CS062

DATA STRUCTURES AND ADVANCED PROGRAMMING

19: 2-3 Search Trees

Alexandra Papoutsaki
she/her/hers

Lecture 19: 2-3 Search Trees

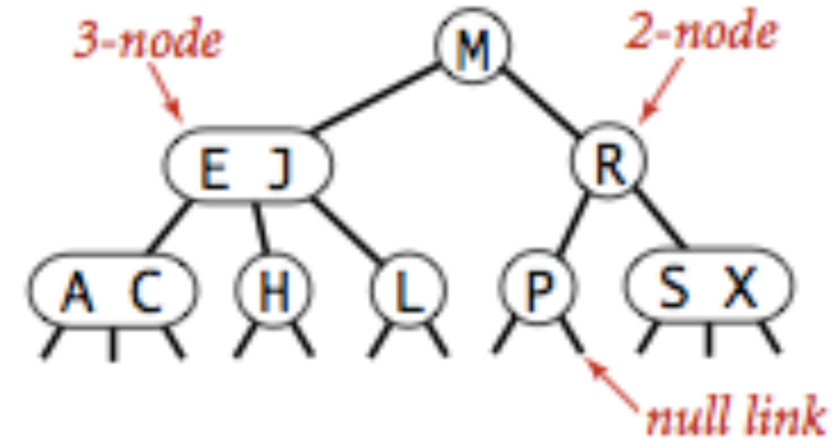
- 2-3 Search Trees
- Search
- Insertion
- Construction
- Performance

Visualization of insertion into a binary search tree

- 255 insertions in random order.

Order of growth for dictionary operations

	Worst case			Average case		
	Search	Insert	Delete	Search	Insert	Delete
BST	n	n	n	$\log n$	$\log n$	$\sqrt{n}$
Goal	$\log n$	$\log n$	$\log n$	$\log n$	$\log n$	$\log n$



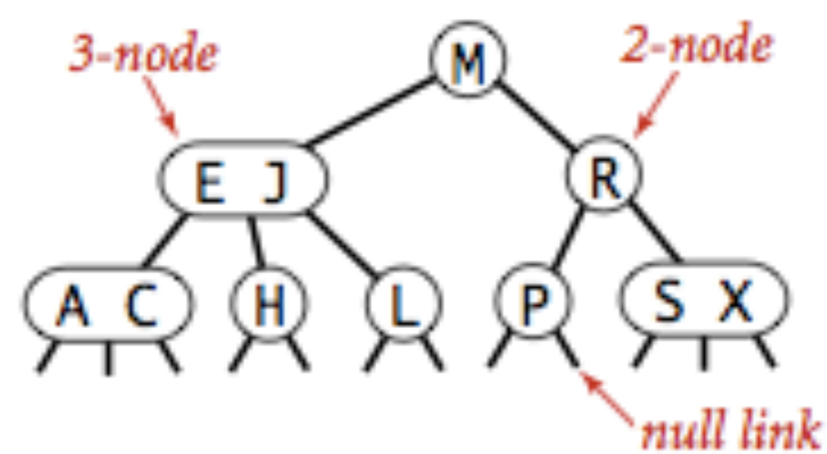
Anatomy of a 2-3 search tree

2-3 tree

- Definition: A 2-3 tree is either empty or a
  - 2-node: one key (and associated value) and two links, a left to a 2-3 search tree with smaller keys, and a right to a 2-3 search tree with larger keys (similarly to standard BSTs), or a
  - 3-node: two keys (and associated values) and three links, a left to a 2-3 search tree with smaller keys than first key, a middle to a 2-3 search tree with keys between the node's keys, and a right to a 2-3 search tree with larger keys than the second key.
- Symmetric order: In-order traversal yields keys in ascending order.
- Perfect balance: Every path from root to null link (empty tree) has the same length.

Example of a 2-3 tree

- 2-node, business as usual with BSTs.
  - (e.g., EJ are smaller than M and R is larger than M).
- In 3-node,
  - left link points to 2-3 search tree with smaller keys than first key,
    - (e.g., AC are smaller than E.)
  - middle link points to 2-3 search tree with keys between first and second key,
    - (e.g. H is between E and J.)
  - right link points to 2-3 search tree with keys larger than second key.
    - (e.g, L is larger than J).



