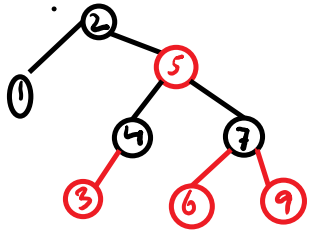


Habib University
shaping futures

CS 201 Data Structure II (L2 / L5)

2-4 Trees

Section 11.5, Data Structures and Algorithms in Python

Muhammad Qasim Pasta

qasim.pasta@sse.habib.edu.pk

Slides are designed to be filled during the lectures. Some details are intentionally mentioned to be discussed in the class. These slides should not be used as reading.

B-Trees (or multiway Trees)

A B-Tree of order k is a $k - way$ tree such that

- All leaf nodes are at the same level
  - All non-leaf nodes (except the root) have at most k children and at least $\frac{k}{2}$ children.
- The number of keys (values)
  - Non-Leaf Nodes: at least one less than the number of children, at most: $k - 1$
  - Leaf-Nodes: at least $\frac{k}{2}$
- Each internal node stores an ordered set of keys (sorted)
- The root may have as few as 2 children unless the tree is the root alone

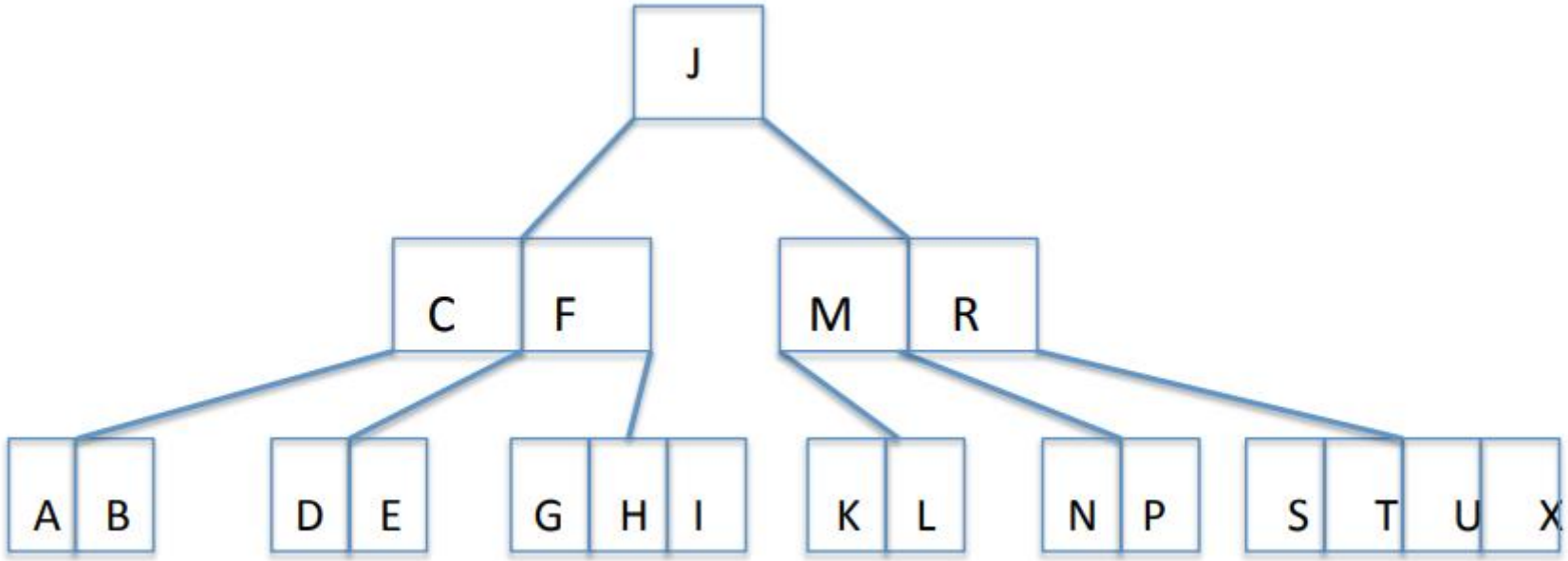
5-way Trees

A B-Tree of order $\mathbf{5}$ is a $\mathbf{5-way}$ tree such that

- All leaf nodes are at the same level
  - All non-leaf nodes (except the root) have at most $\mathbf{5}$ children and at least $\frac{\mathbf{5}}{\mathbf{2}} = \mathbf{2}$ children.
- The number of keys (values)
  - Non-Leaf Nodes: at least one less than the number of children, at most: $\mathbf{5-1=4}$
  - Leaf-Nodes: at least $\frac{\mathbf{5}}{\mathbf{2}} = \mathbf{2}$
- Each internal node stores an ordered set of keys (sorted)
- The root may have as few as 2 children unless the tree is the root alone

