Balanced Search Trees & LLRBs

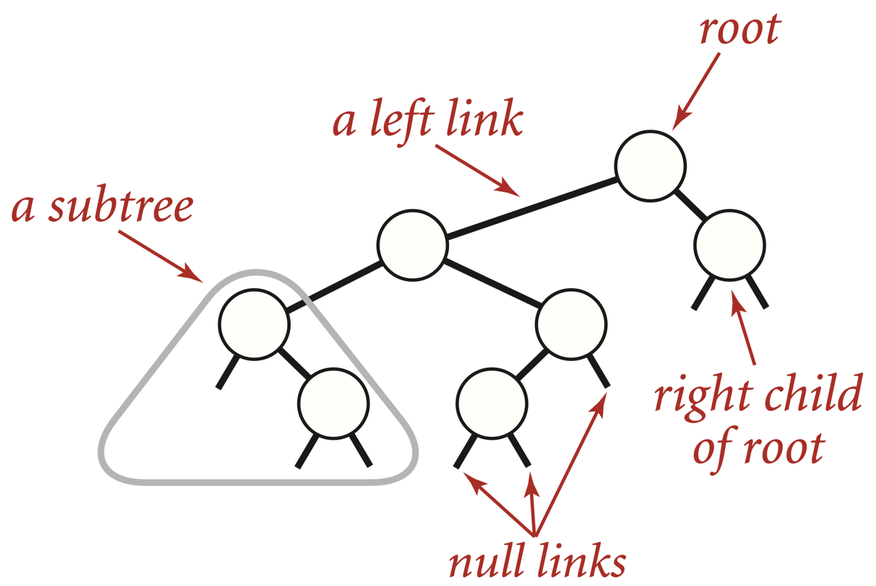
Discussion 08

CS 61B // Presented by: Jacob Wu

Agenda & Key Learning Outcomes

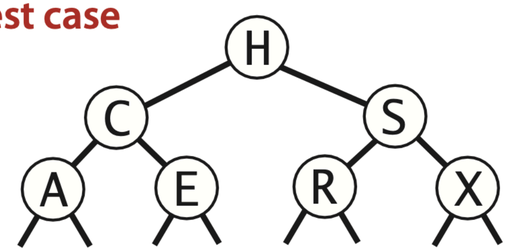
- Develop an appreciation of the foundational ideas of a Balanced Search Tree
- Discuss implementation and motivate the discovery of Red-Black BST
- Compare and contrast different implementations of BST, Balanced Search Tree, and Red-Black BST

