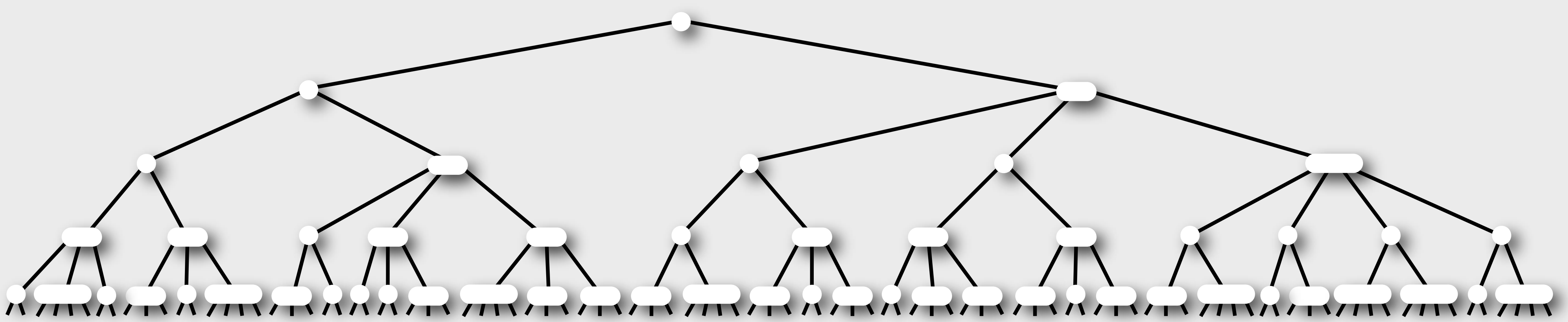
Introduction

2-3-4 Trees

LLRB Trees

Deletion

Analysis



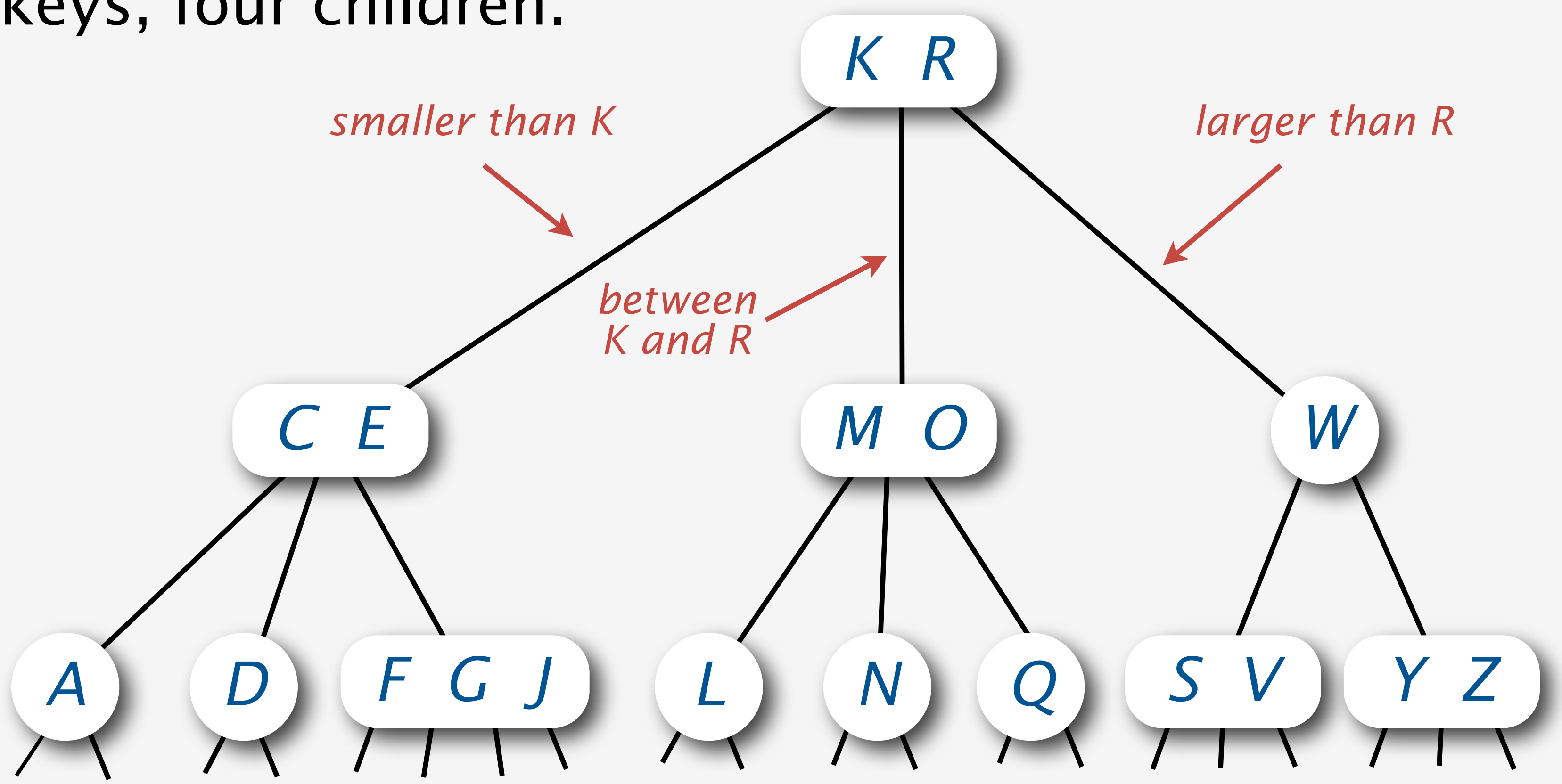
2-3-4 Tree

Generalize BST node to allow multiple keys.
Keep tree in perfect balance.

Perfect balance. Every path from root to leaf has same length.

Allow 1, 2, or 3 keys per node.

- 2-node: one key, two children.
- 3-node: two keys, three children.
- 4-node: three keys, four children.



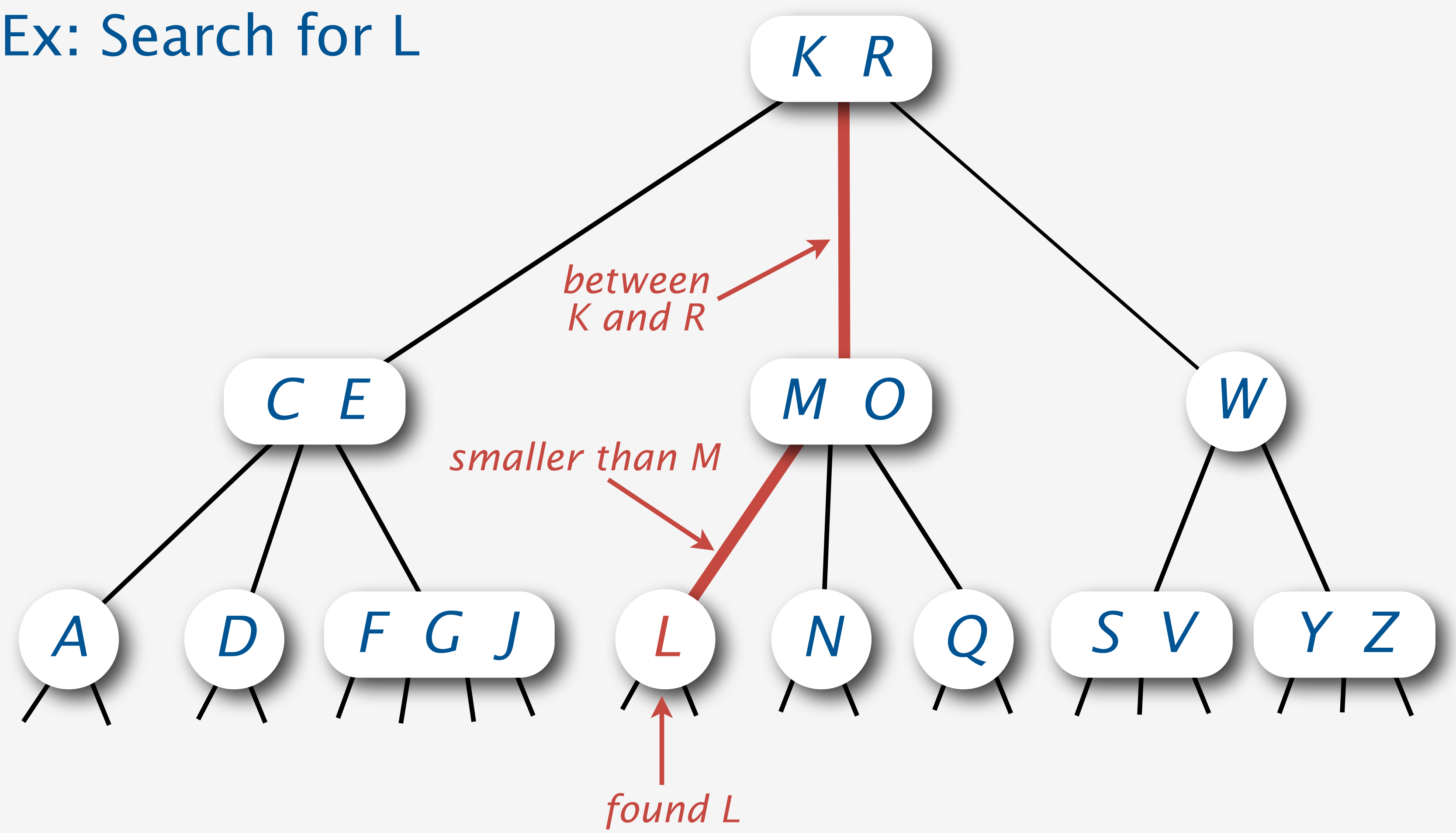
Search in a 2-3-4 Tree

Compare node keys against search key to guide search.