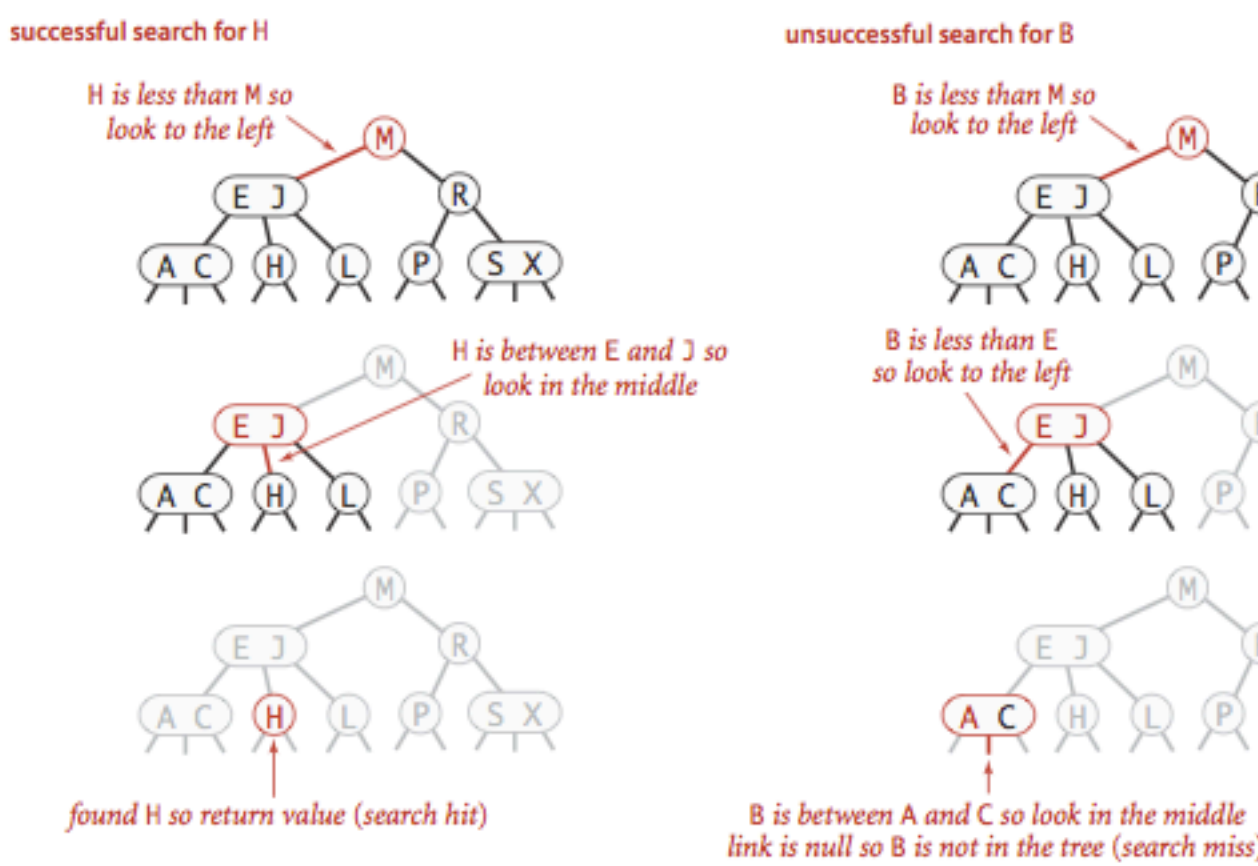
Anatomy of a 2-3 search tree

Lecture 19: 2-3 Search Trees

- 2-3 Search Trees
- Search
- Insertion
- Construction
- Performance

How to search for a key

- Compare search key against (every) key in node.
- Find interval containing search key (left, potentially middle, or right).
- Follow associated link, recursively.



Search hit (left) and search miss (right) in a 2-3 tree

http://algs4.cs.princeton.edu

3.3 2–3 Tree Demo

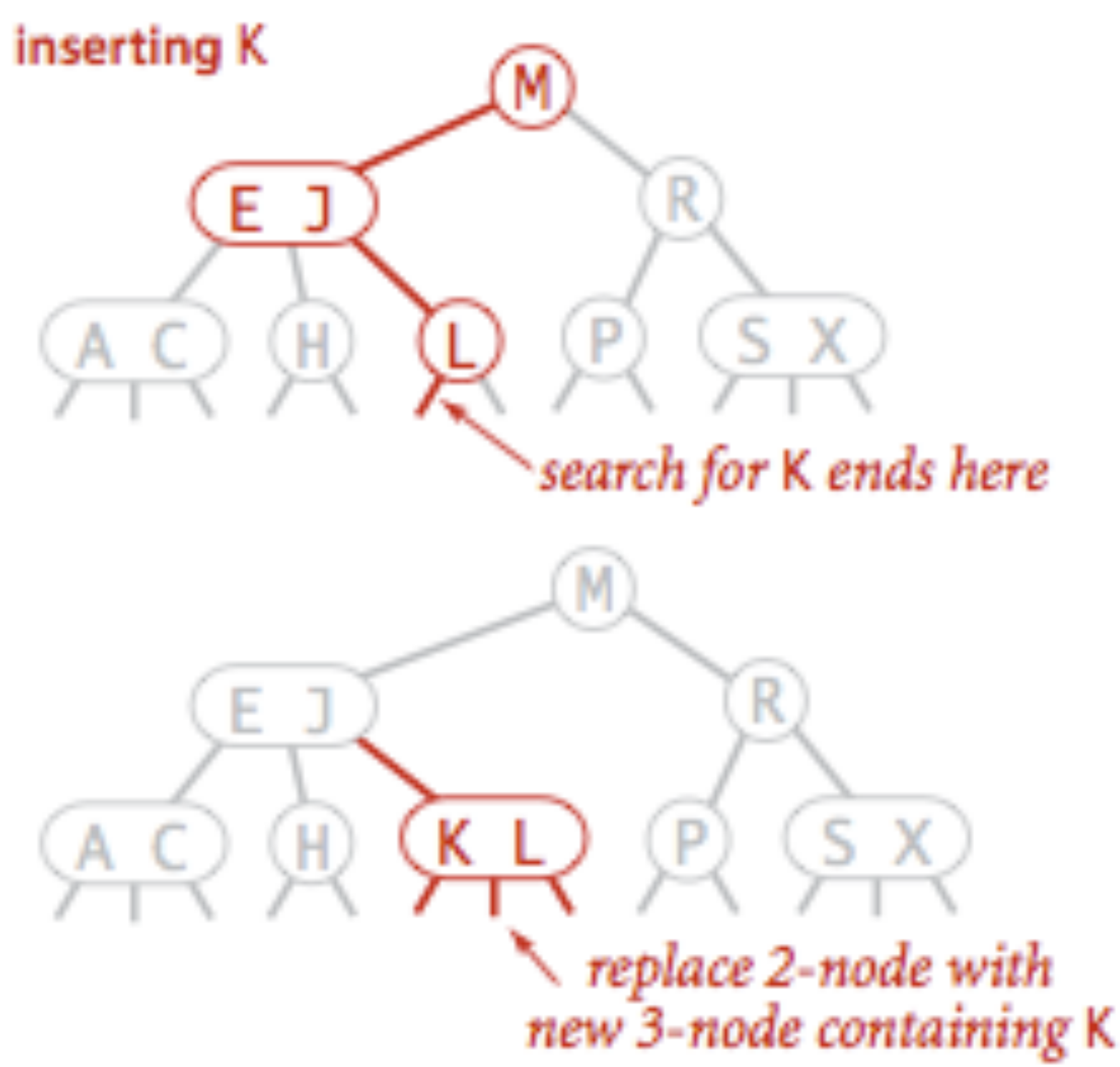
- *search*
- *insertion*
- *construction*

Lecture 19: 2-3 Search Trees

- 2-3 Search Trees
- Search
- Insertion
- Construction
- Performance

How to insert into a 2-node at bottom

- Search for key and add new key to 2-node to create a 3-node.