Reminder: BST

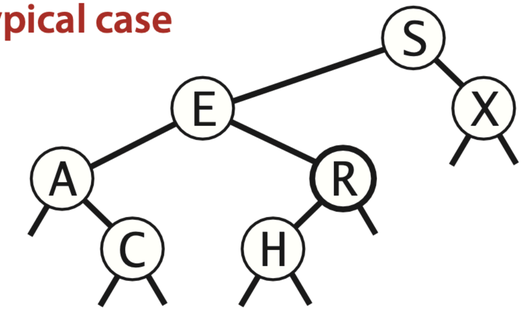


Anatomy of a binary tree

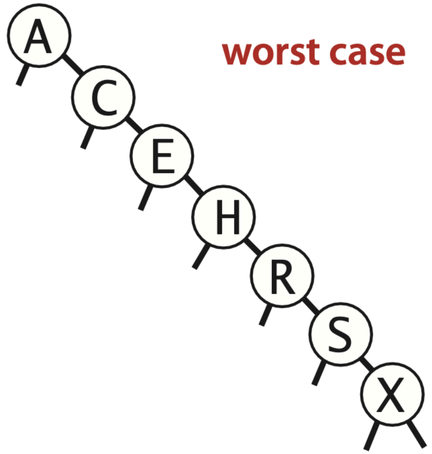
best case



typical case

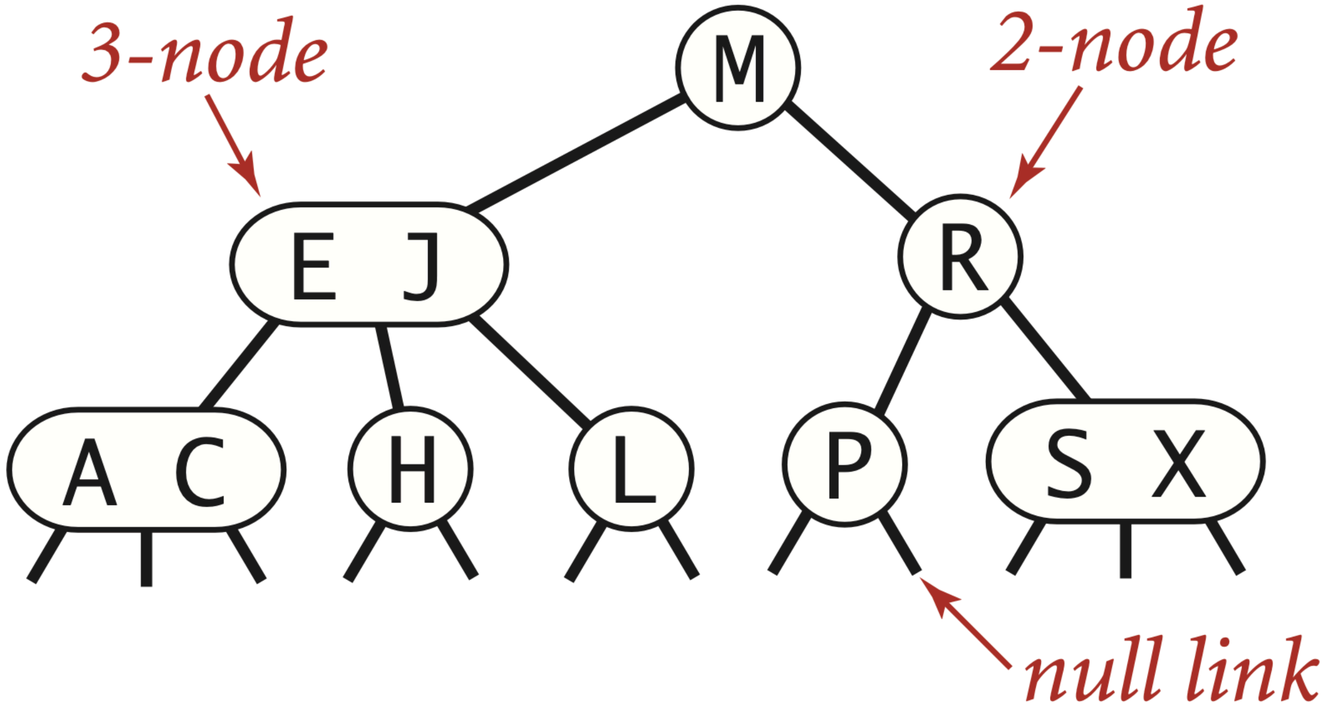


worst case



BST possibilities

B-Trees
(a.k.a. 2-3 Trees)



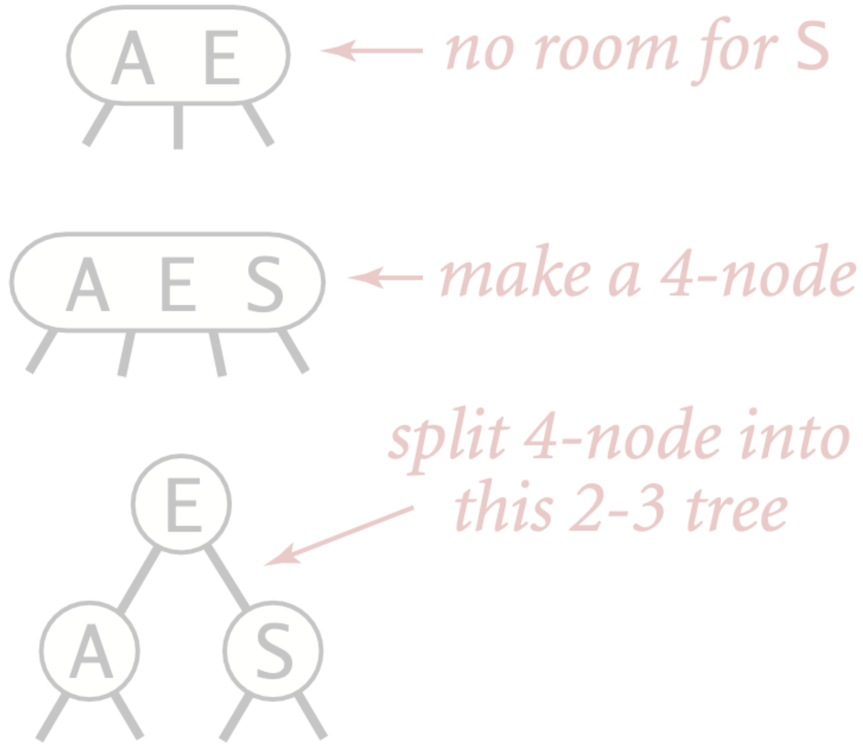
- *2-node*: one item, two children
- *3-node*: two items, three children
- Guaranteed $\theta(\log N)$ runtime

