a single node that replaces X. Send the blank X up one level.
- If blank ends up as the new root, just delete the blank and we are done.



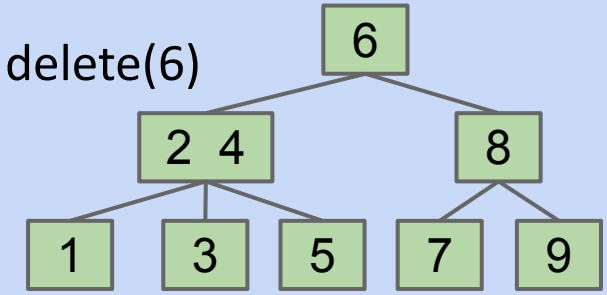




FIEN Case 3 Exercise

In FIEN Case 3: The parent and all siblings have only one item.

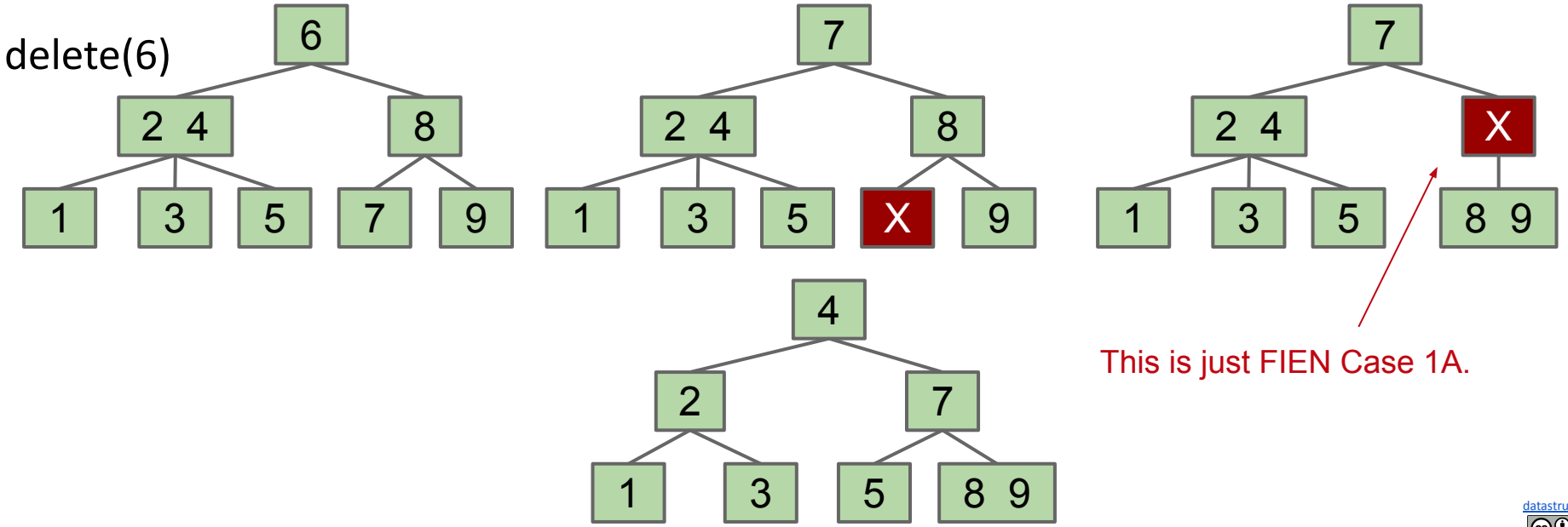
- Combine 1 sibling and parent into a single node that replaces X. Send the blank X up one level.
- If blank ends up as the new root, just delete the blank and we are done.



FIEN Case 3 Exercise

In FIEN Case 3: The parent and all siblings have only one item.

- Combine 1 sibling and parent into a single node that replaces X. Send the blank X up one level.
- If blank ends up as the new root, just delete the blank and we are done.

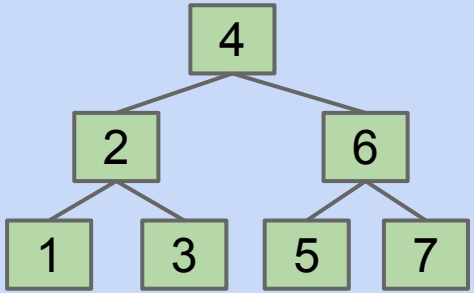


FIEN Case 3 Exercise #2

In FIEN Case 3: The parent and all siblings have only one item.

- Combine 1 sibling and parent into a single node that replaces X. Send the blank X up one level.
- If blank ends up as the new root, just delete the blank and we are done.

delete(4)



FIEN Case 3 Exercise #2

In FIEN Case 3: The parent and all siblings have only one item.

- Combine 1 sibling and parent into a single node that replaces X. Send the blank X up one level.
- If blank ends up as the new root, just delete the blank and we are done.

delete(4)