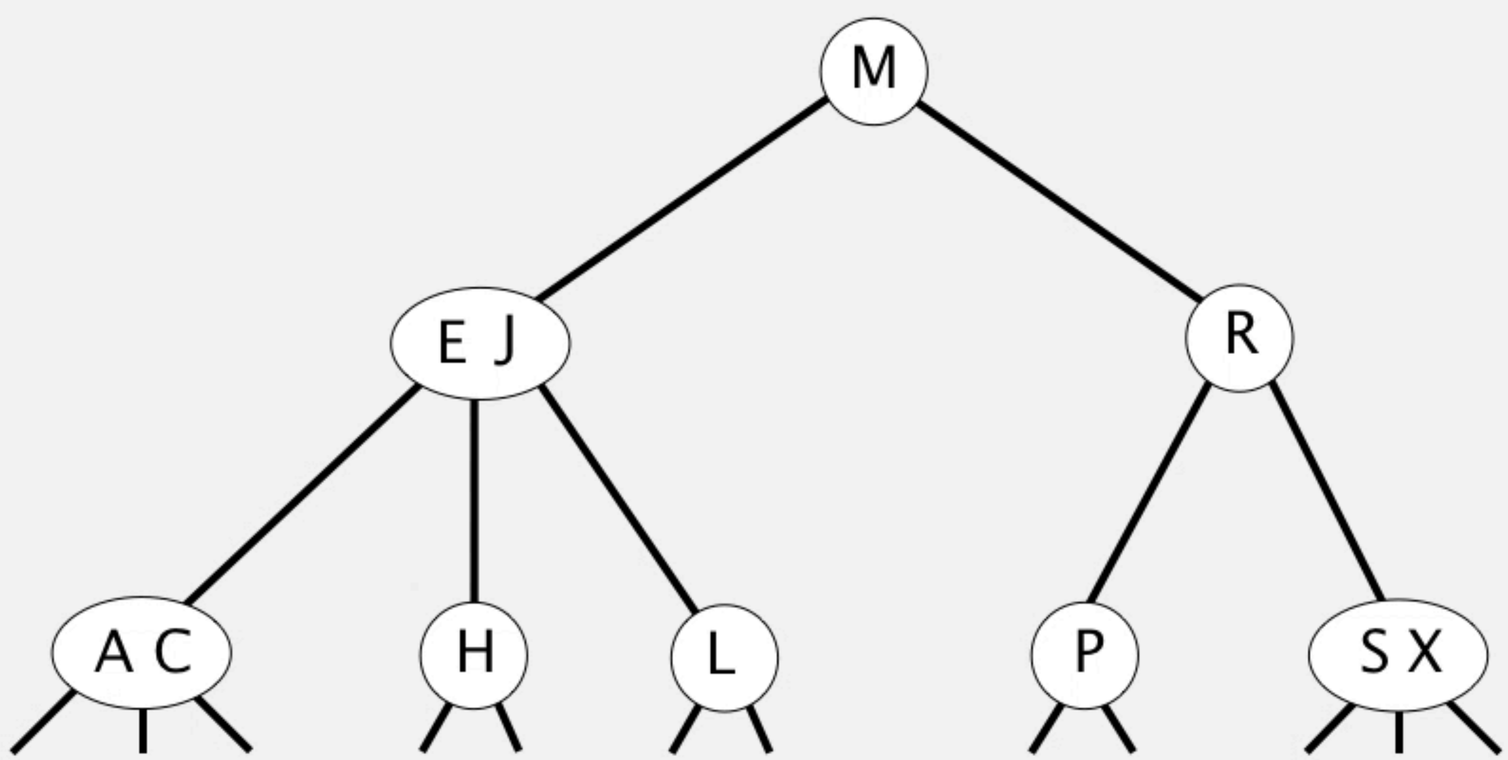
Insert into a 2-node

2-3 tree demo: insertion

Insert into a 2-node at bottom.