5-way Tree: Example

- Validate properties

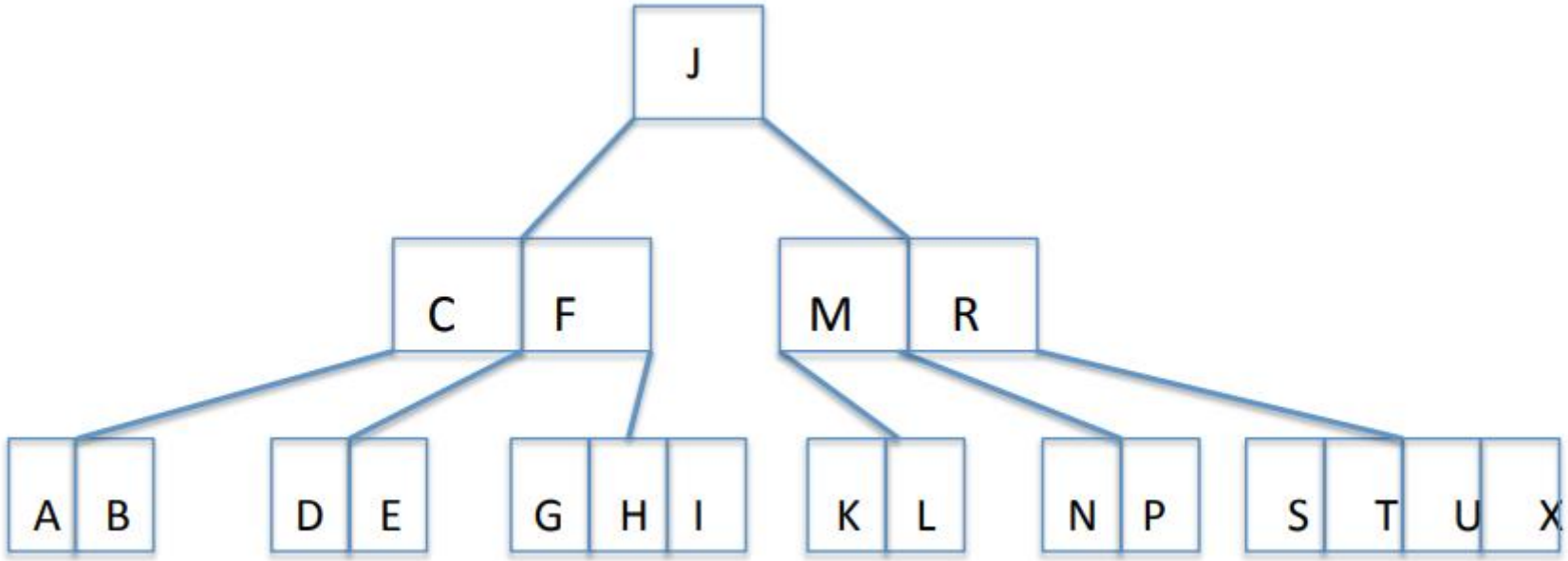


Searching in B-Trees

1. Start at the root node.
2. Compare the search key with the keys in the current node.
3. If the search key is found, return the associated data.
4. If the search key is less than the smallest key in the current node, follow the leftmost child node.
5. If the search key is greater than the largest key in the current node, follow the rightmost child node.
6. If the search key is between two keys in the current node, follow the child node immediately to the right of the smallest key that is greater than or equal to the search key.
7. Repeat steps 2-6 until the search key is found or a leaf node is reached.
8. If a leaf node is reached and the search key is not found, the search is unsuccessful.

Search: Example

- Find H



Inserting in B-Trees

1. Start at the root node.
2. Find the leaf node where the key should be inserted.
3. If the leaf node has room for the key, insert it in the correct position and update the tree.
4. If the leaf node is full, split it into two nodes.
5. Move the middle key up to the parent node.
6. If the parent node is also full, split it into two nodes and move the middle key up to its parent node.
7. Repeat steps 5-6 until a node is found that has room for the new key or the root node is split.
8. If the root node is split, create a new root node with the middle key as the only key and the two split nodes as its children.
9. Update the tree structure and ensure that the B-tree properties are maintained.