Insert Into a *2-node*

inserting K

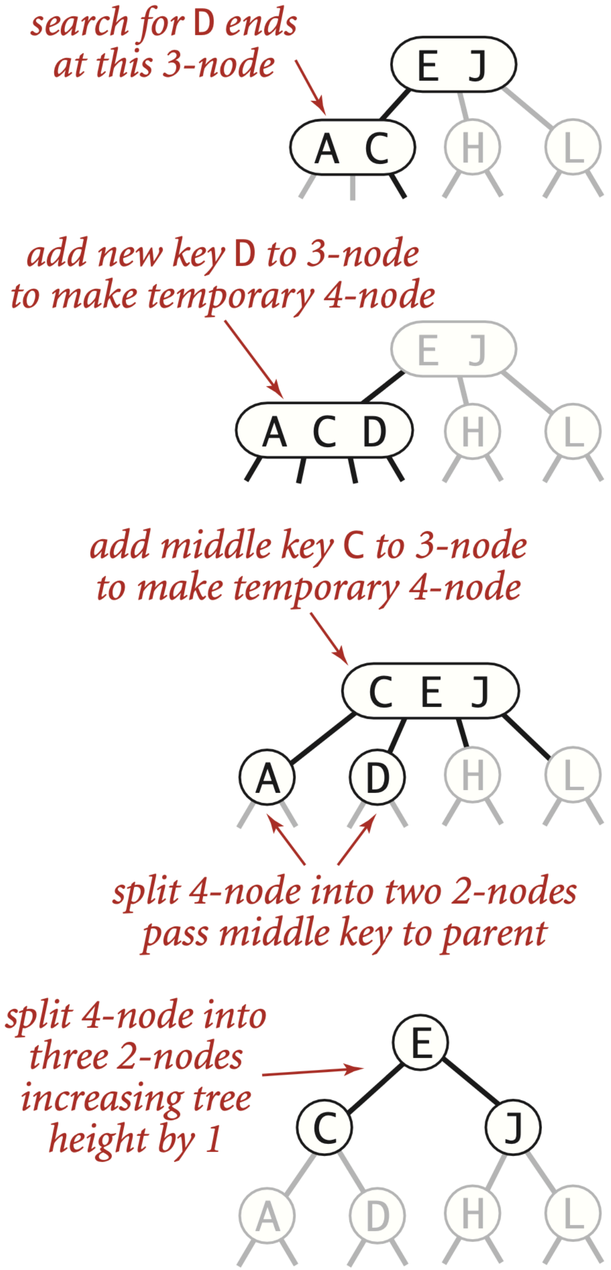
Insert Into a *3-node*

inserting S



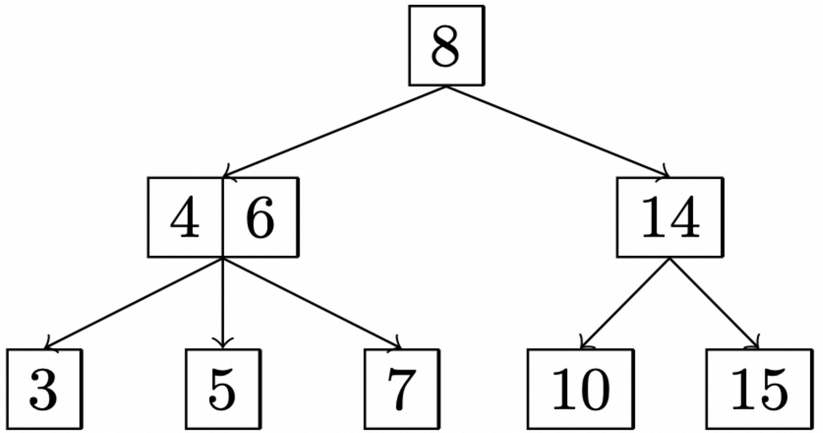
Insert into a single 3-node

inserting D


search for D ends at this 3-node
E J
A C
H
L
add new key D to 3-node to make temporary 4-node
E J
A C D
H
L
add middle key C to 3-node to make temporary 4-node
C E J
A
D
H
L
split 4-node into two 2-nodes pass middle key to parent
split 4-node into three 2-nodes increasing tree height by 1
E
C
J
A
D
H
L

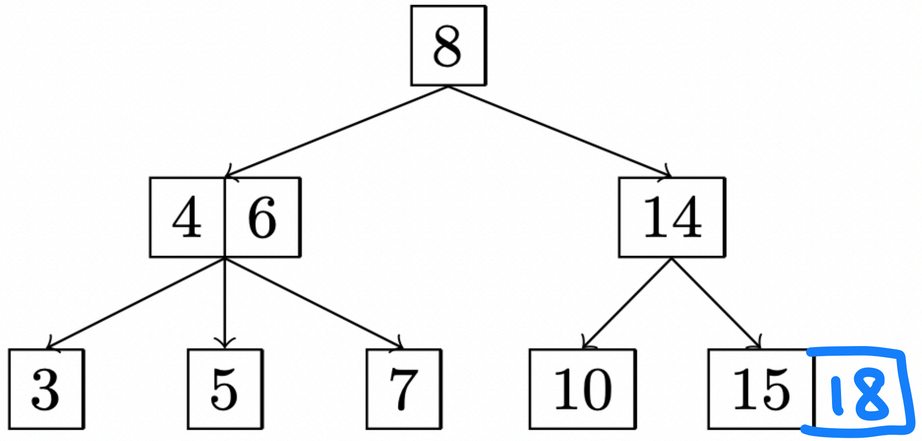
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



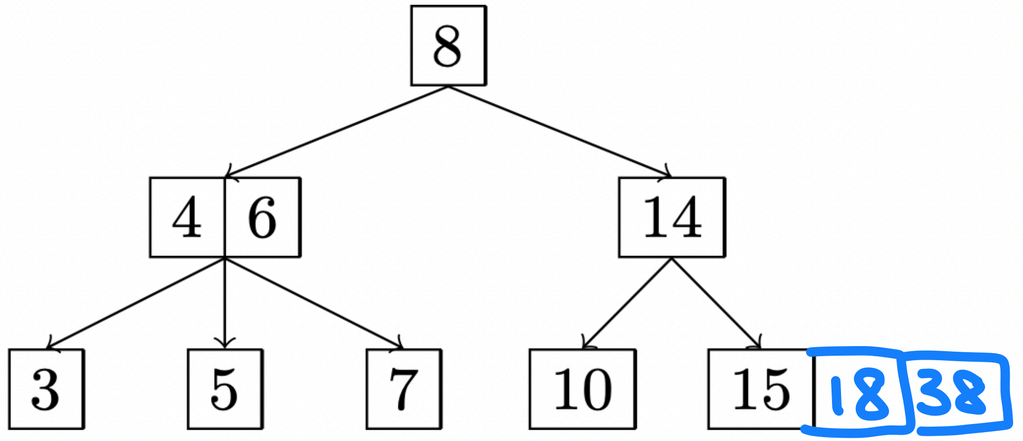
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