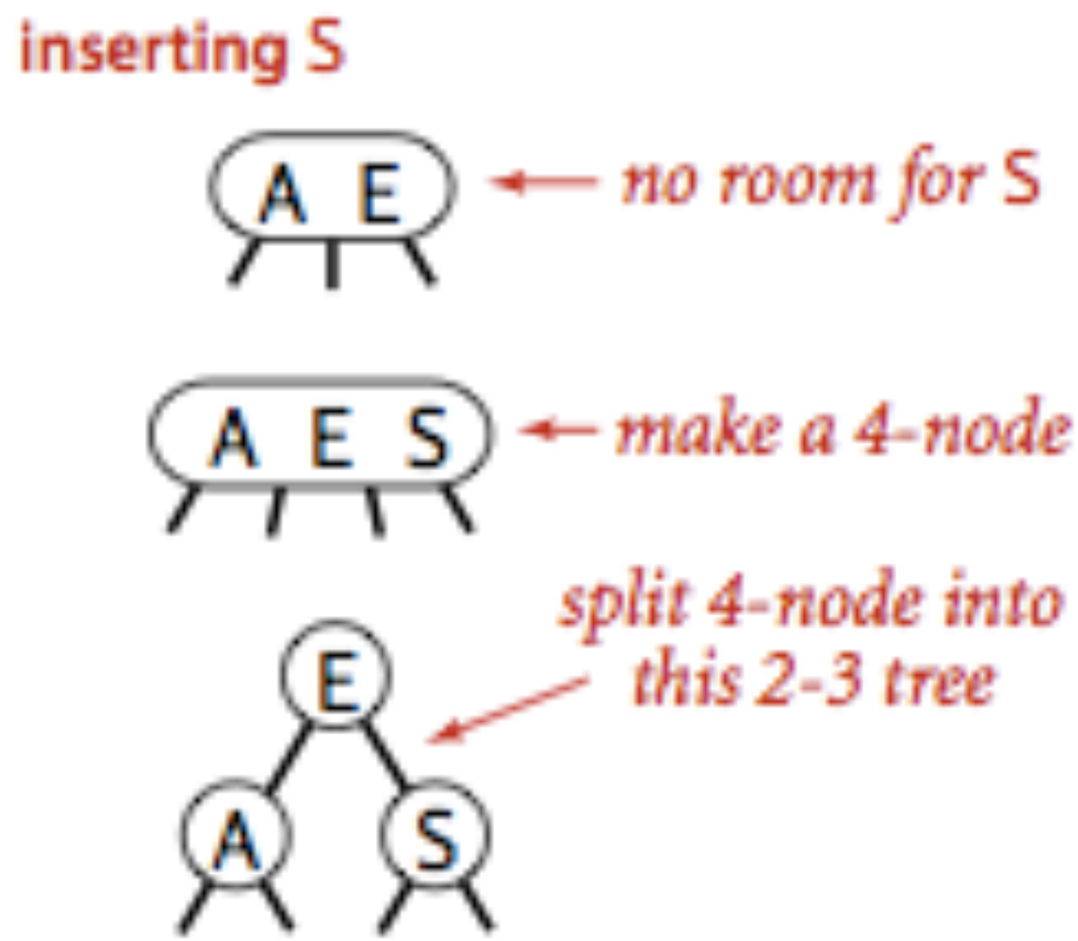
- Search for key, as usual.
- Replace 2-node with 3-node.

insert K



How to insert into a tree consisting of a single 3-node

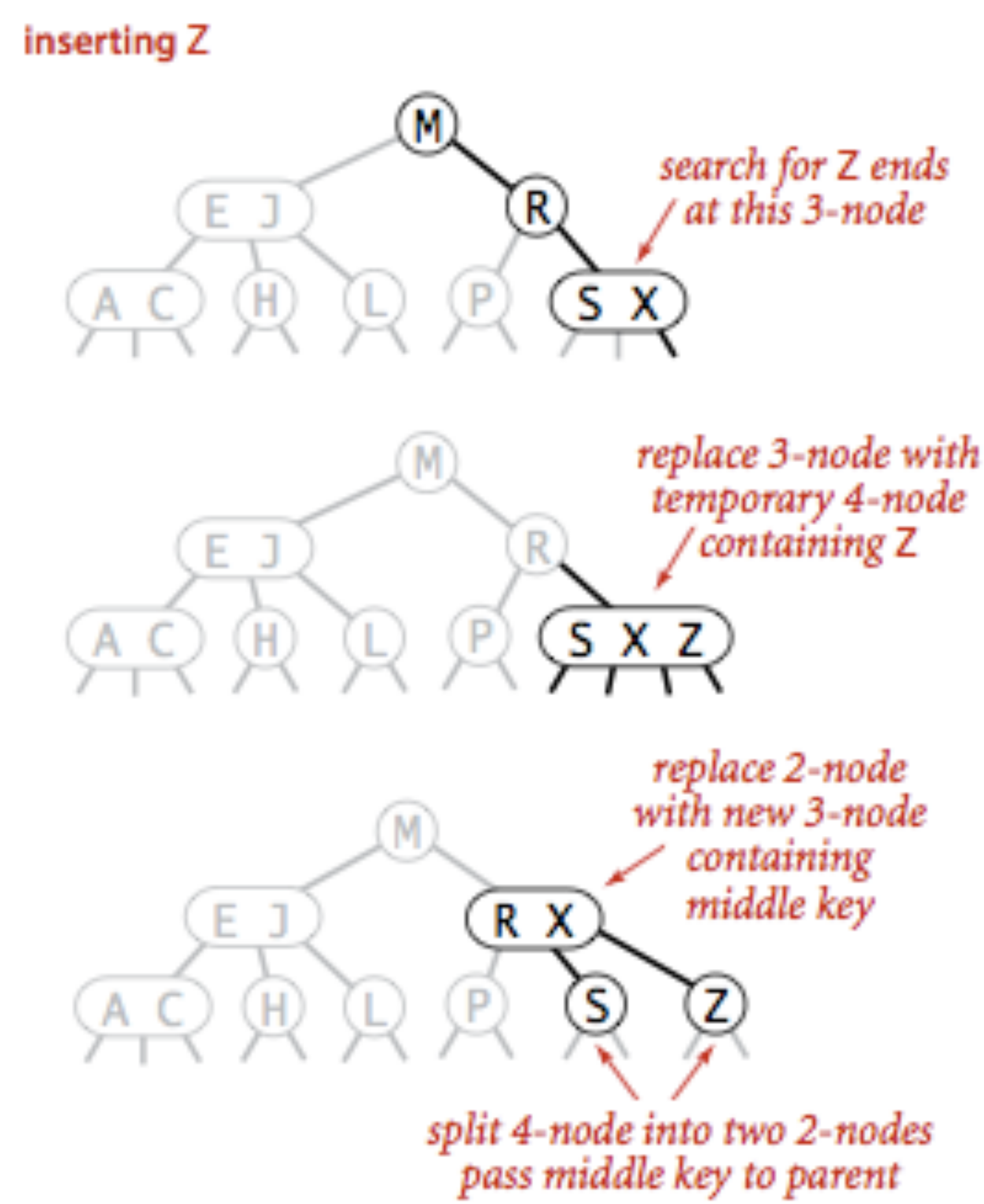
- Add new key to 3-node to create a temporary 4-node.
- Move middle key in 4-node into parent.
- Split 4-node into two 2-nodes.
- Height went up by 1.