Insert: 5-way Tree

- Insert: ~~A~~, G, F, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P

A			

Insert: 5-way Tree

- Insert: ~~A, G~~, F, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P

A	**G**		

Insert: 5-way Tree

- Insert: ~~A, G, F~~, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P

A	**F**	**G**	

Insert: 5-way Tree

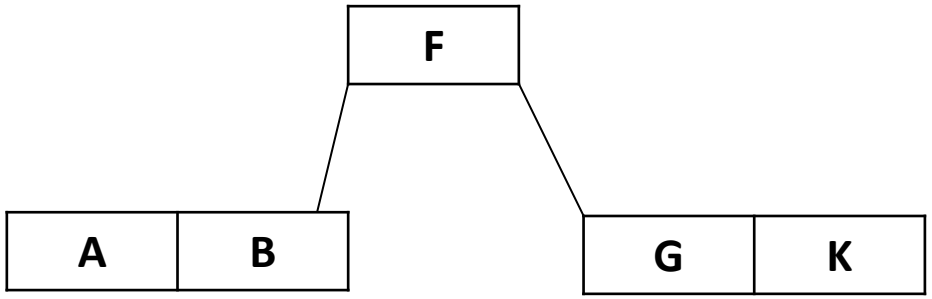
- Insert: ~~A, G, F, B,~~ K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P

A	B	F	G

Insert: 5-way Tree

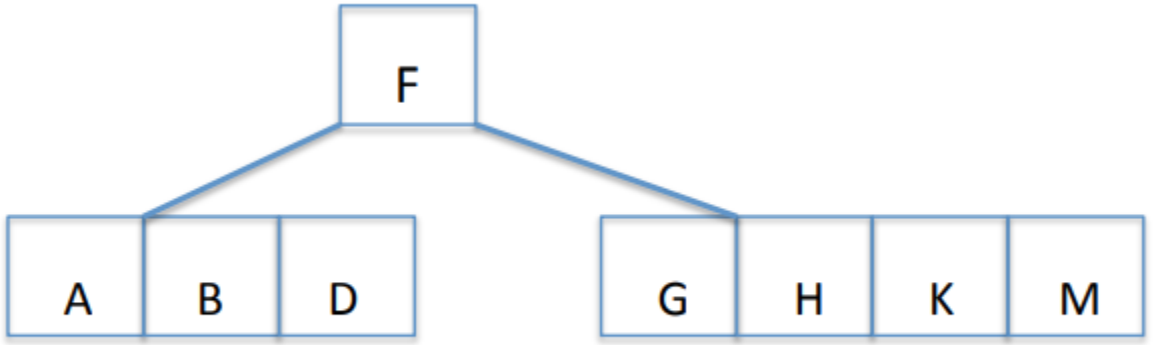
- Insert: ~~A, G, F, B, K,~~ D, H, M, J, E, S, I, R, X, C, L, N, T, U, P

- Split



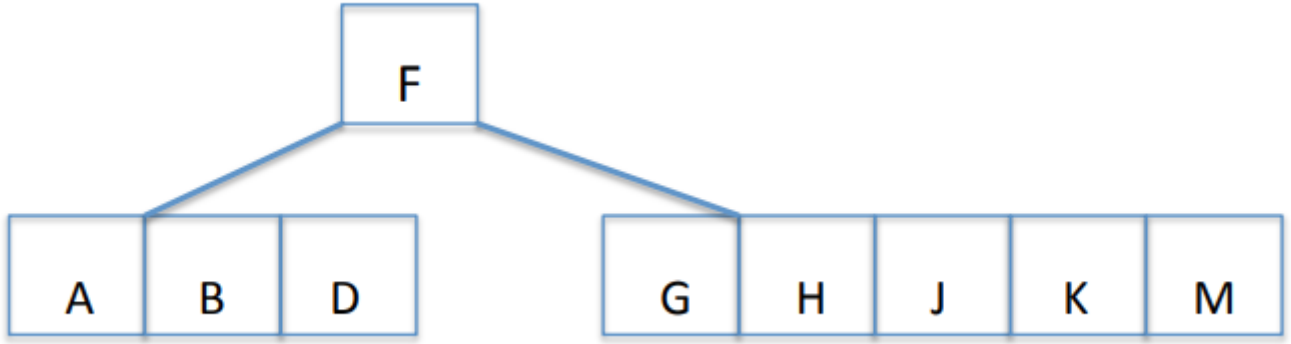
Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M~~, J, E, S, I, R, X, C, L, N, T, U, P