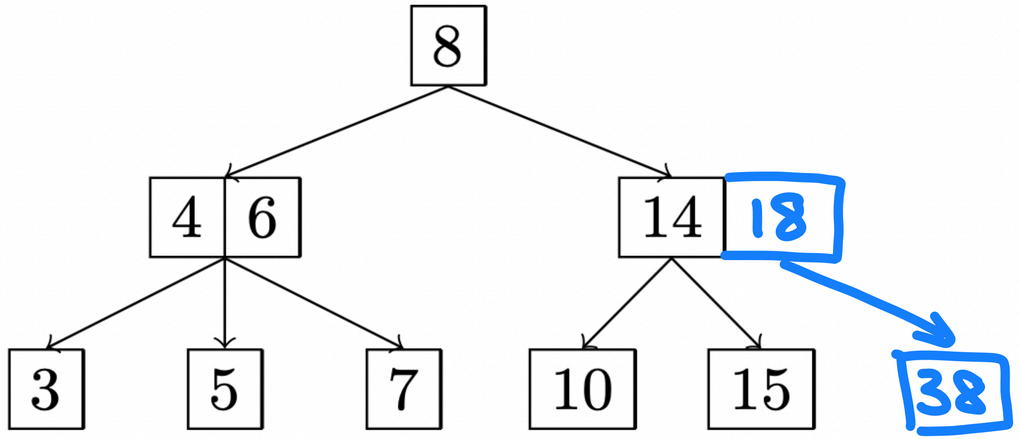
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



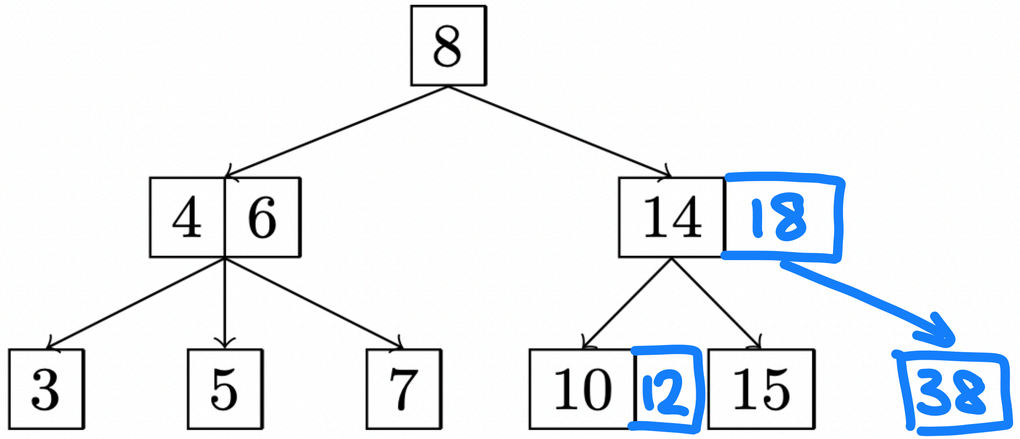
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



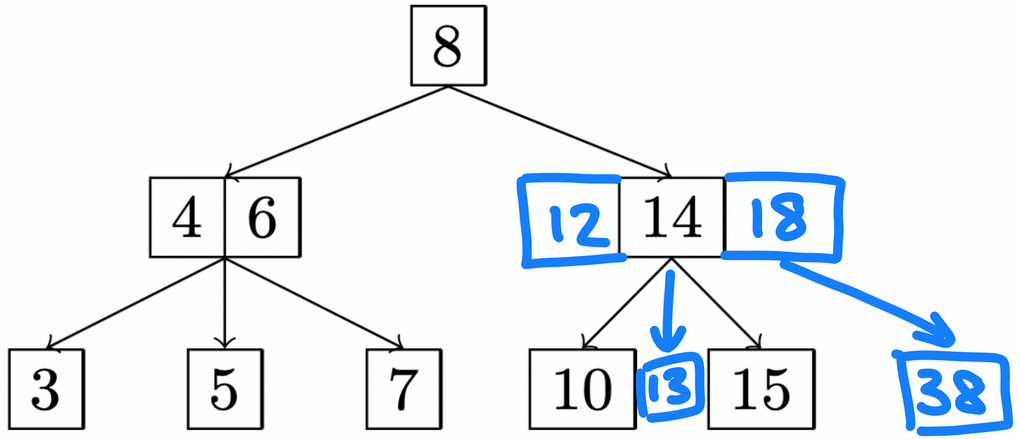
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