Insert into a single 3-node

How to insert into a 3-node whose parent is a 2-node

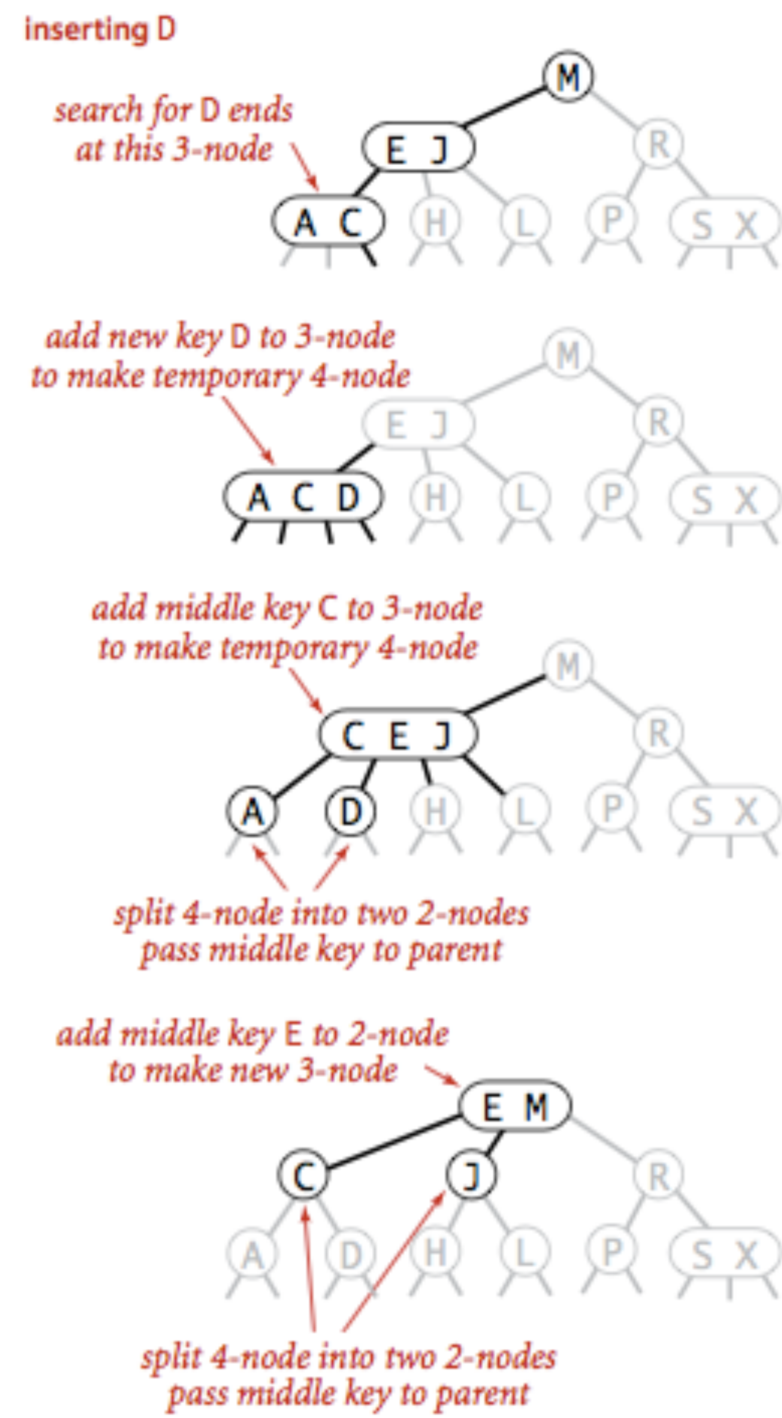
- Add new key to 3-node to create a temporary 4-node.
- Split 4-node into two 2-nodes and pass middle key to parent.
- Replace 2-node parent with 3-node.