Search.

- Compare search key against keys in node.
- Find interval containing search key.
- Follow associated link (recursively).

Ex: Search for L



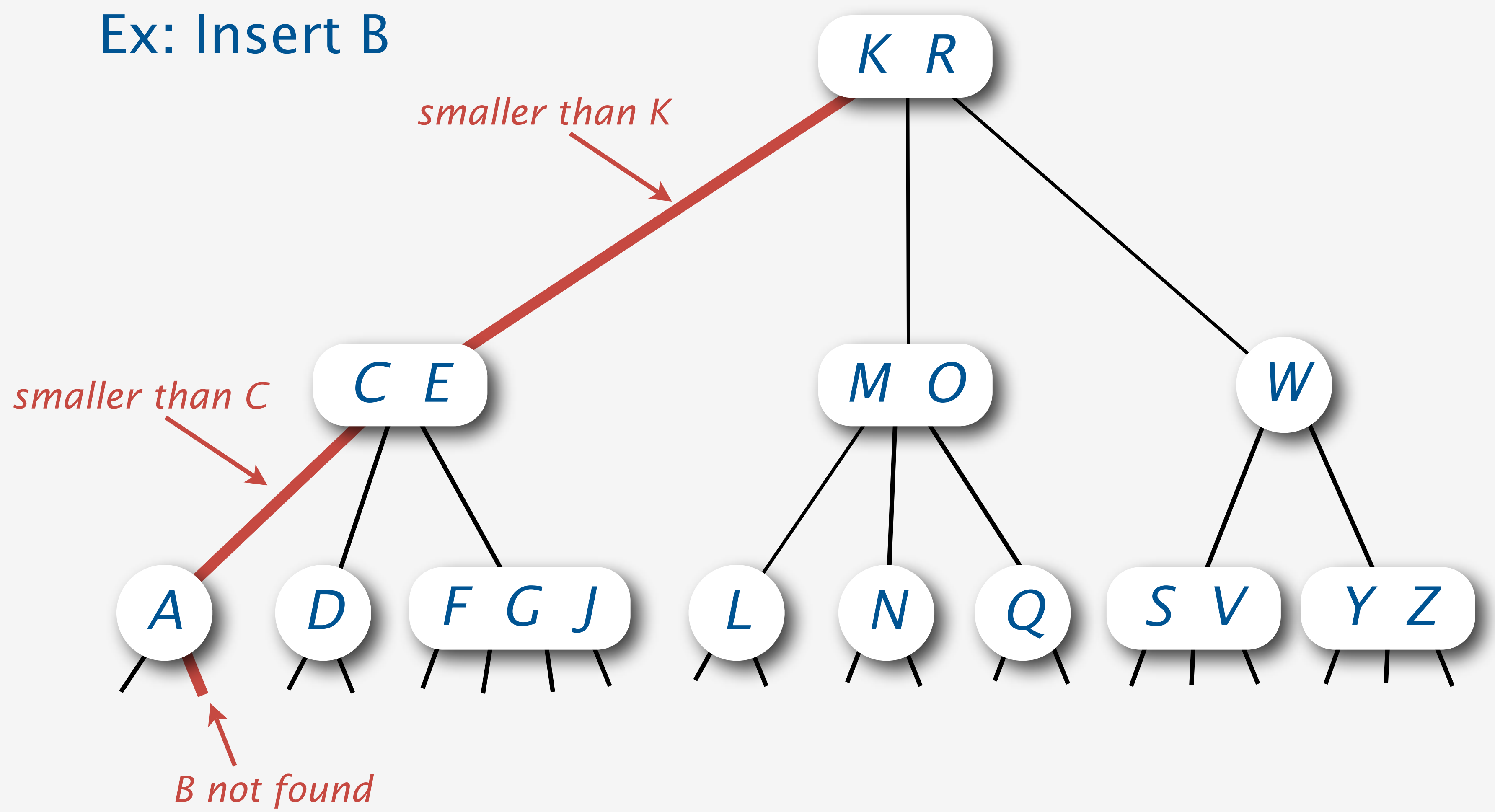
Insertion in a 2-3-4 Tree

Add new keys at the bottom of the tree.

Insert.

- Search to bottom for key.

Ex: Insert B



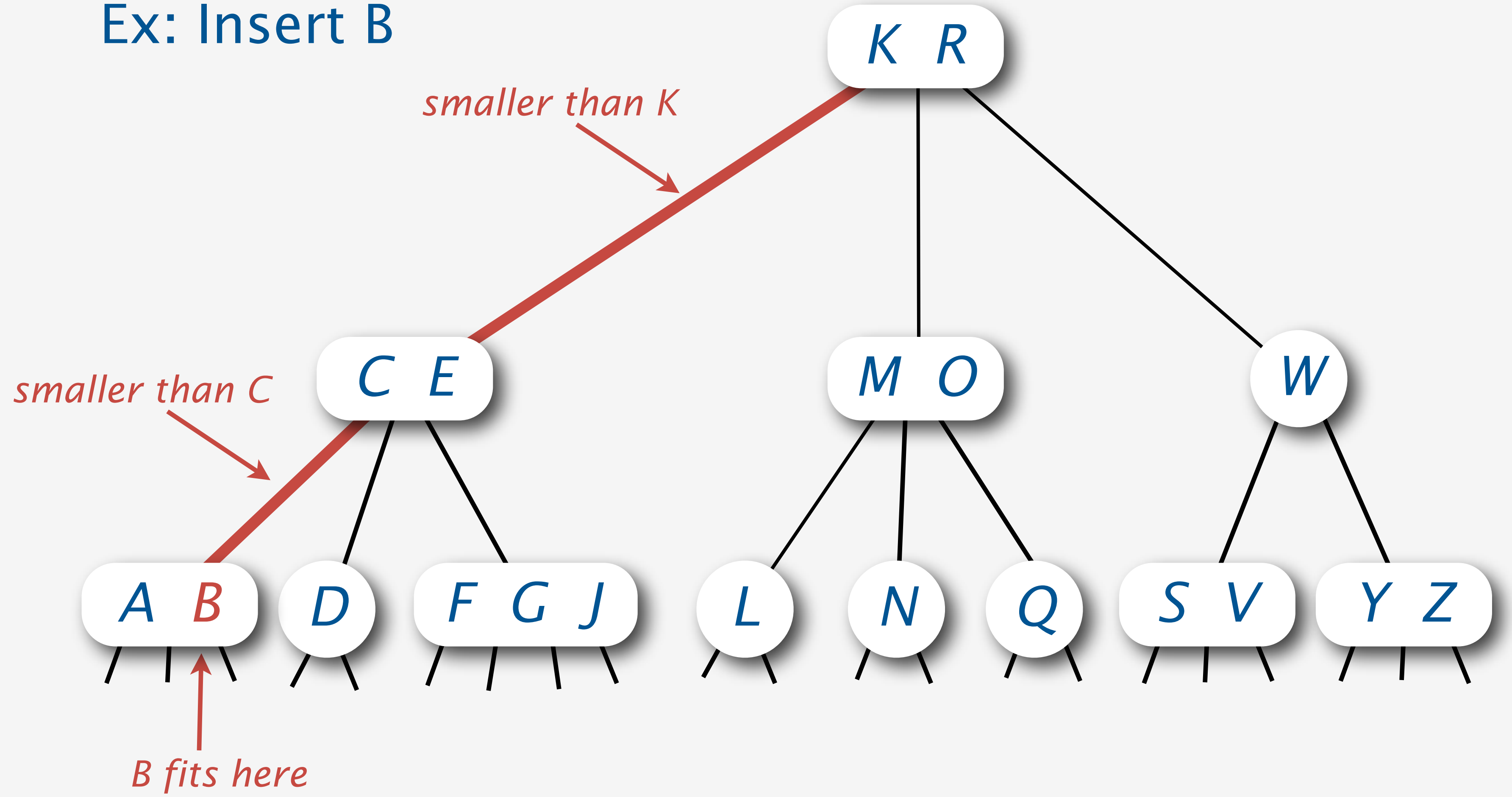
Insertion in a 2-3-4 Tree

Add new keys at the bottom of the tree.

Insert.

- Search to bottom for key.
- 2-node at bottom: convert to a 3-node.