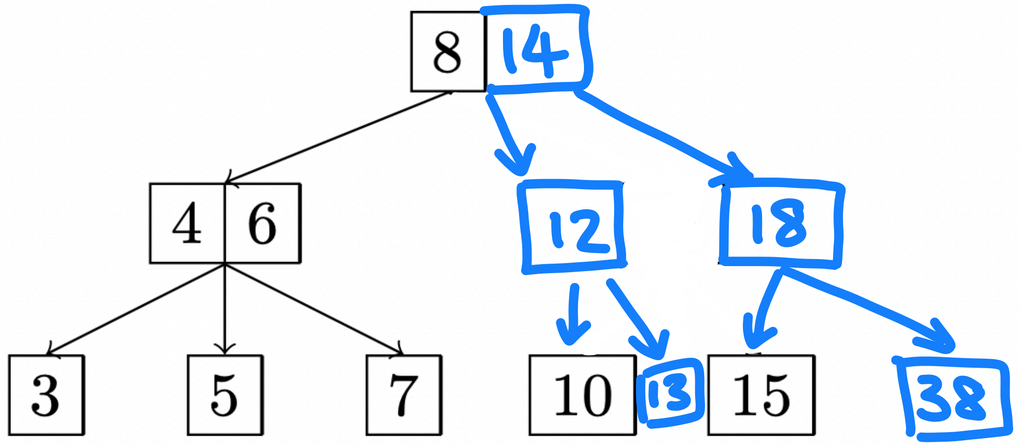
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



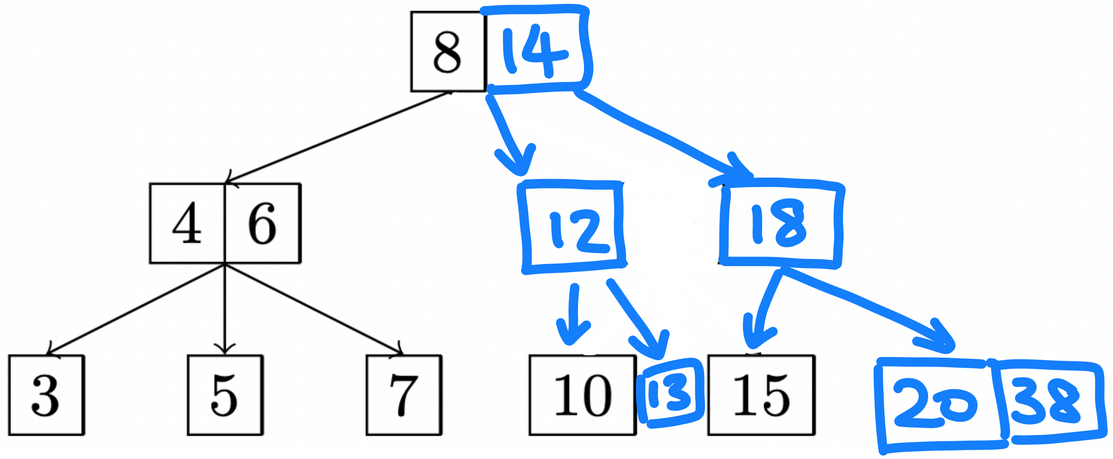
1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.