Insert into a 3-node whose parent is a 2-node

How to insert into a 3-node whose parent is a 3-node

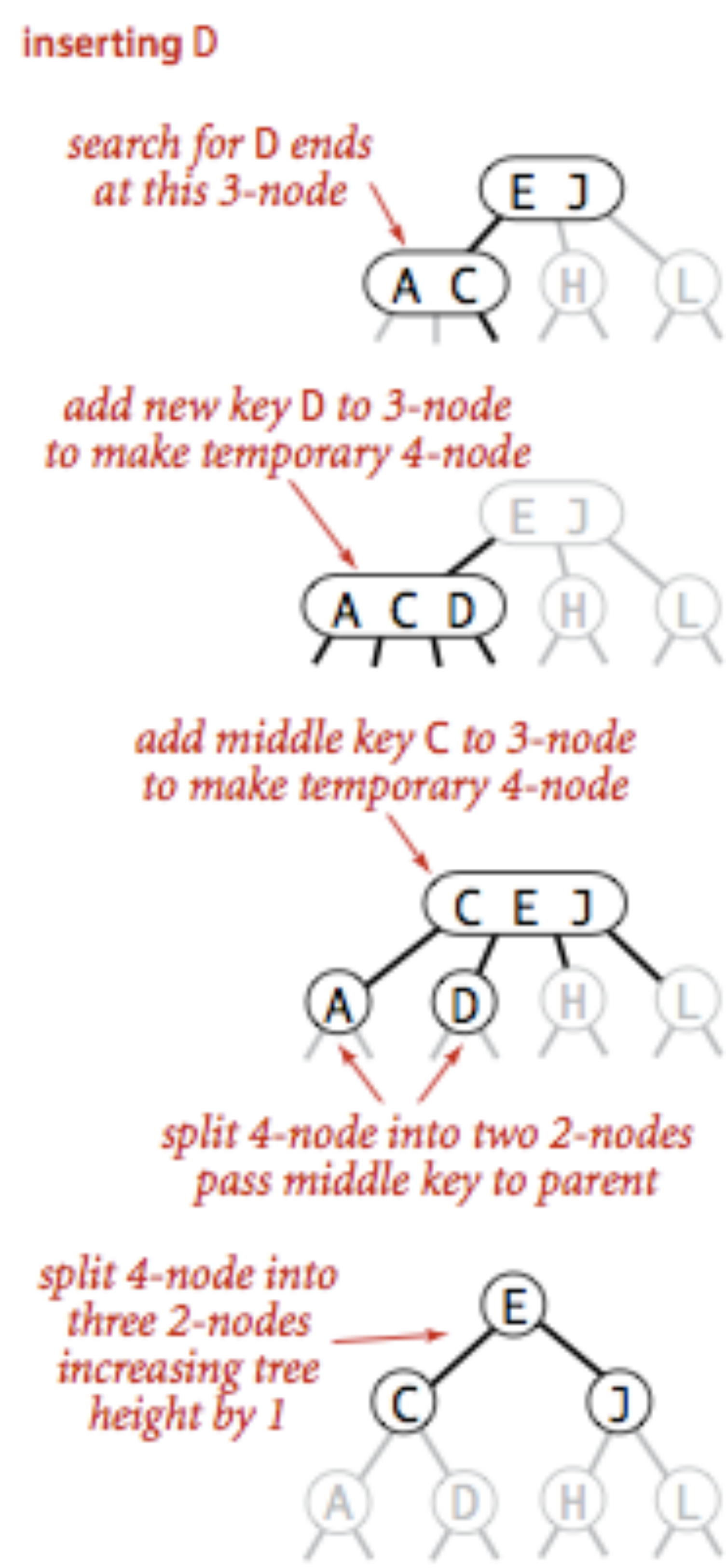
- Add new key to 3-node to create a temporary 4-node.
- Split 4-node into two 2-nodes and pass middle key to parent creating a temporary 4-node.
- Split 4-node into two 2-nodes and pass middle key to parent.
- Repeat up the tree, as necessary.



Insert into a 3-node whose parent is a 3-node

Splitting the root

- If end up with a temporary 4-node root, split into three 2-nodes.
- Increases height by 1 but perfect balance is preserved.



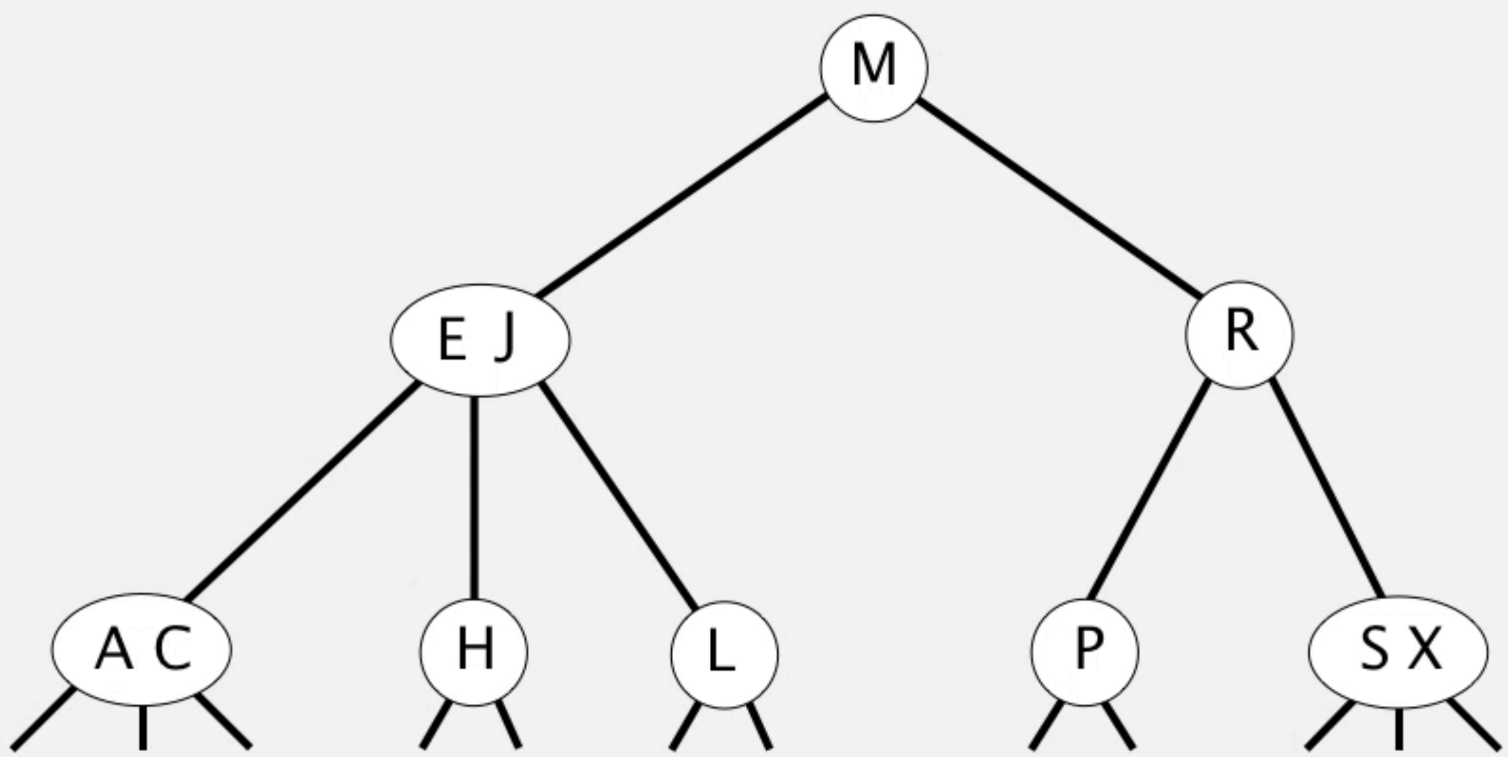
Splitting the root

2-3 tree demo: insertion

Insert into a 2-node at bottom.