Ex: Insert B



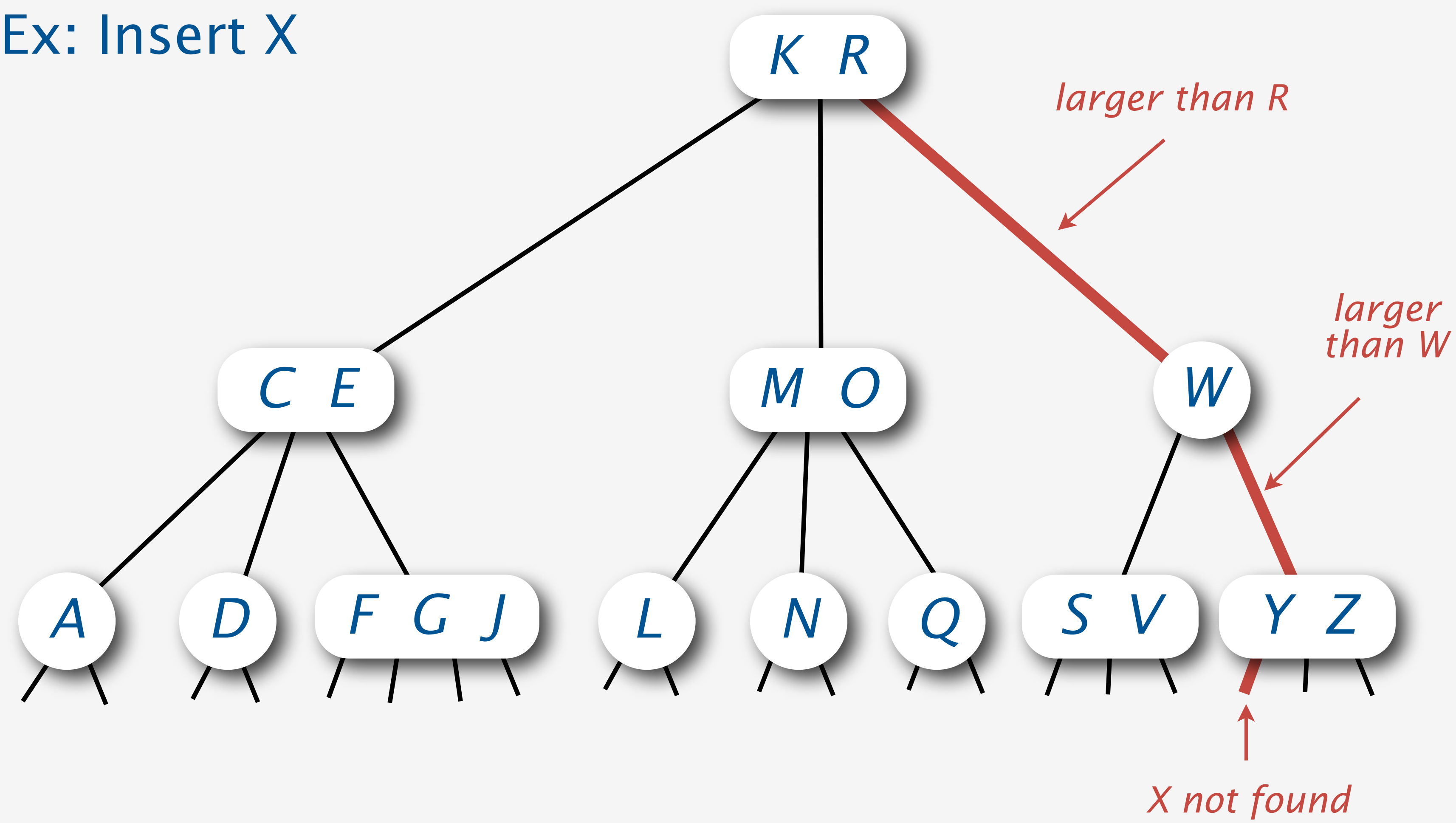
Insertion in a 2-3-4 Tree

Add new keys at the bottom of the tree.

Insert.

- Search to bottom for key.

Ex: Insert X



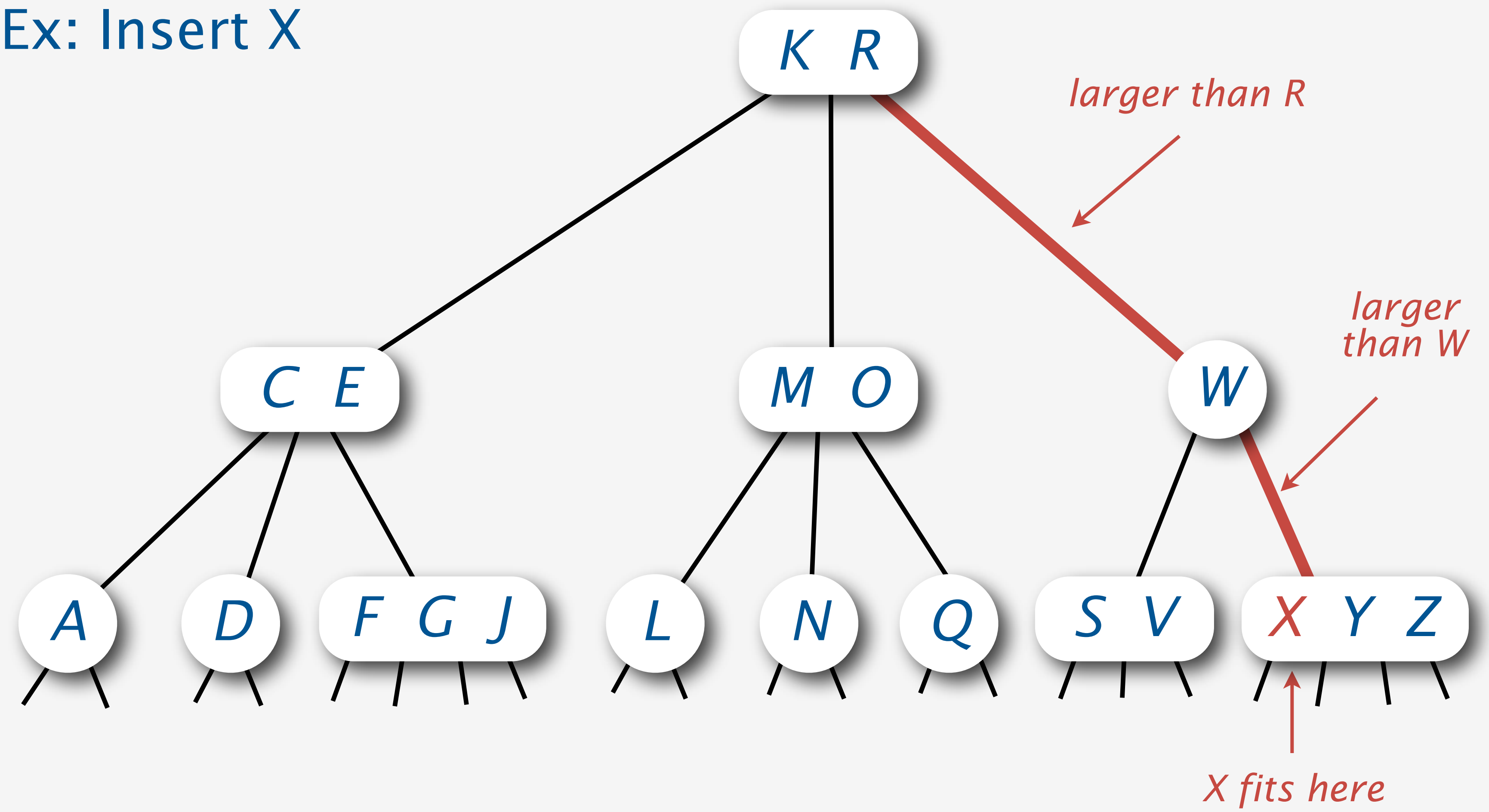
Insertion in a 2-3-4 Tree

Add new keys at the bottom of the tree.

Insert.

- Search to bottom for key.
- 3-node at bottom: convert to a 4-node.