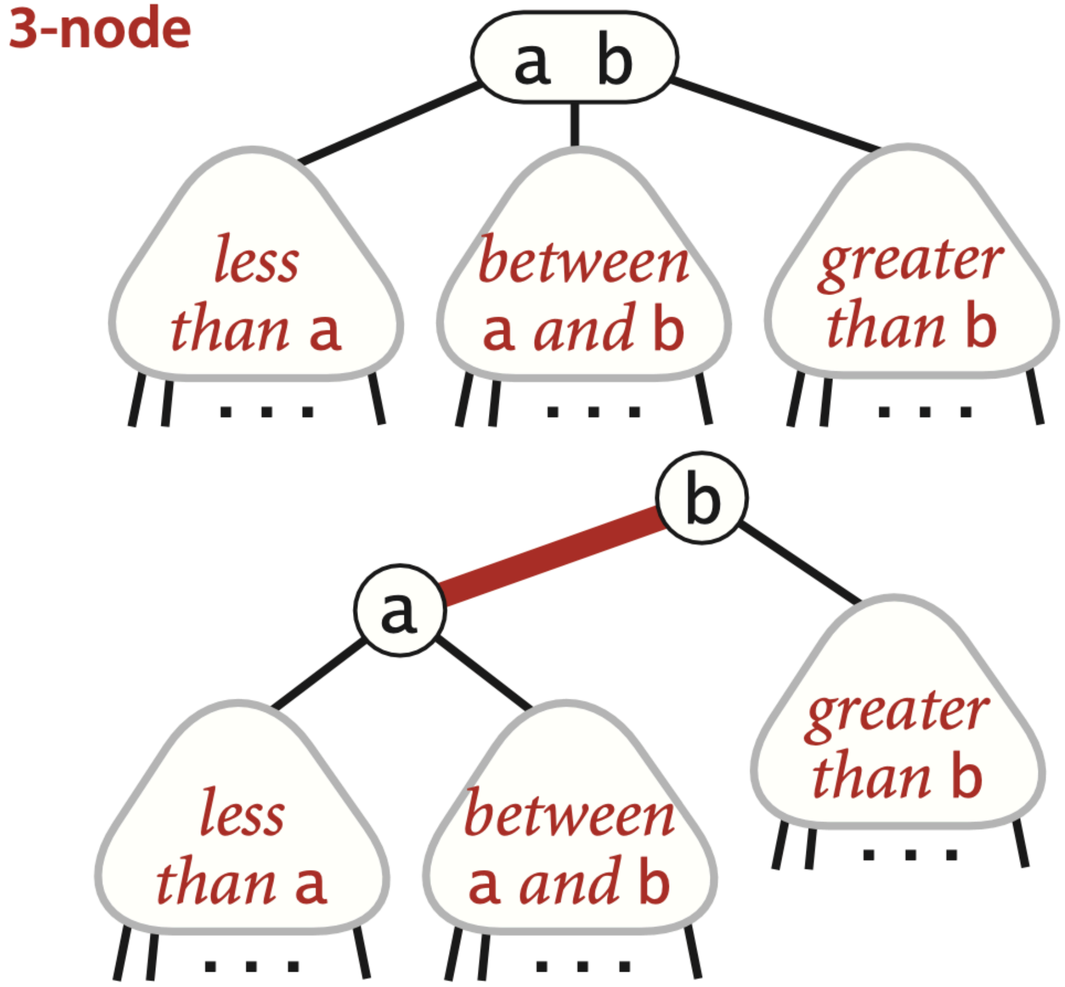


1 2-3 Trees and LLRB's

(a) Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.

