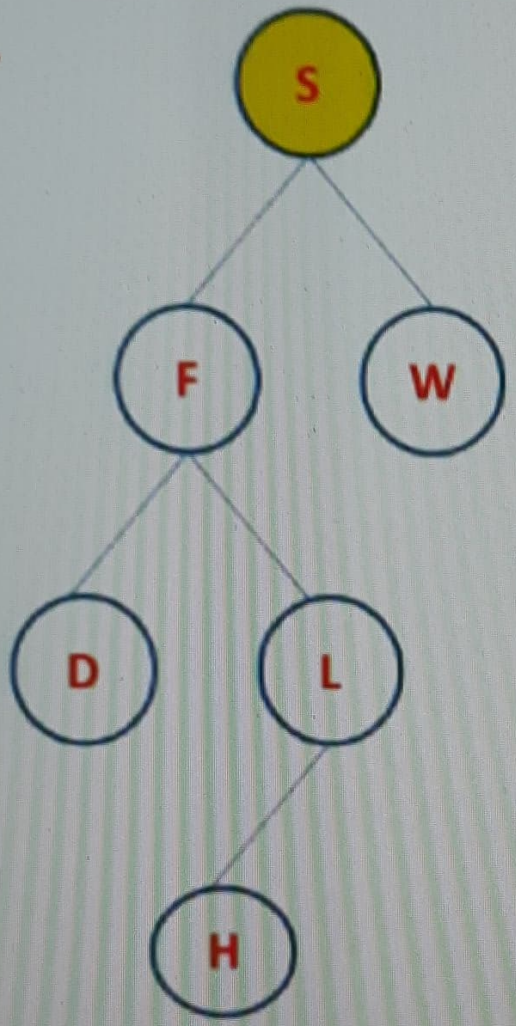
Binary Search Trees

- All keys in the tree are distinct
- For every non-empty subtree
  - Left child has a value less than its parent
  - Right child has a value greater than its parent
- Both Left and right subtrees are binary subtrees