Ex: Insert X



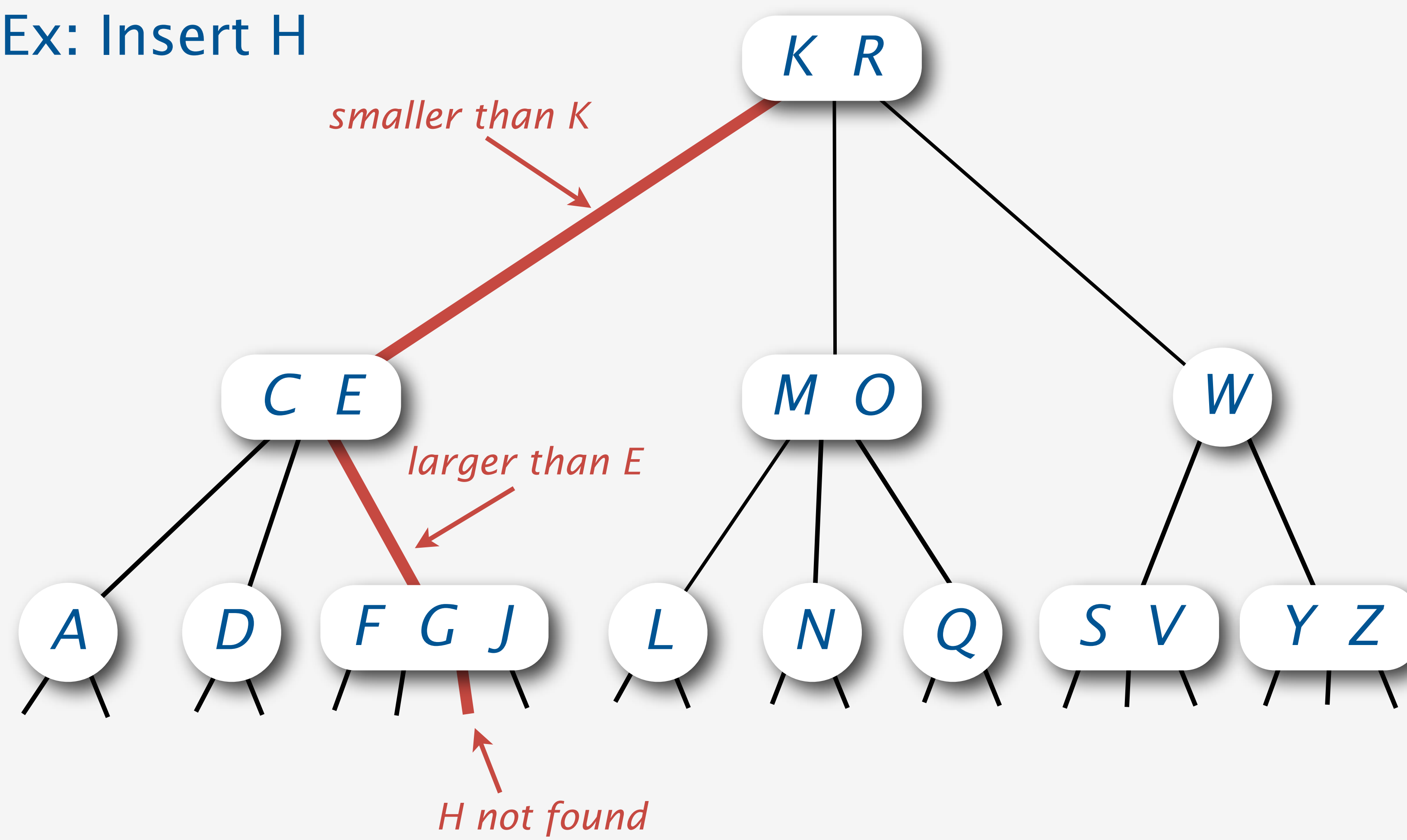
Insertion in a 2-3-4 Tree

Add new keys at the bottom of the tree.

Insert.

- Search to bottom for key.

Ex: Insert H



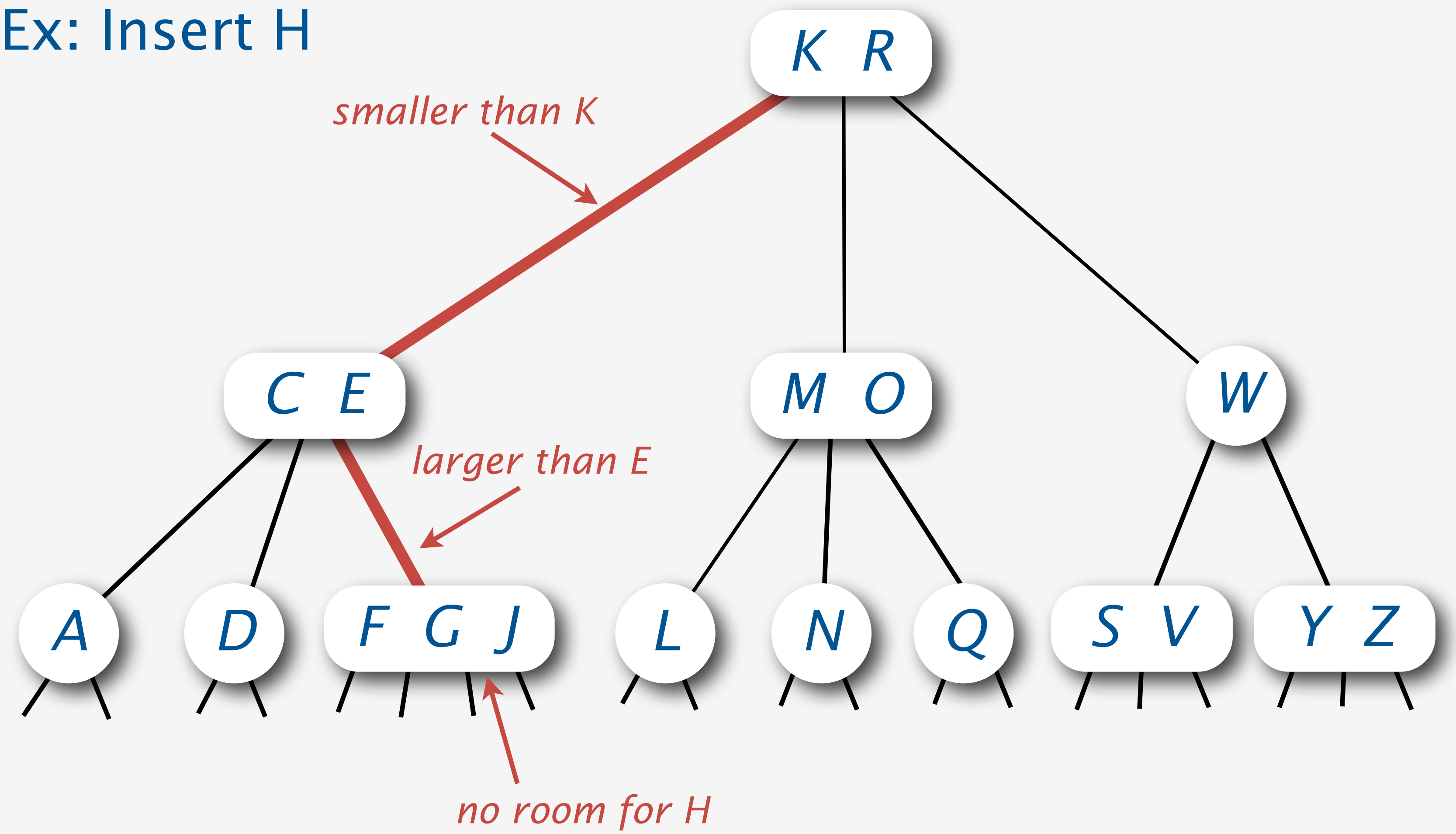
Insertion in a 2-3-4 Tree

Add new keys at the bottom of the tree.

Insert.

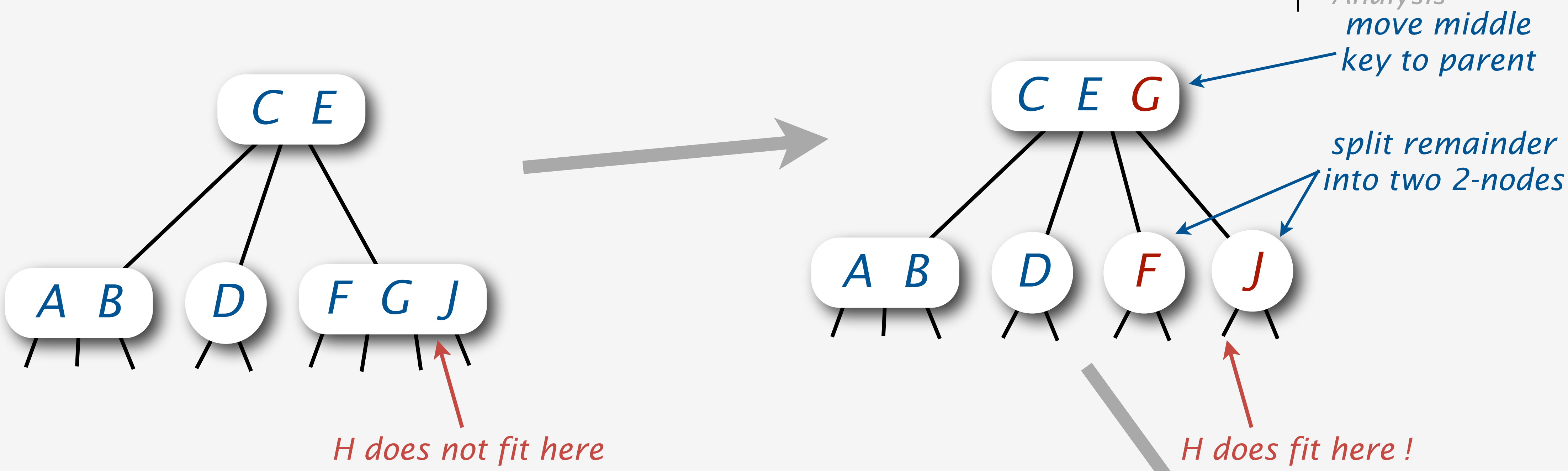
- Search to bottom for key.
- 2-node at bottom: convert to a 3-node.
- 3-node at bottom: convert to a 4-node.
- 4-node at bottom: no room for new key.