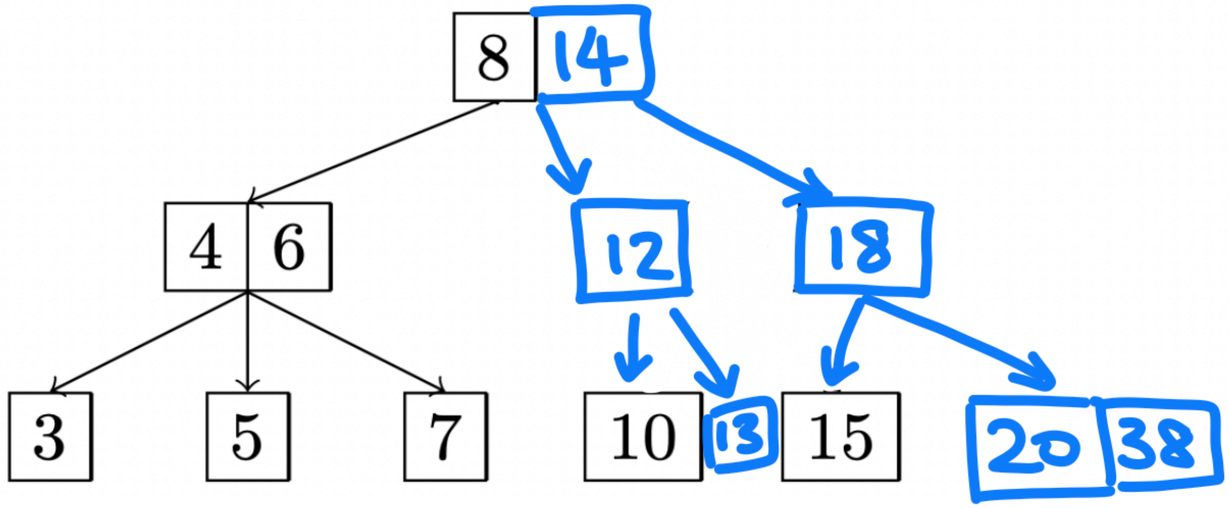


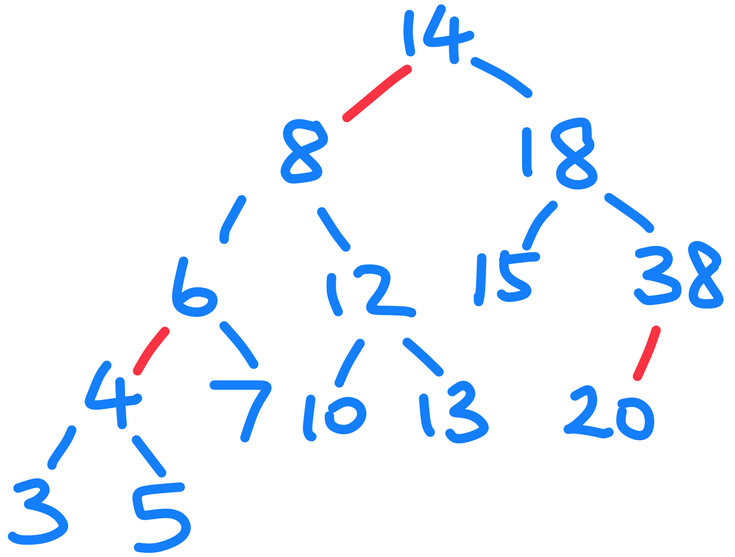
Red-Black BST



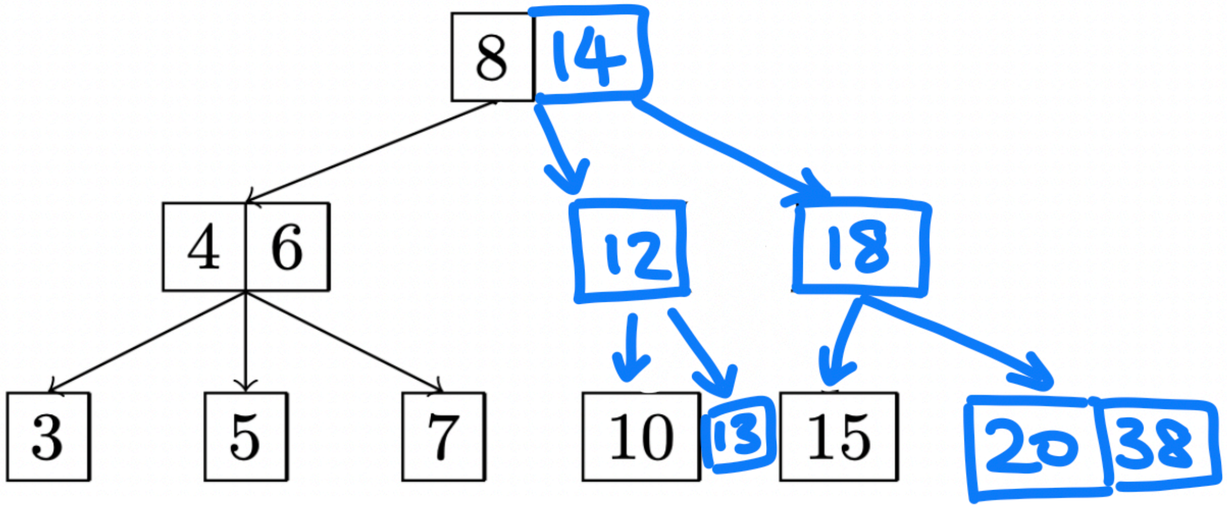
- Red links lean left
- No node has two red links connected to it
- **Perfect black balance**: every path from the root to a null link has the same number of black links

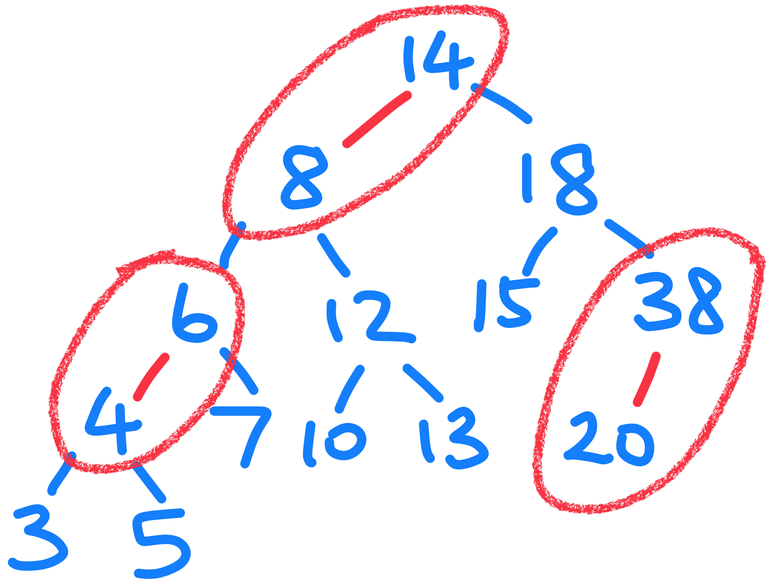
(b) Now, convert the resulting 2-3 tree to a left-leaning red-black tree.



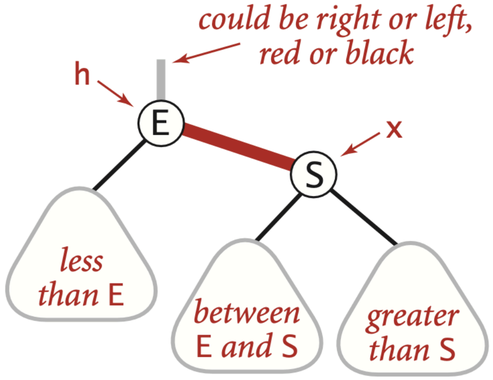


(b) Now, convert the resulting 2-3 tree to a left-leaning red-black tree.