Search

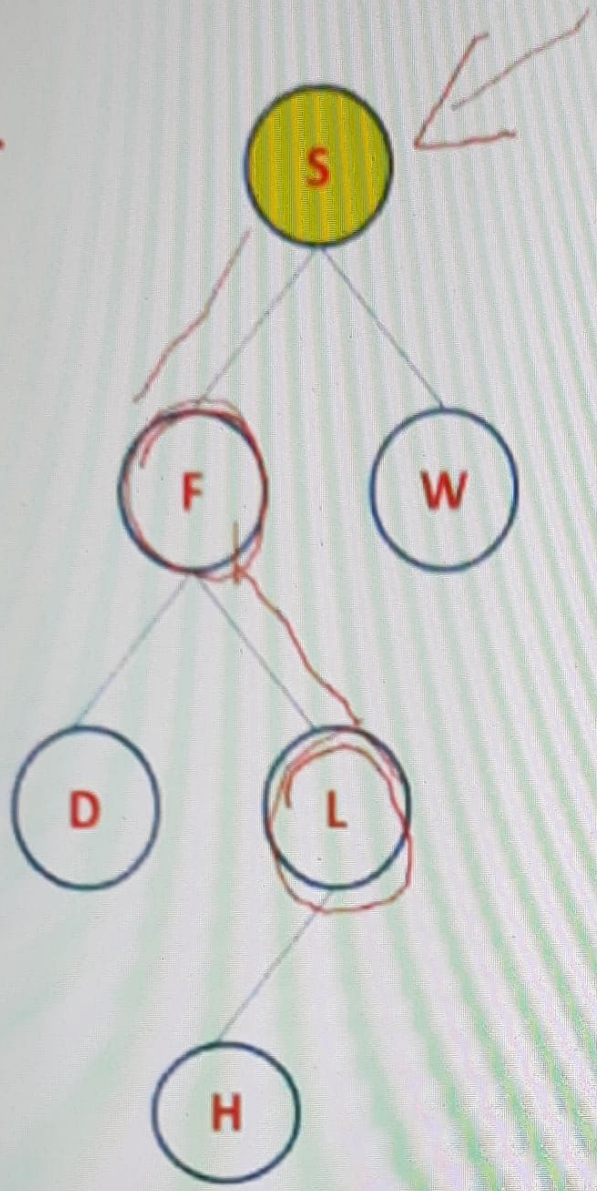
BST-Search

- Search L



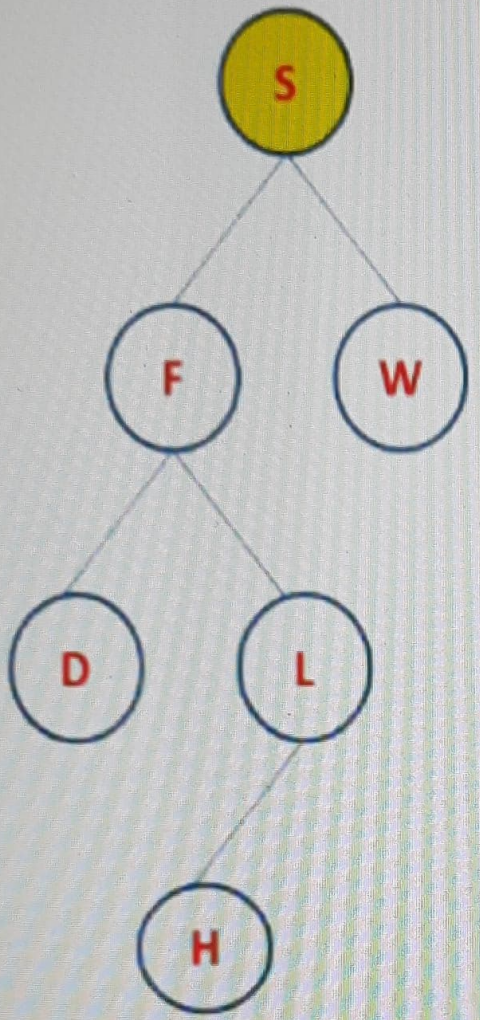
BST-Search

- Search L



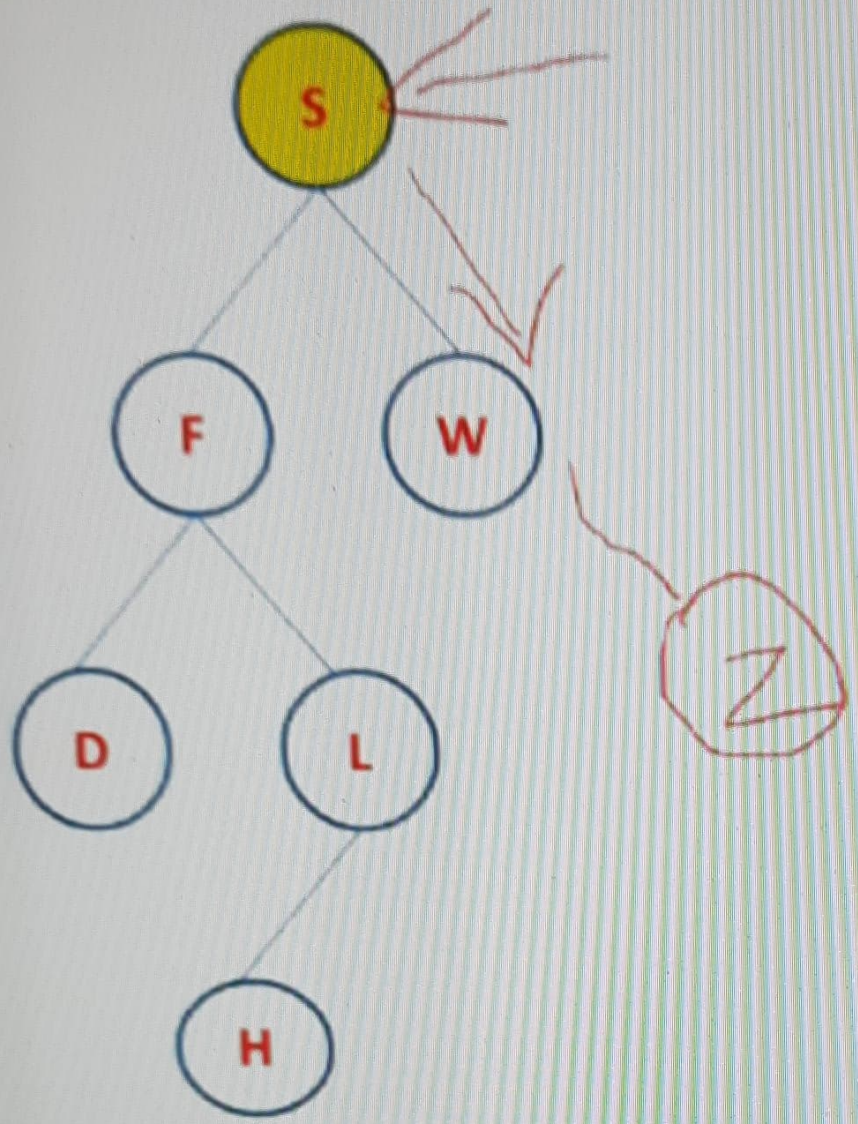