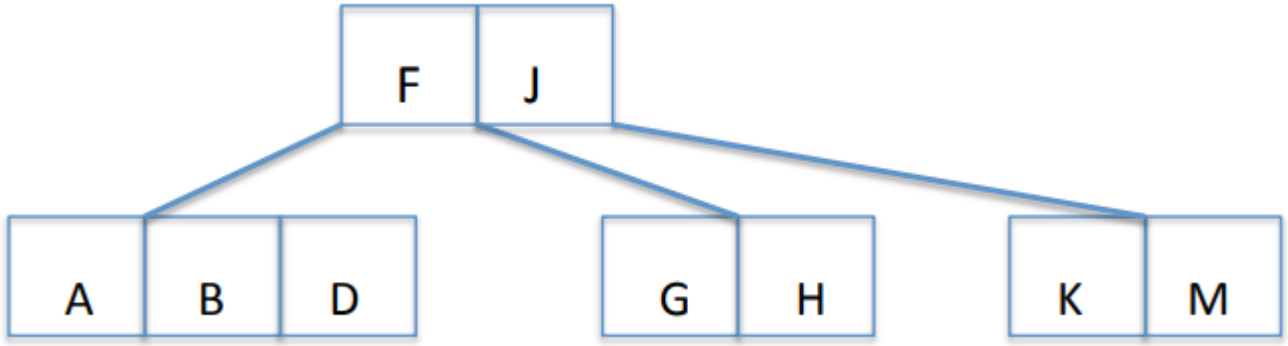


Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J~~, E, S, I, R, X, C, L, N, T, U, P

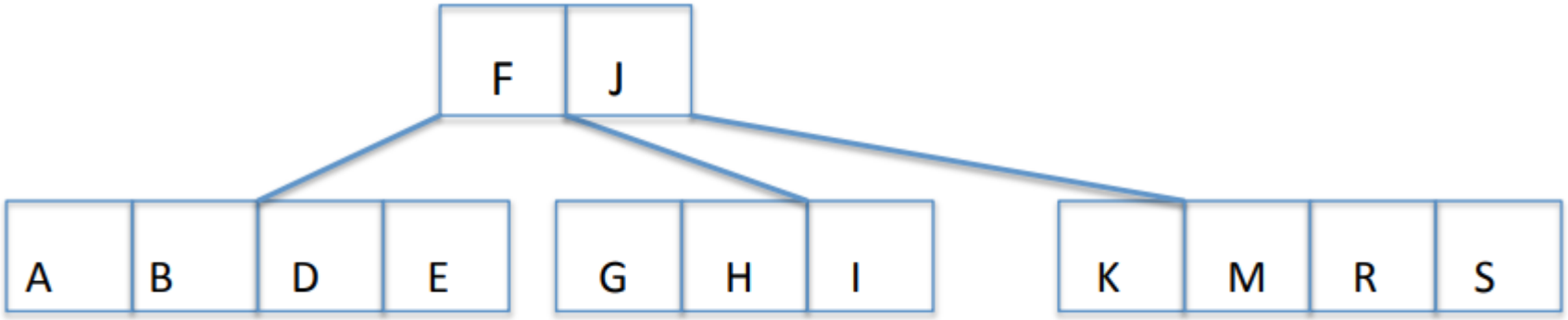


- Split



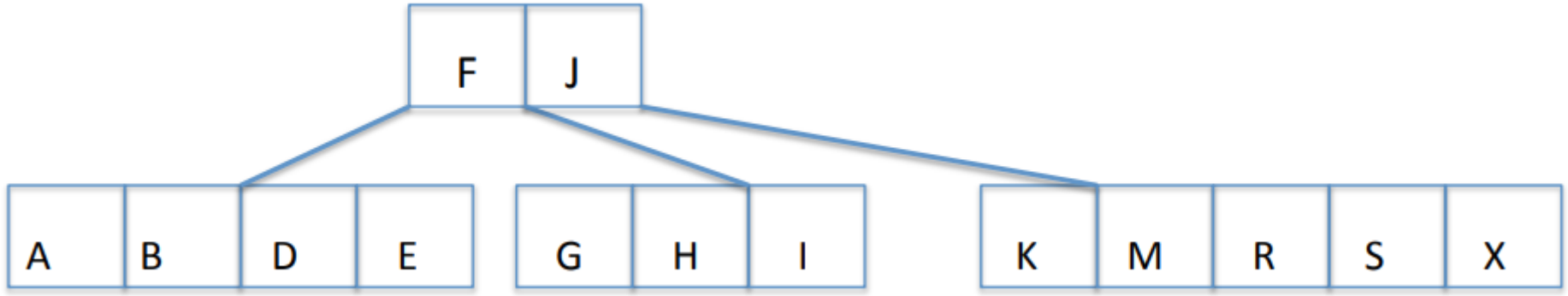
Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J, E, S, I, R,~~ X, C, L, N, T, U, P