- Search for key, as usual.
- Replace 2-node with 3-node.

insert K

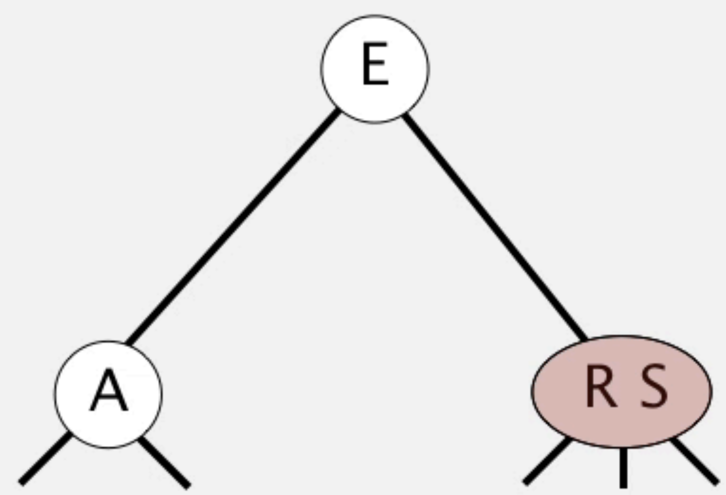


Lecture 19: 2-3 Search Trees

- 2-3 Search Trees
- Search
- Insertion
- Construction
- Performance

2-3 tree demo: construction

insert R

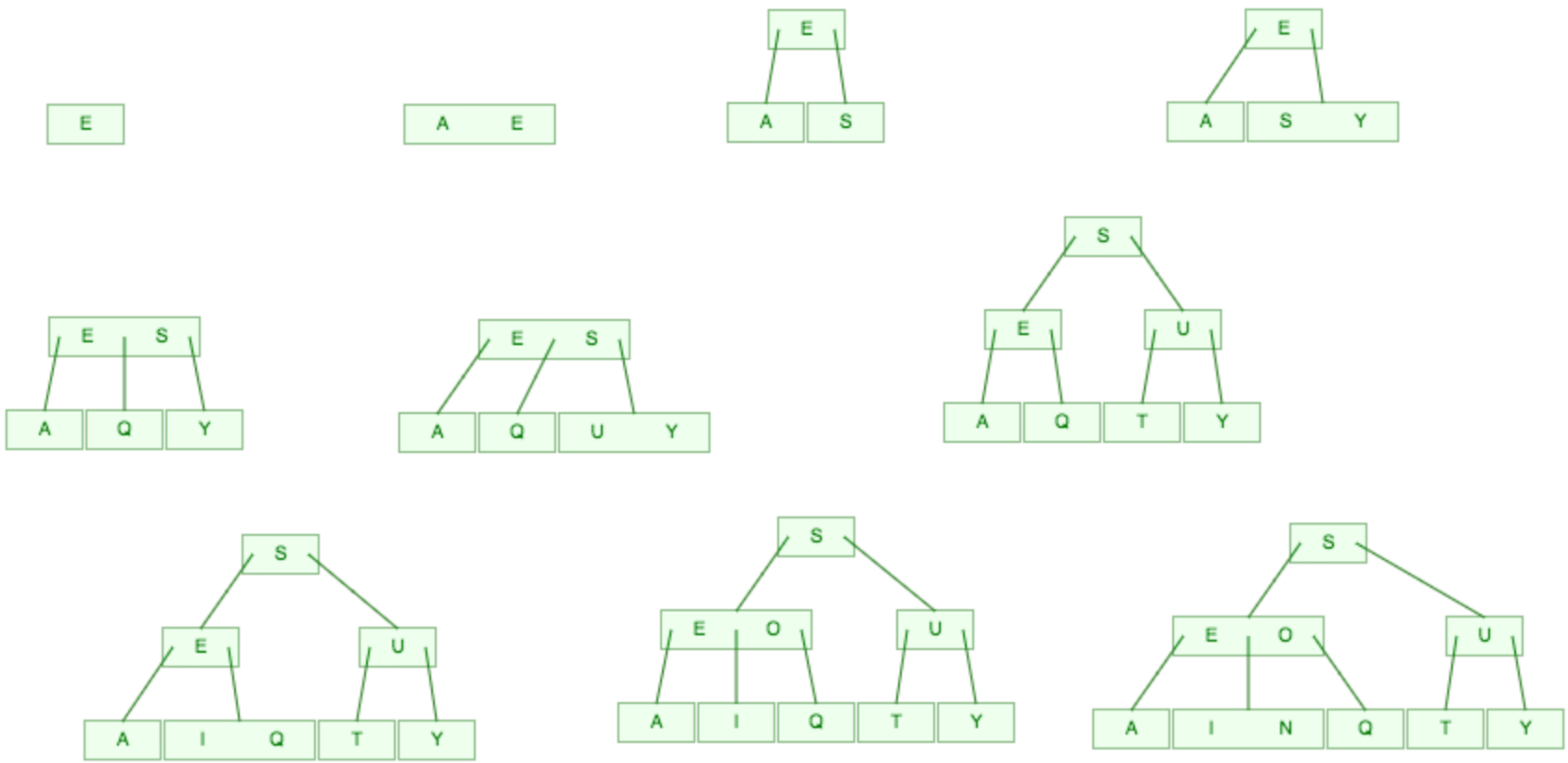


Practice Time - Worksheet #19

- Draw the 2-3 tree that results when you insert the keys:
  E A S Y Q U T I O N in that order in an initially empty tree.