BST- Insertion

- Insert Z



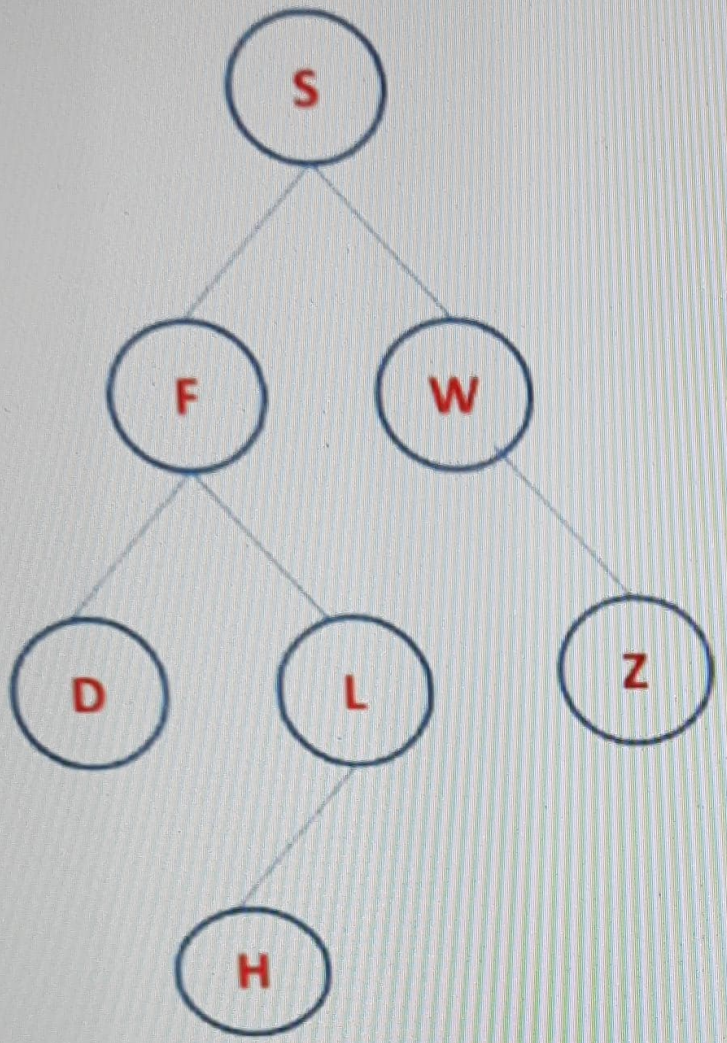
BST- Insertion

- Insert Z



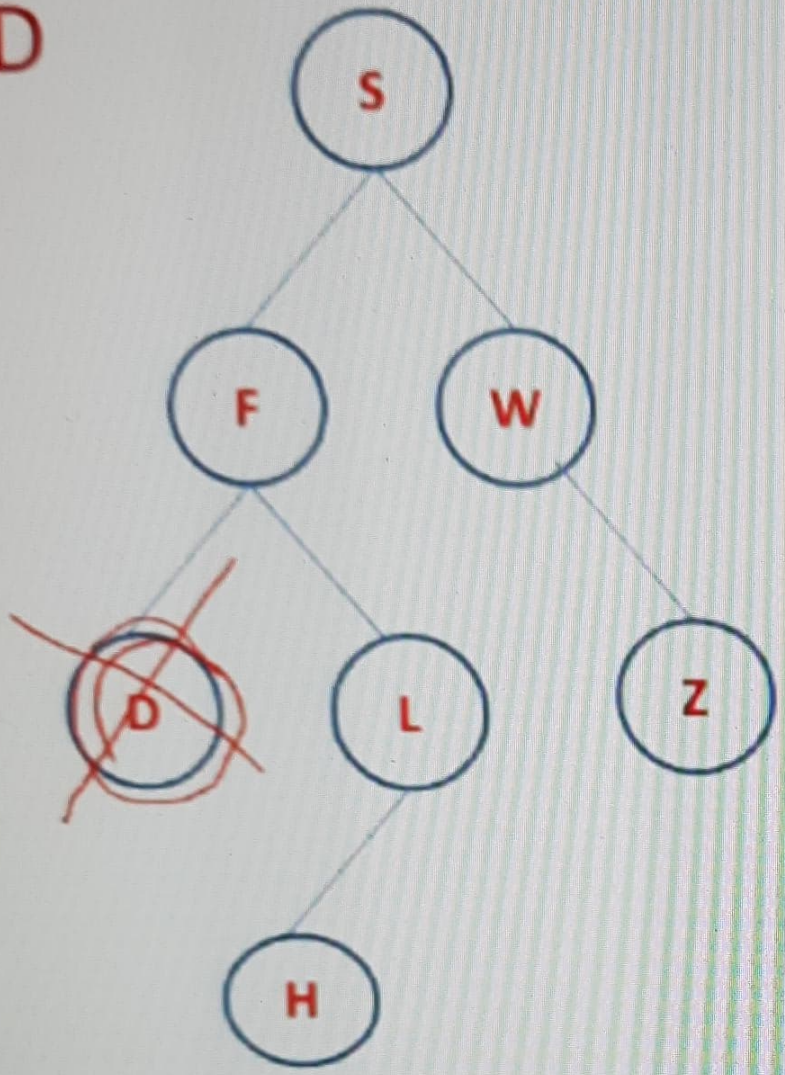
BST- Deletion

- Delete D