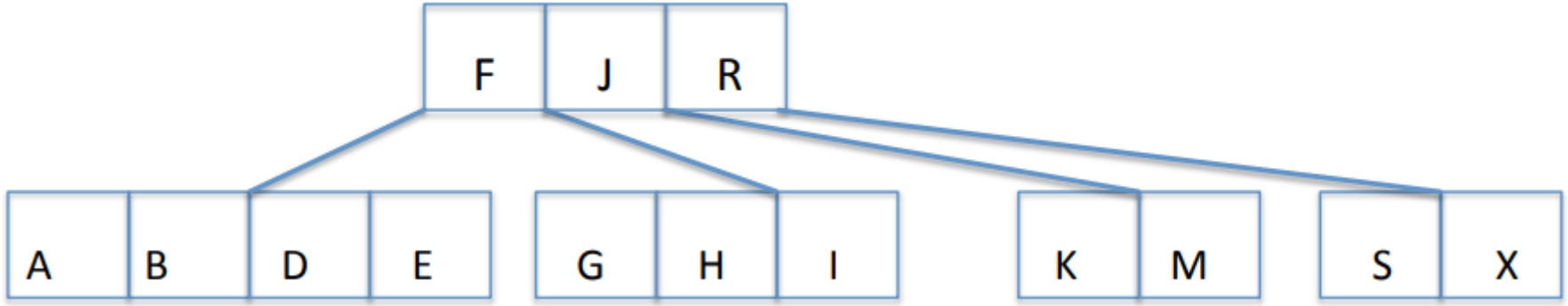


Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J, E, S, I, R, X,~~ C, L, N, T, U, P

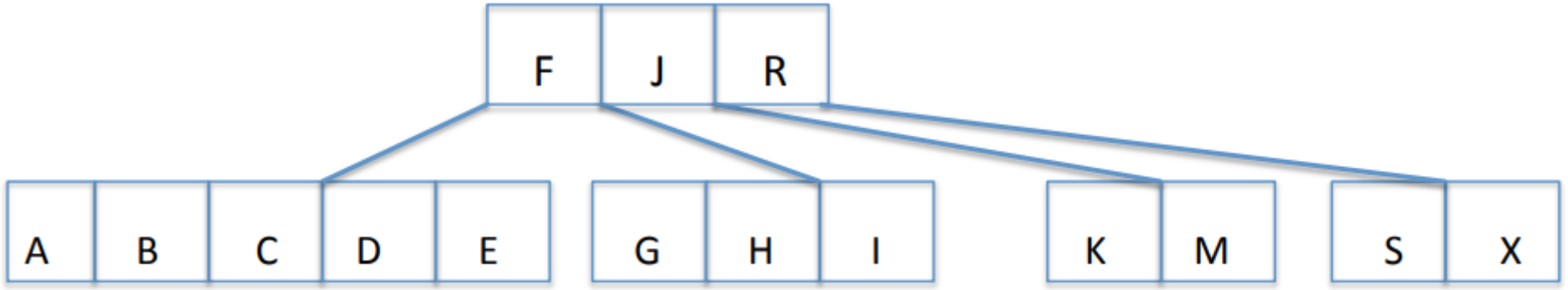


Split

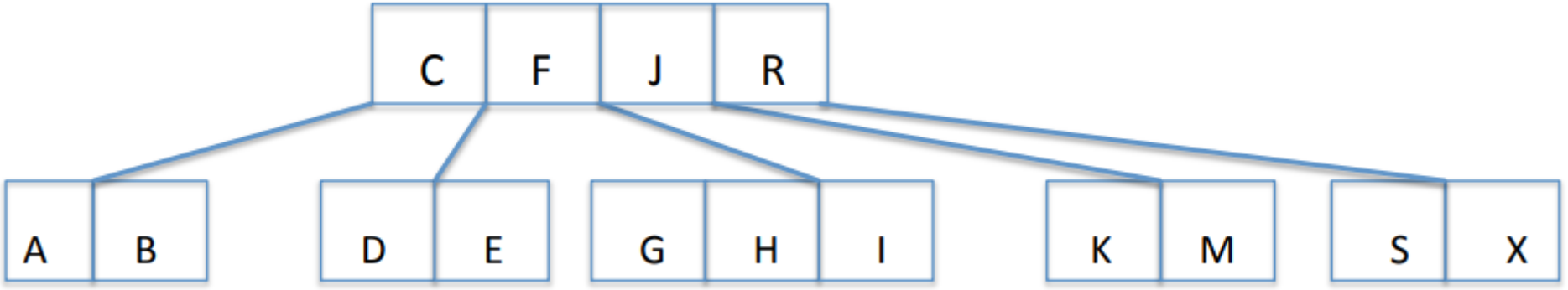


Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J, E, S, I, R, X, C~~, L, N, T, U, P