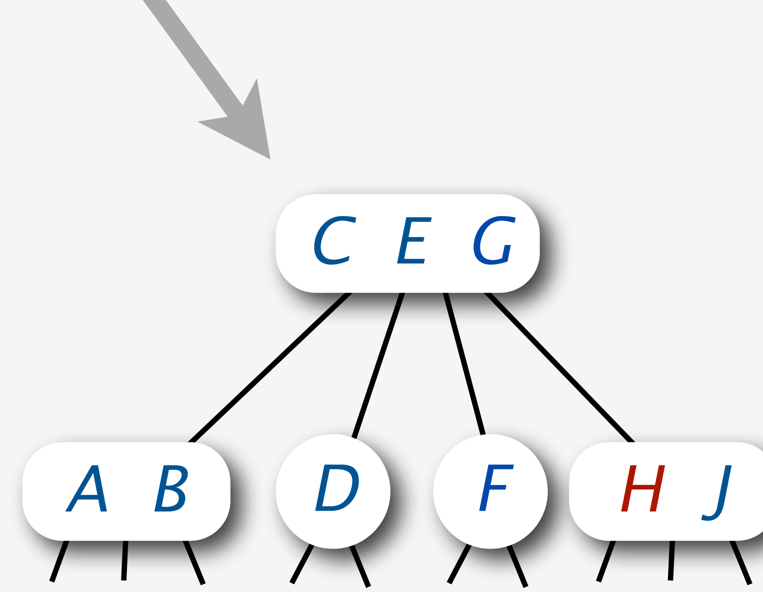
Ex: Insert H



Splitting 4-nodes in a 2-3-4 tree

is an effective way to make room for insertions





Problem: Doesn't work if parent is a 4-node

Bottom-up solution (Bayer, 1972)

- Use same method to split parent
- Continue up the tree while necessary

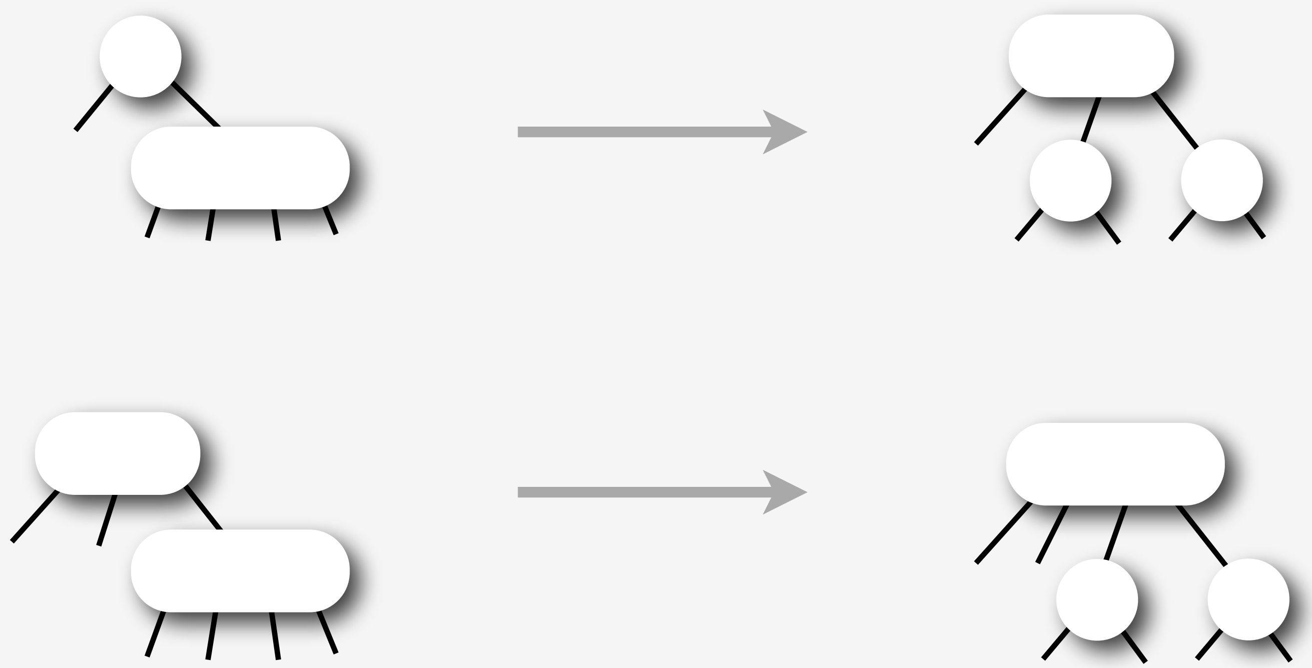
Top-down solution (Guibas-Sedgewick, 1978)

- Split 4-nodes on the way **down**
- Insert at bottom

Splitting 4-nodes on the way down

ensures that the “current” node is not a 4-node

Transformations to split 4-nodes:



local transformations that work anywhere in the tree

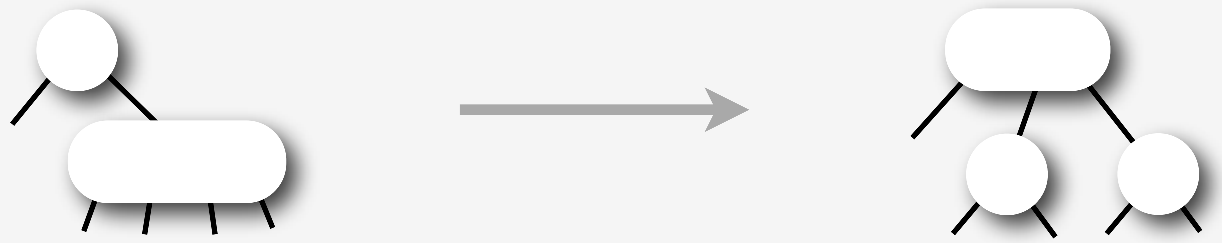
Invariant: “Current” node is not a 4-node

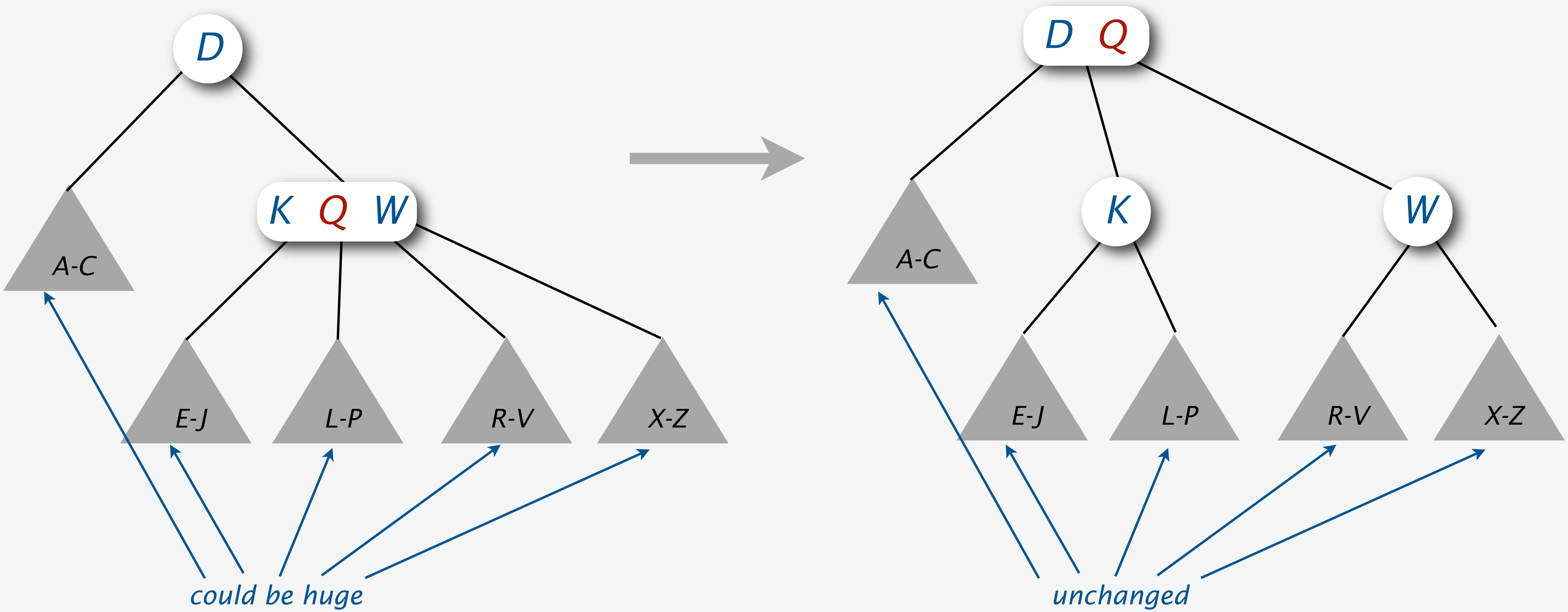
Consequences:

- 4-node below a 4-node case never happens
- Bottom node reached is always a 2-node or a 3-node

Splitting a 4-node below a 2-node

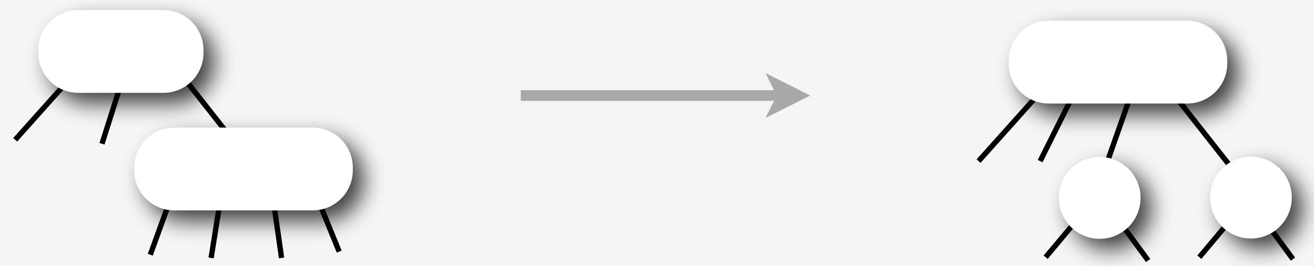
is a local transformation that works anywhere in the tree

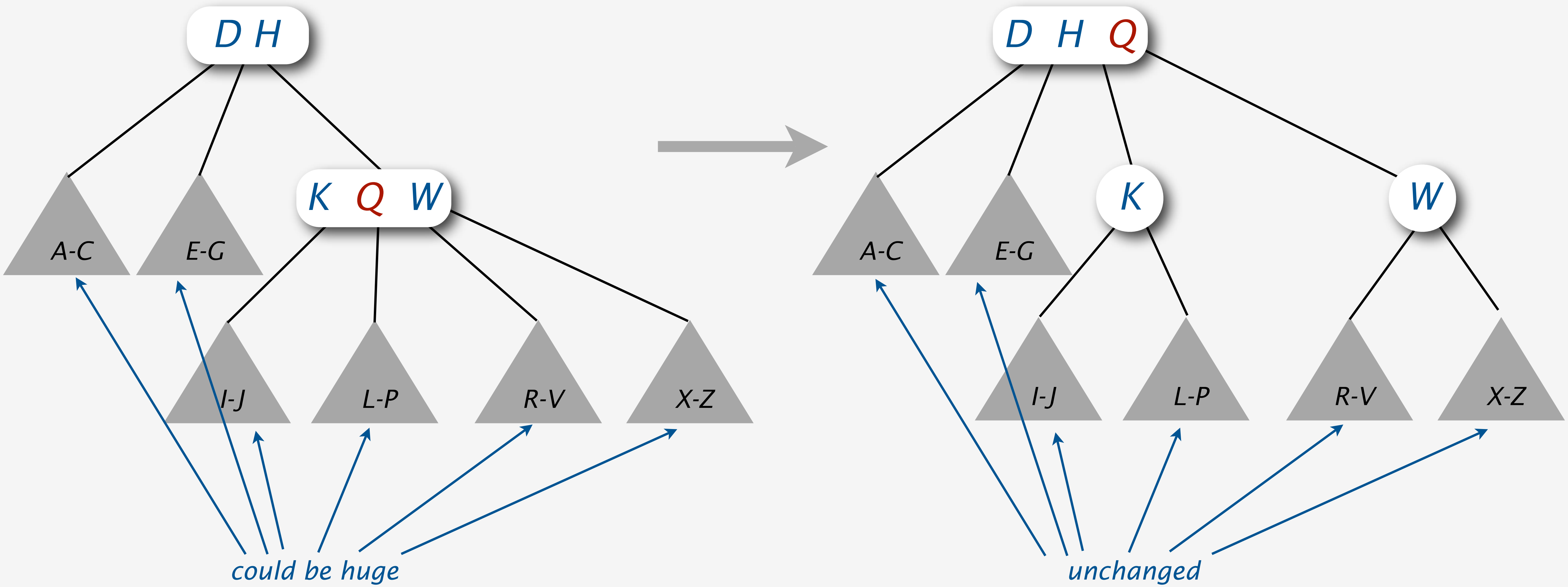




Splitting a 4-node below a 3-node

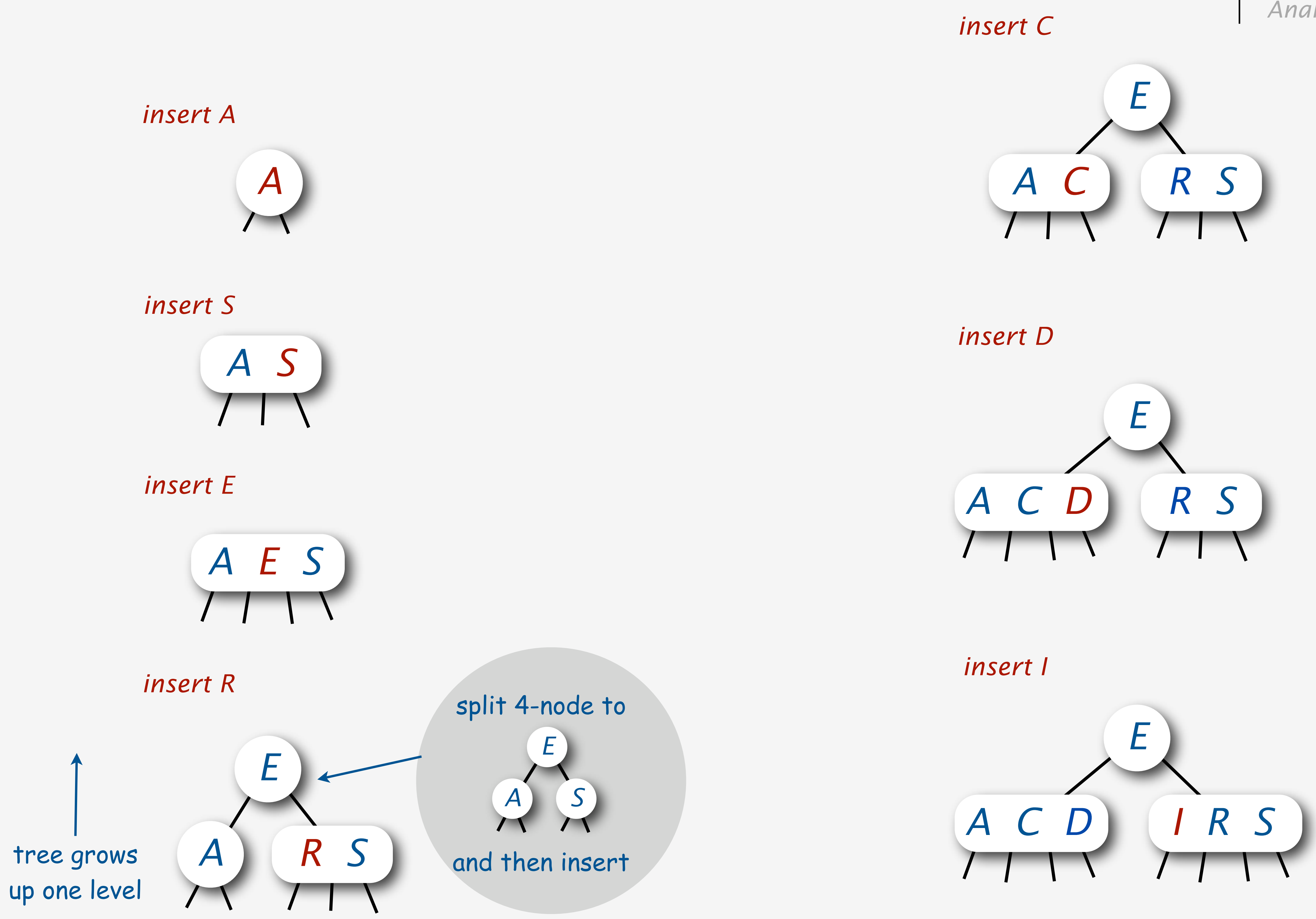
is a local transformation that works anywhere in the tree