ANSWER

- E A S Y Q U T I O N



https://www.cs.usfca.edu/~galles/visualization/BTree.html

Lecture 19: 2-3 Search Trees

- 2-3 Search Trees
- Search
- Insertion
- Construction
- Performance

Height of 2-3 search trees

- Worst case: $\log n$ (all 2-nodes).
- Best case: $\log_3 n = 0.631 \log n$ (all 3-nodes)
  - That means that storing a million nodes will lead to a tree with height between 12 and 20, and storing a billion nodes to a tree with height between 19 and 30 (not bad!).
- Search and insert are $O(\log n)$!
- But implementation is a pain and the overhead incurred could make the algorithms slower than standard BST search and insert.
- We did provide insurance against a worst case but we would prefer the overhead cost for that insurance to be low. Stay tuned!

Summary for dictionary operations

	Worst case			Average case		
	Search	**Insert**	**Delete**	**Search**	**Insert**	**Delete**
BST	n	n	n	$\log n$	$\log n$	$\sqrt{n}$
2-3 search trees	$\log n$	$\log n$	$\log n$	$\log n$	$\log n$	$\log n$

Readings:

- Recommended Textbook: Chapter 3.3 (Pages 424-447)
- Website:
  - https://algs4.cs.princeton.edu/33balanced/
- Visualization:
  - https://www.cs.usfca.edu/~galles/visualization/BTree.html (for 2-3 trees)

Worksheet:

- Lecture 19 worksheet

Problem 1 (Problem 3.3.2 in the book)

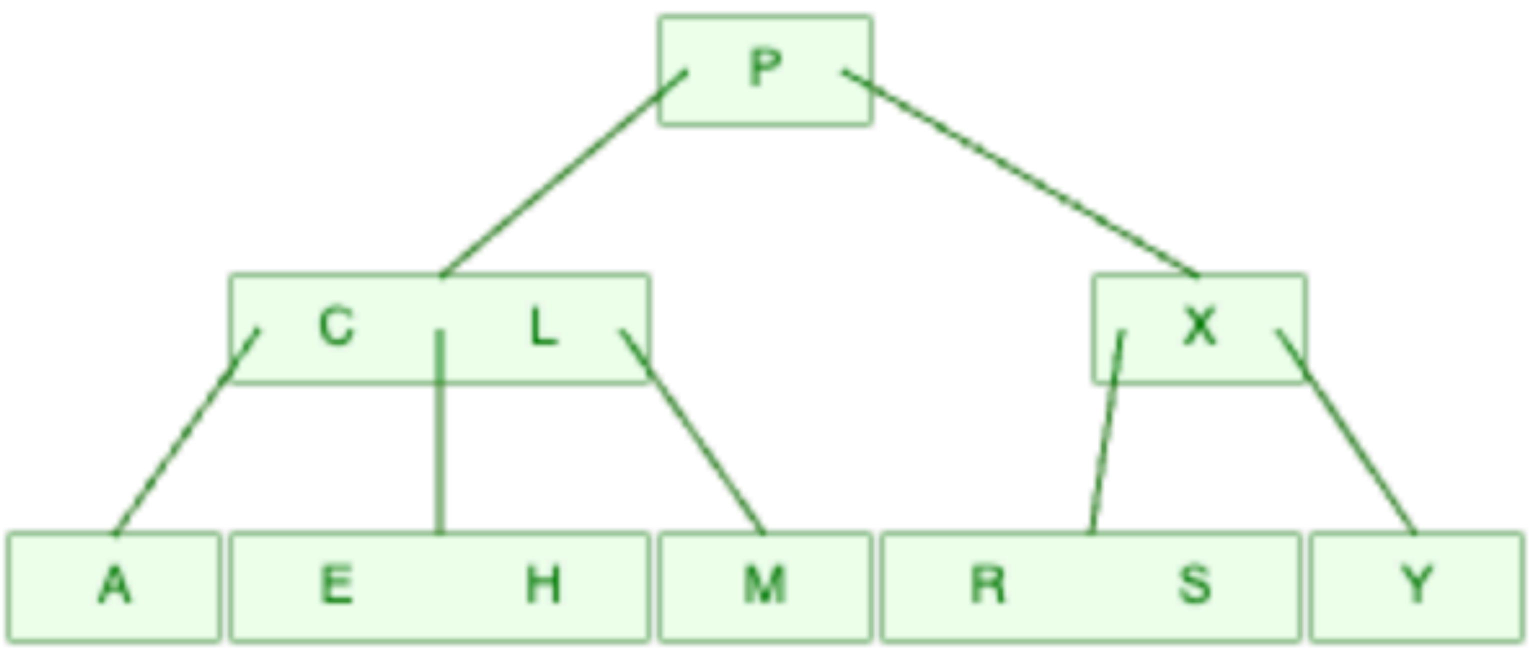
- Draw the 2-3 tree that results when you insert the keys Y, L, P, M, X, H, C, R, A, E, S) in that order into an initially empty tree.

Problem 2 (Problem 3.3.3 in the book)

- Find an insertion order for the keys S, E, A, R, C, H, X, M that leads to a 2-3 search tree of height 1.

ANSWER 1 (Problem 3.3.2 in the book)

- Draw the 2-3 tree that results when you insert the keys Y, L, P, M, X, H, C, R, A, E, S) in that order into an initially empty tree.



ANSWER 2 (Problem 3.3.3 in the book)

- Find an insertion order for the keys S, E, A, R, C, H, X, M that leads to a 2-3 search tree of height 1.
- Insertion order: E A M X R C H S
-