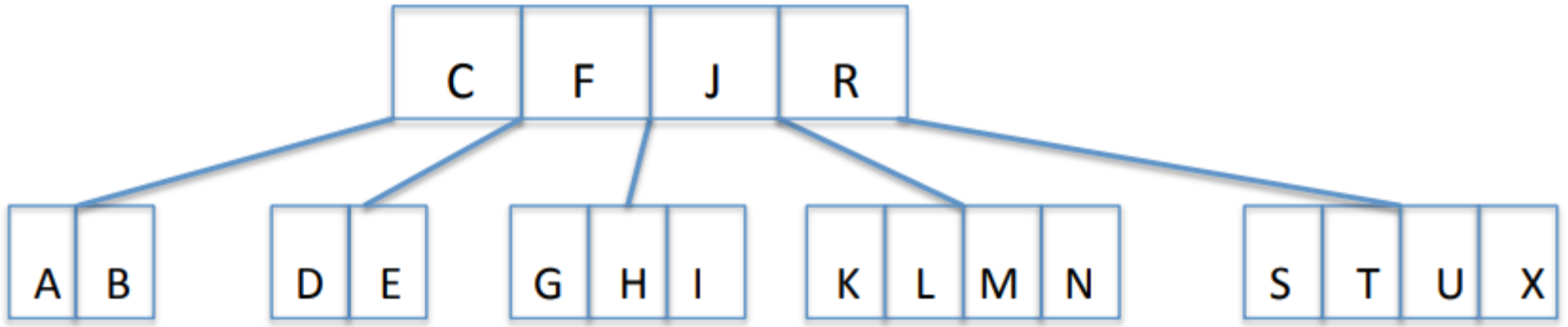


- Split



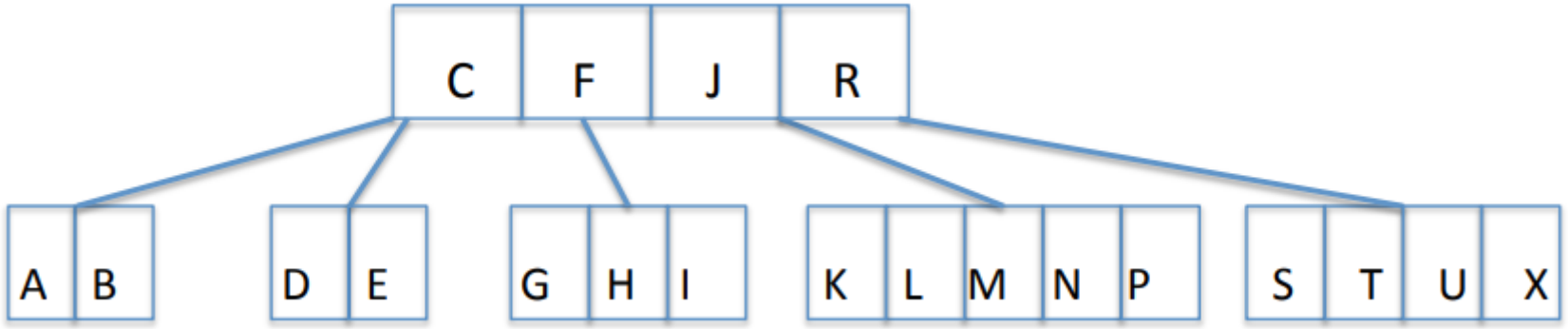
Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U,~~ P