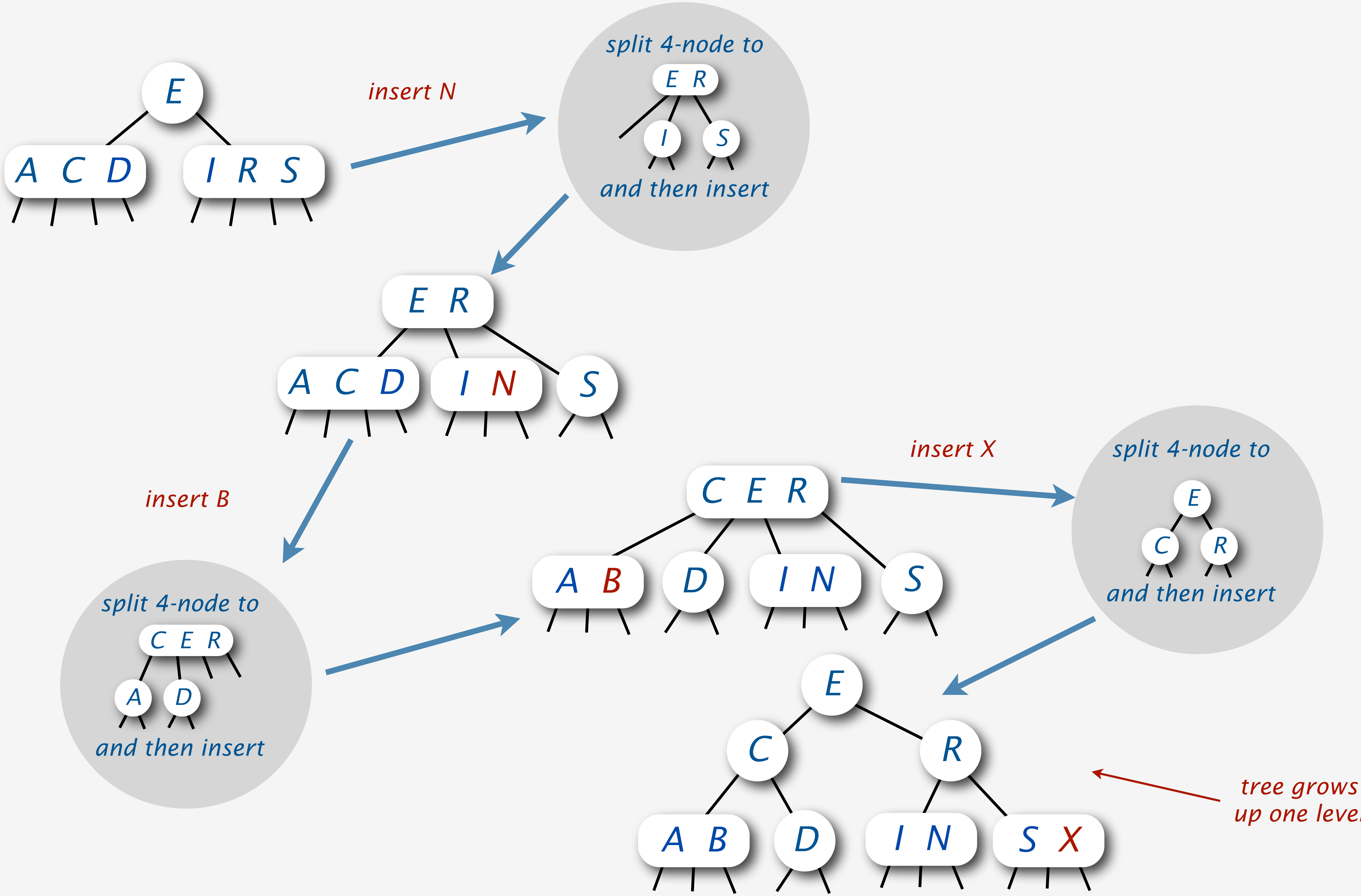
Growth of a 2-3-4 tree

happens upwards from the bottom



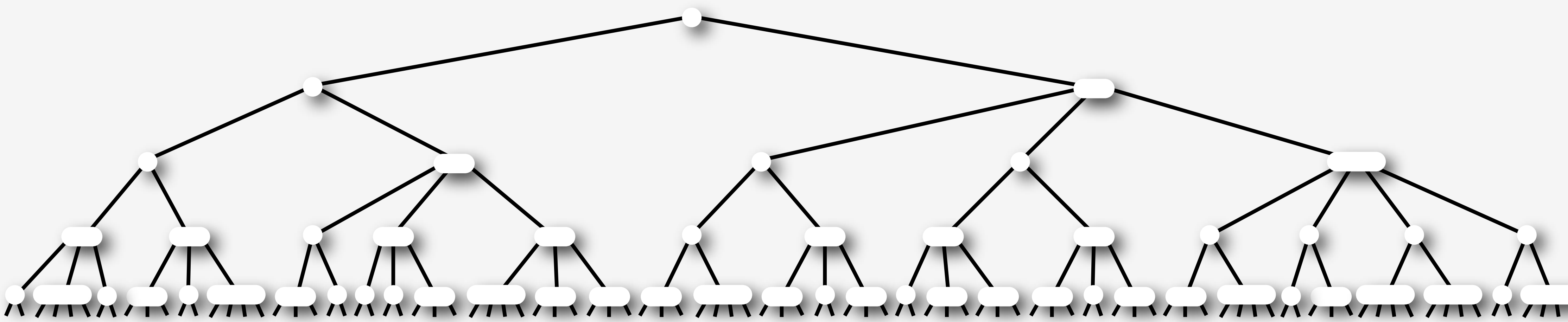
Growth of a 2-3-4 tree (continued)

happens upwards from the bottom



Balance in 2-3-4 trees

Key property: All paths from root to leaf are the same length



Tree height.

- Worst case: lg N [all 2-nodes]
- Best case: log4 N = 1/2 lg N [all 4-nodes]
- Between 10 and 20 for 1 million nodes.
- Between 15 and 30 for 1 billion nodes.

Guaranteed logarithmic performance for both search and insert.

Direct implementation of 2-3-4 trees

is complicated because of code complexity.

Maintaining multiple node types is cumbersome.

- Representation?
- Need multiple compares to move down in tree.
- Large number of cases for splitting.
- Need to convert 2-node to 3-node and 3-node to 4-node.

fantasy code

```
private void insert(Key key, Val val)
{
   Node x = root;
   while (x.getTheCorrectChild(key) != null)
   {
      x = x.getTheCorrectChild(key);
      if (x.is4Node()) x.split();
   }
   if      (x.is2Node()) x.make3Node(key, val);
   else if (x.is3Node()) x.make4Node(key, val);
   return x;
}
```

Bottom line: Could do it, but stay tuned for an easier way.