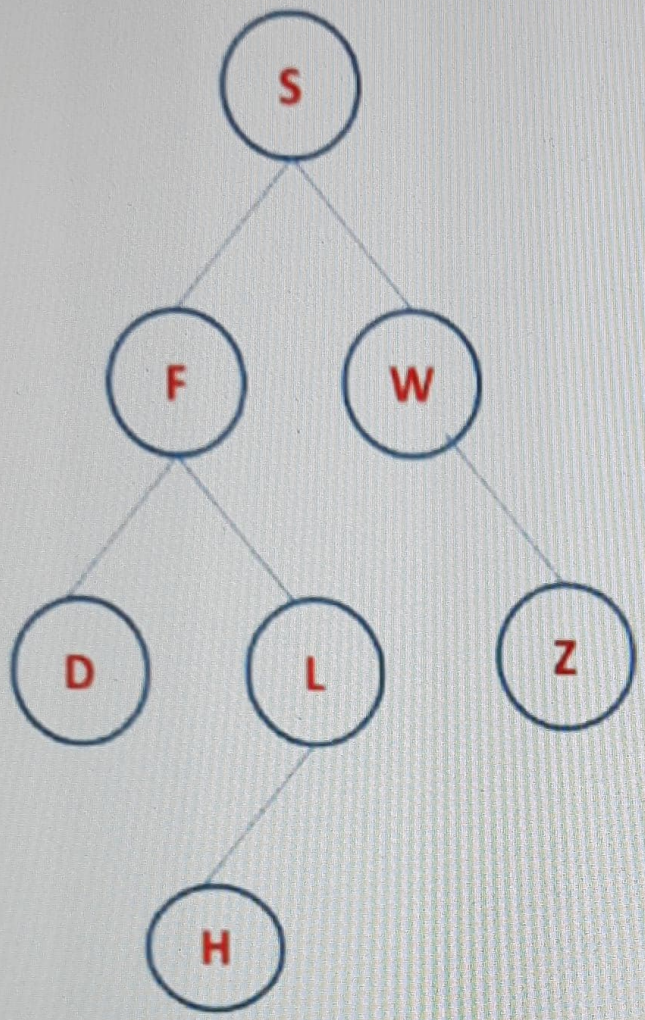
BST- Deletion

- Delete D



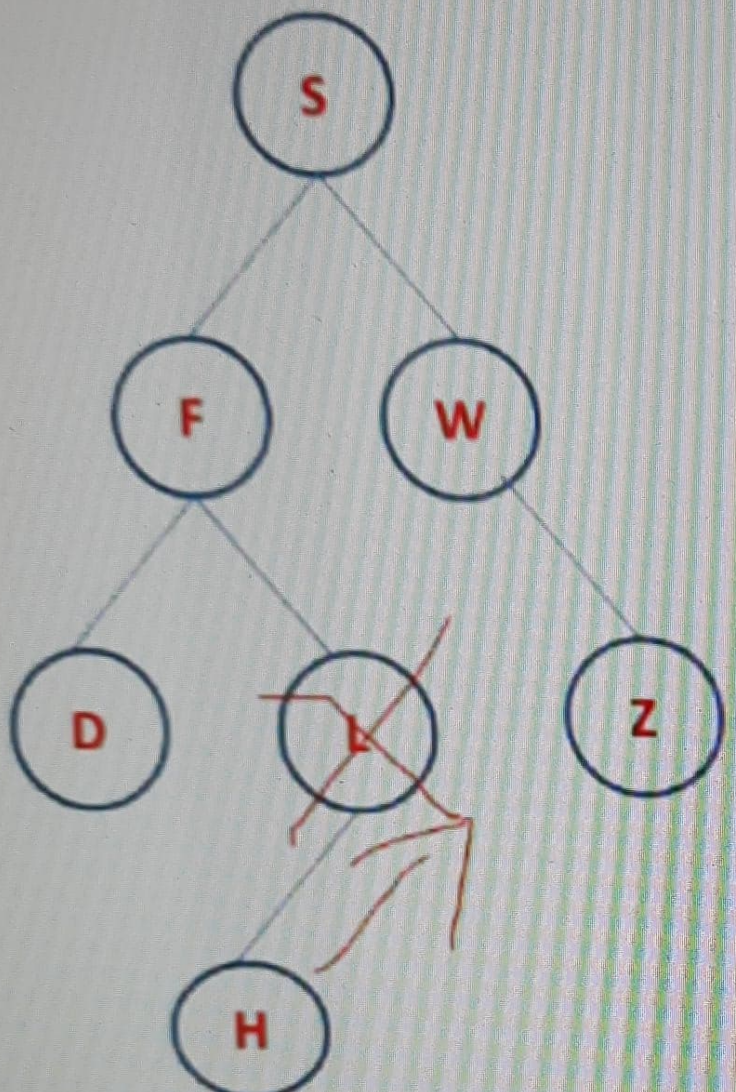
BST- Deletion

- Delete L



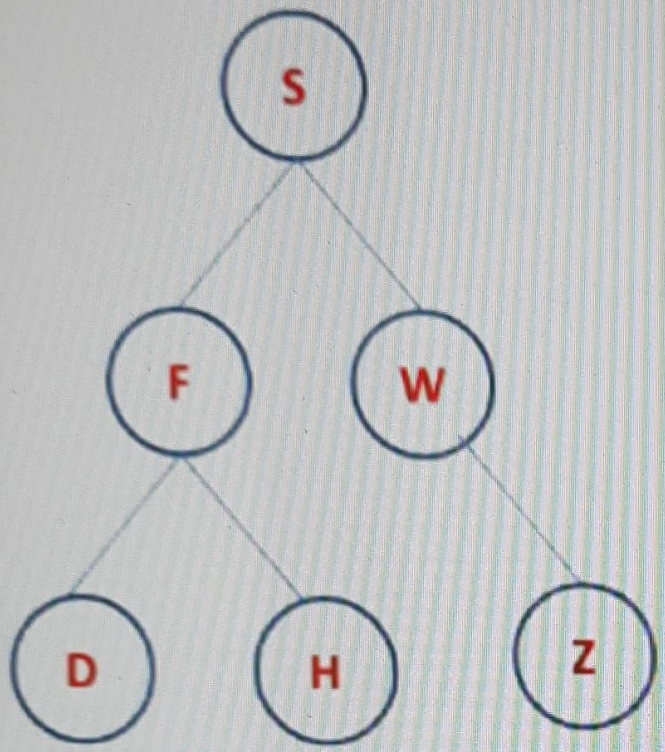