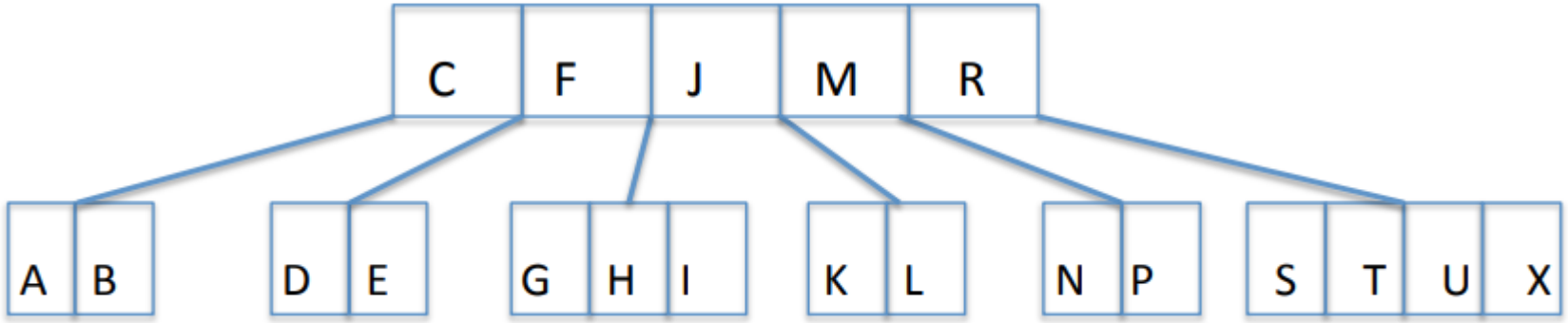


Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P~~

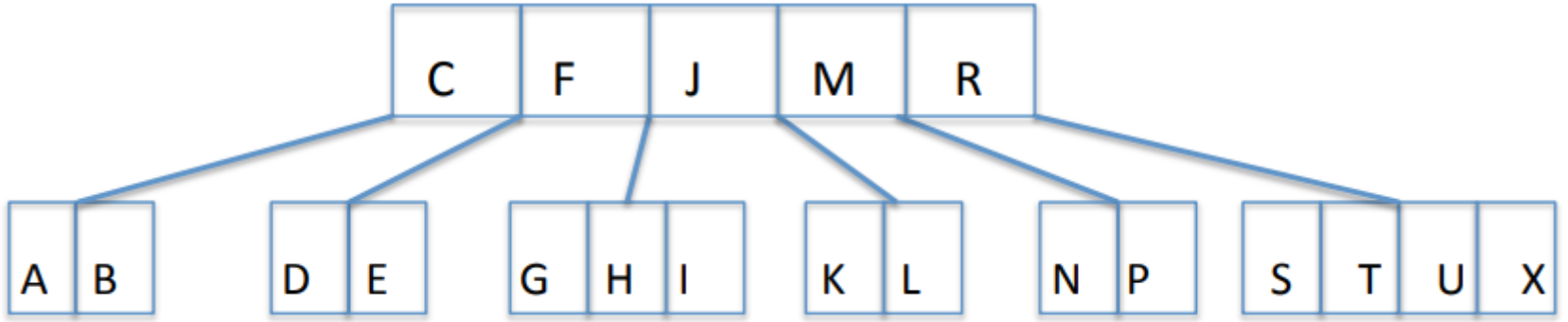


Split->

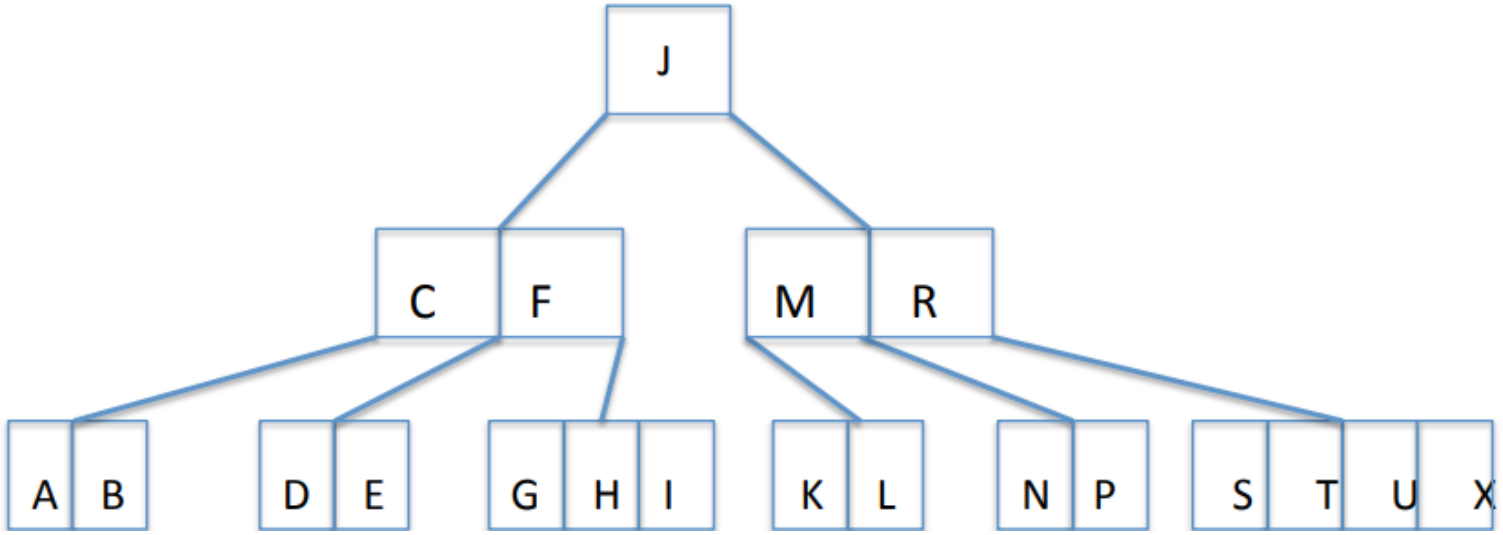


Insert: 5-way Tree

- Insert: ~~A, G, F, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P~~



Split->



2-4 / 2-3-4 Trees

- A 4-way tree that maintains two properties:

1. Size Property: Every internal node has at most four children
2. Depth Property: All external node have the same depth

- Maximum number of children a node can have:
- Minimum number of children a node can have (other than root):
- Possible number of children a node can have (other than root):

Example: 2-4 Tree

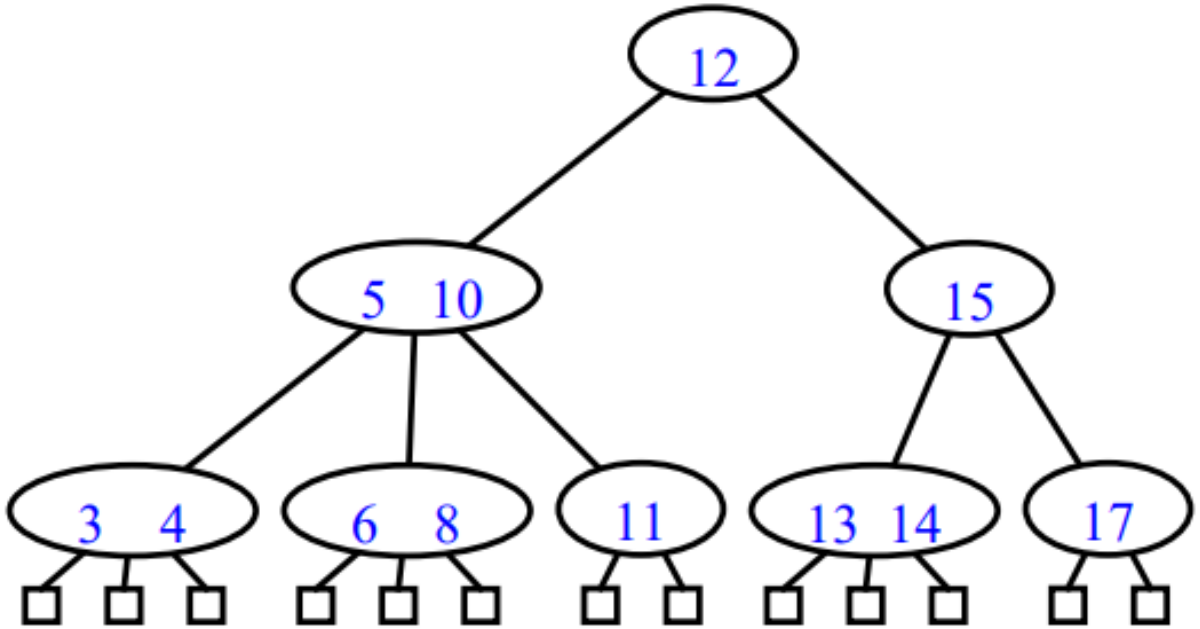


Image: CS 251 Data Structures, Purdue University

Insert: 4-way Tree (2-4 Tree / 2-3-4 Tree)

- Insert: A, G, F, B, K, D, H, M, J, E, S, I, R, X, C, L, N, T, U, P

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

Height of a 2-4 Trees (or a B-tree)

- The number of nodes at level h is 2^h

- Lemma 9.1 (ODS) For a 2-4 tree of height h, it has at least 2^h leaves. So, total number of nodes must be:

$$n \geq 2^h \Rightarrow h = O(\log n)$$

- Consider a 2-4 tree of height h with all nodes are full (k-1 = 3 keys).

$$n = 3 \cdot 4^0 + 3 \cdot 4^1 + 3 \cdot 4^2 + \ \ldots + 4^h$$

$$n = 3\left(4^0 + 4^1 + 4^2 + \ \ldots 4^h\right)$$

$$n = 3\left(\frac{4^{h+1} - 1}{4 - 1}\right) \Rightarrow n = 4^{h+1} - 1$$

$$n + 1 = 4^{h+1} \Rightarrow h + 1 = \log_4(n+1) \Rightarrow h \geq \log_4(n+1)$$

$$h = \Omega(\log_4(n+1))$$