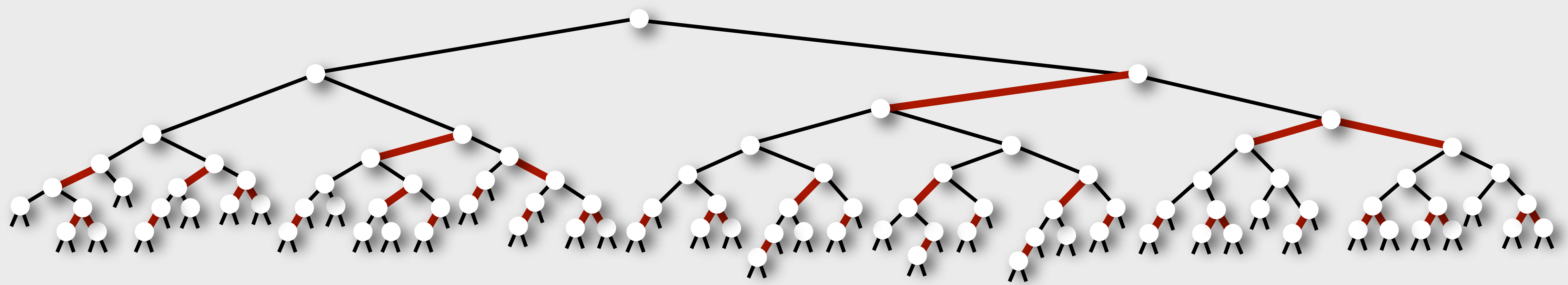
Introduction

2-3-4 Trees

LLRB Trees

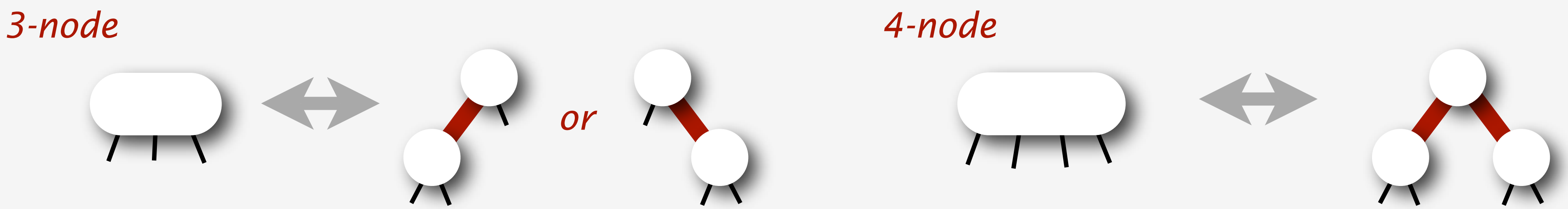
Deletion

Analysis



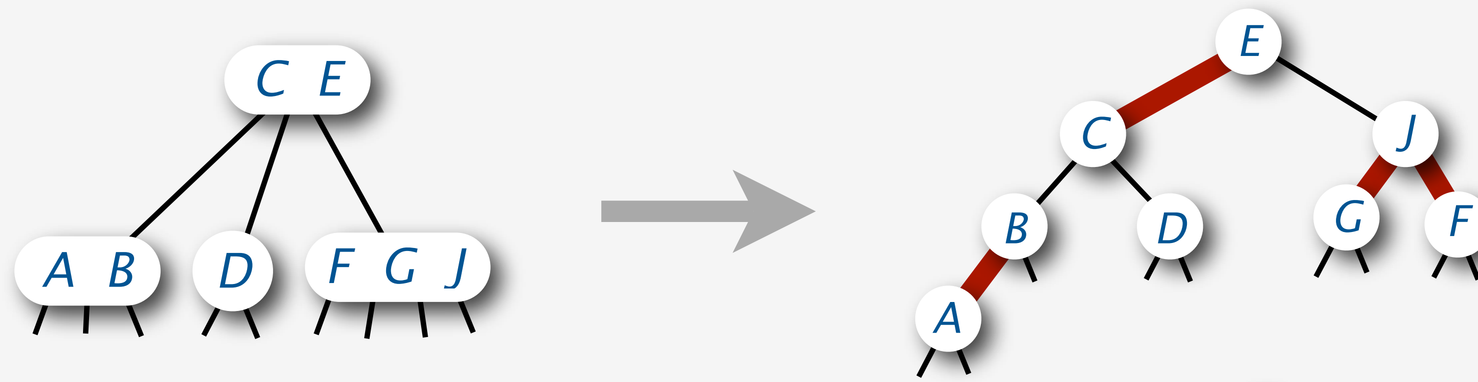
Red-black trees (Guibas-Sedgewick, 1978)

1. Represent 2-3-4 tree as a BST.
2. Use "internal" red edges for 3- and 4- nodes.

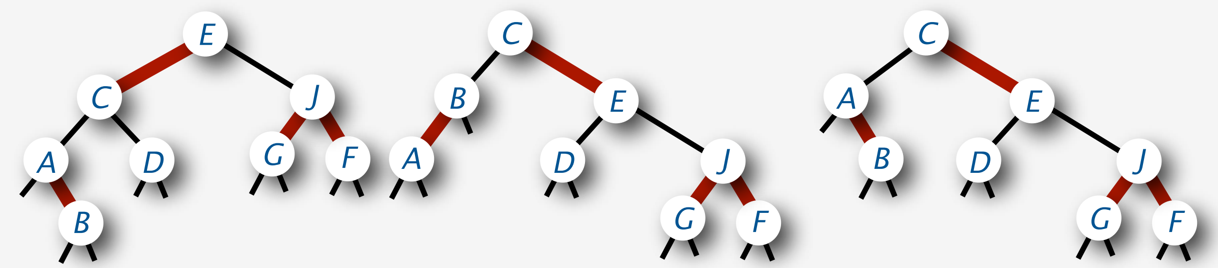


Key Properties

- elementary BST search works
- easy to maintain a correspondence with 2-3-4 trees (and several other types of balanced trees)



Note: correspondence is not 1-1.
(3-nodes can lean either way)

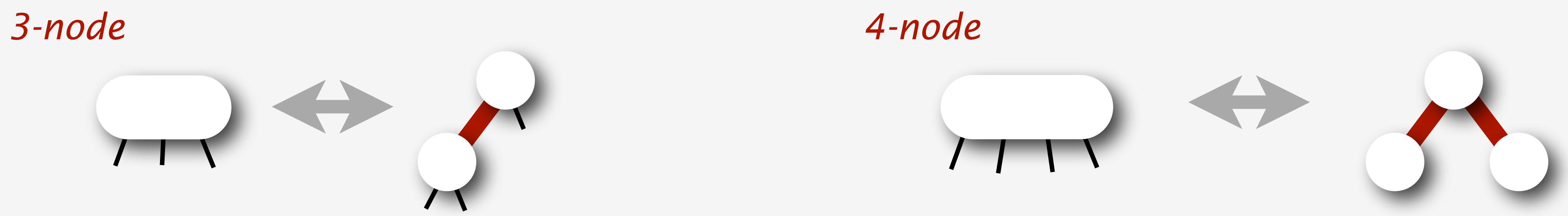


Many variants studied (details omitted.)

NEW VARIANT (this talk): Left-leaning red-black trees

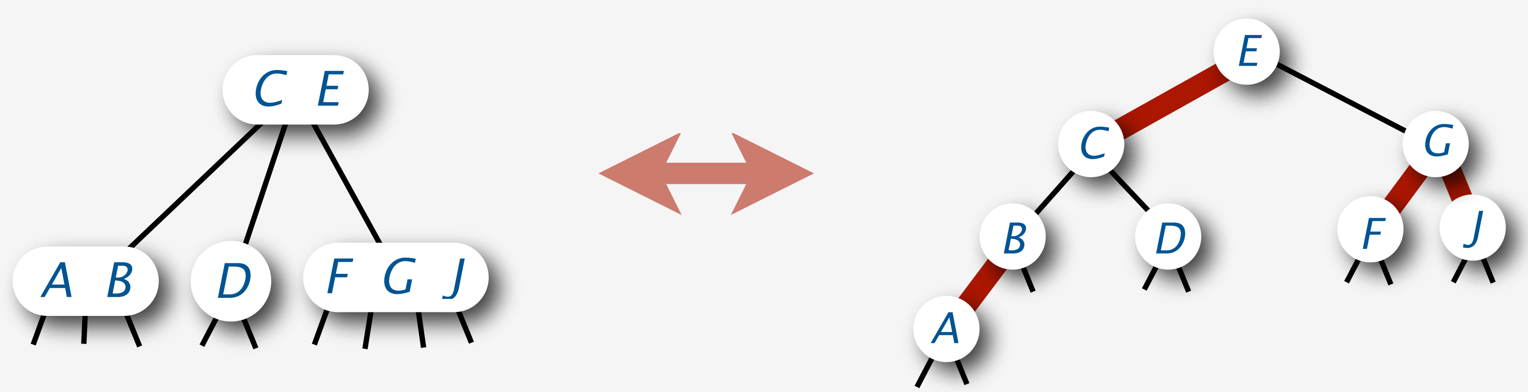
Left-leaning red-black trees

1. Represent 2-3-4 tree as a BST.
2. Use "internal" red edges for 3- and 4- nodes.
3. Require that 3-nodes be left-leaning.



Key Properties

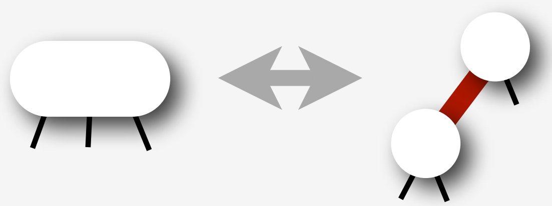
- elementary BST search works
- easy-to-maintain 1-1 correspondence with 2-3-4 trees
- trees therefore have perfect black-link balance



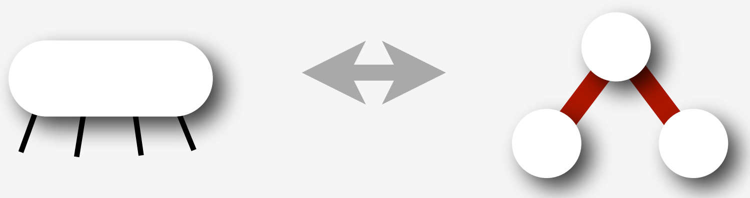
Left-leaning red-black trees

1. Represent 2-3-4 tree as a BST.
2. Use "internal" red edges for 3- and 4- nodes.
3. Require that 3-nodes be left-leaning.

3-node

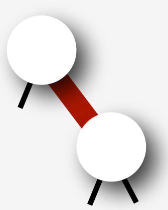


4-node



Disallowed

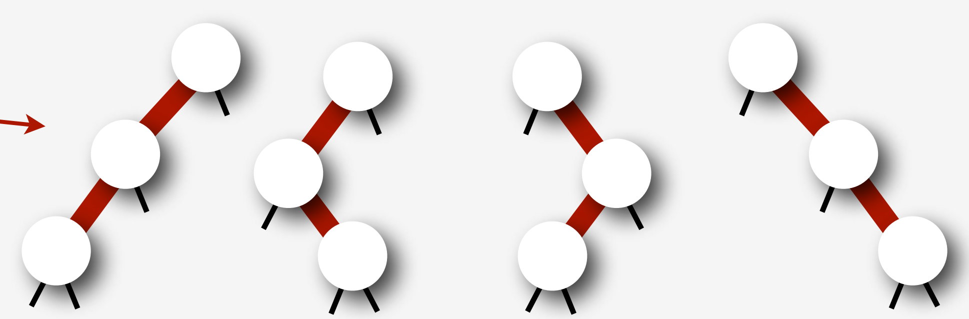
- right-leaning 3-node representation



standard red-black trees allow this one

- two reds in a row

original version of left-leaning trees used this 4-node representation



single-rotation trees allow all of these