BST- Deletion

- Delete L



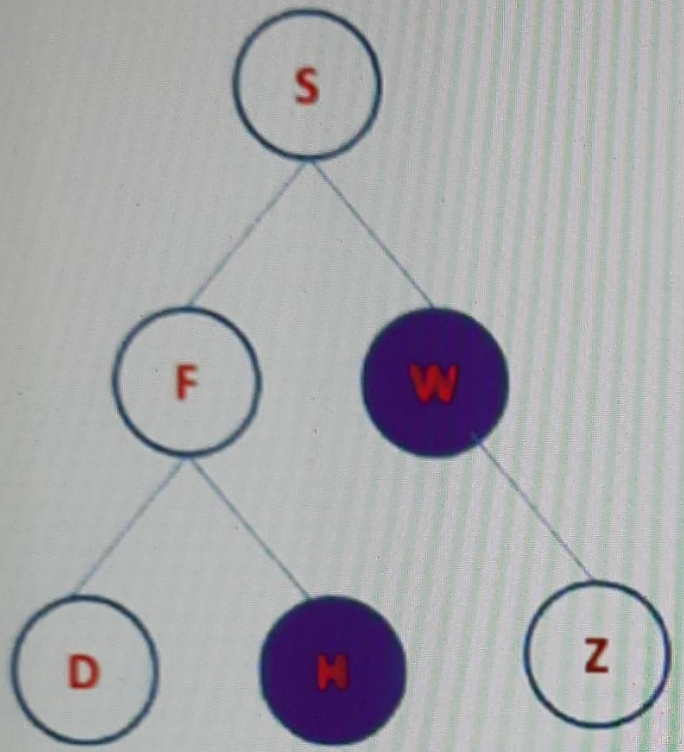
BST- Deletion

- Delete S



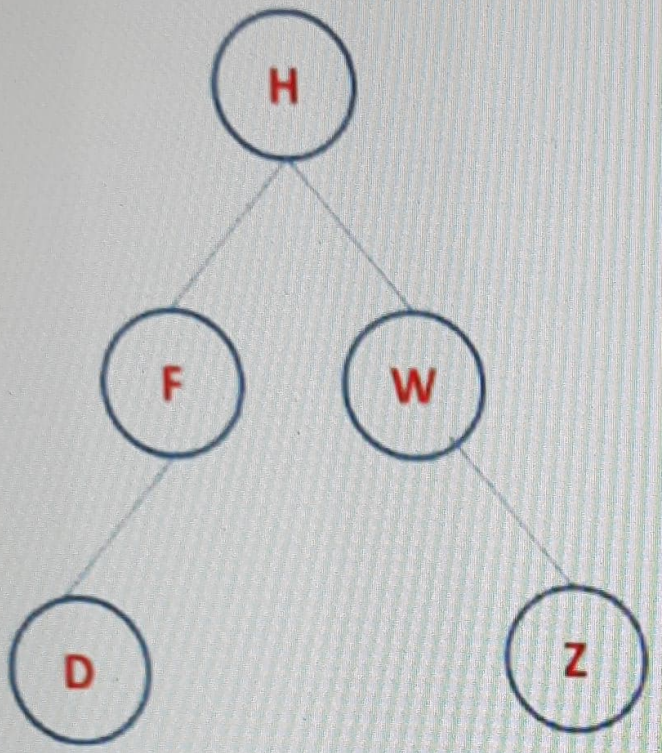
BST- Deletion

- Delete S



BST- Deletion

- Delete S



BST

- Best case height: $\Theta(\ln(n))$
- Worst case height: $\mathbf{O}(n)$

Requirement:

- Define and maintain a *balance* to ensure $\Theta(\ln(n))$ height