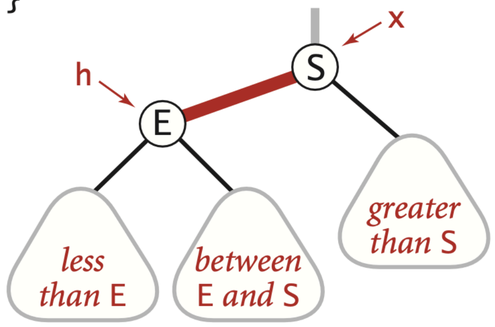




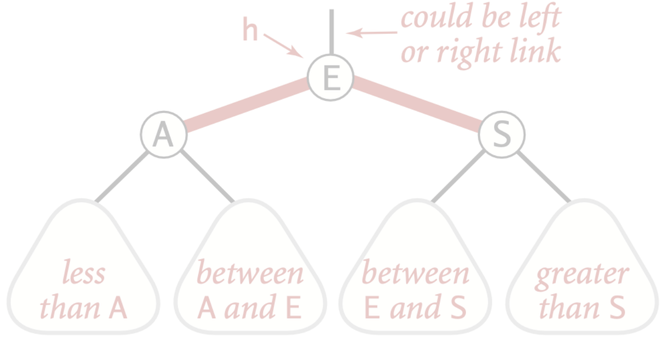
LLRB Operations



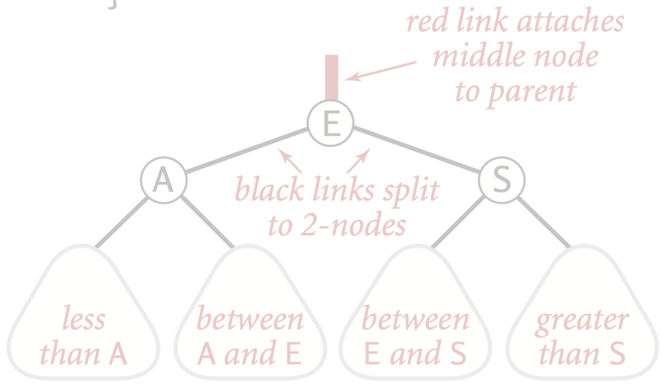

```
Node rotateLeft(Node h)
{
   Node x = h.right;
   h.right = x.left;
   x.left = h;
   x.color = h.color;
   h.color = RED;
   x.N = h.N;
   h.N = 1 + size(h.left)
           + size(h.right);
   return x;
}
```



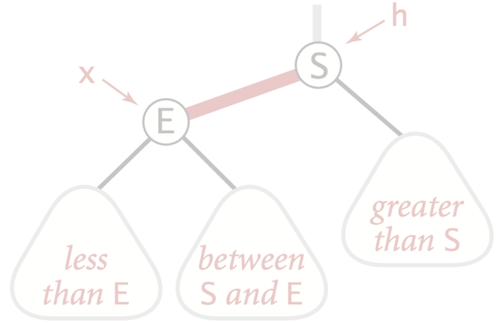
Left rotate (right link of h)



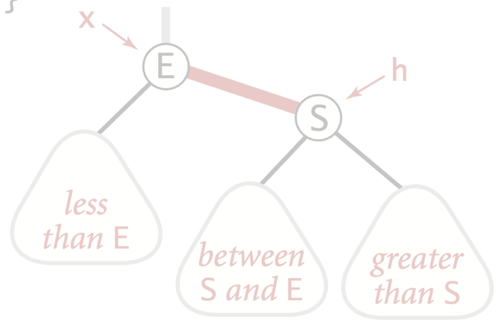

```
void flipColors(Node h)
{
   h.color = RED;
   h.left.color = BLACK;
   h.right.color = BLACK;
}
```



Flipping colors to split a 4-node

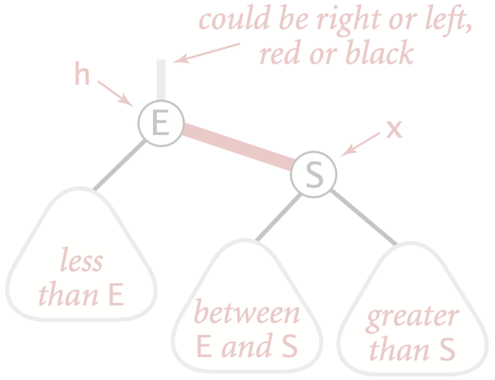



```
Node rotateRight(Node h)
{
   Node x = h.left;
   h.left = x.right;
   x.right = h;
   x.color = h.color;
   h.color = RED;
   x.N = h.N;
   h.N = 1 + size(h.left)
           + size(h.right);
   return x;
}
```

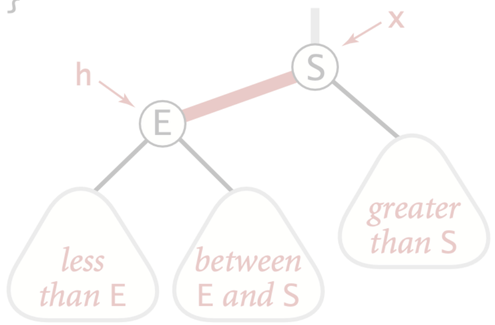


Right rotate (left link of h)

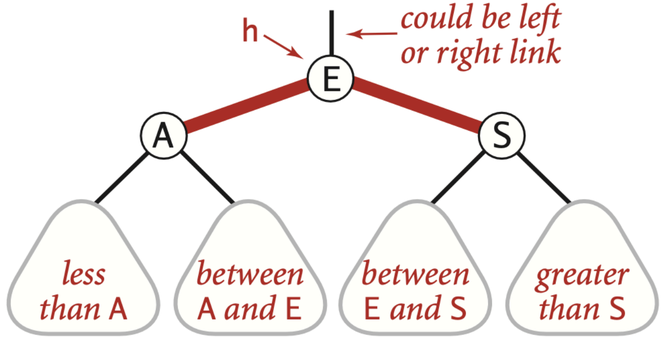
LLRB Operations



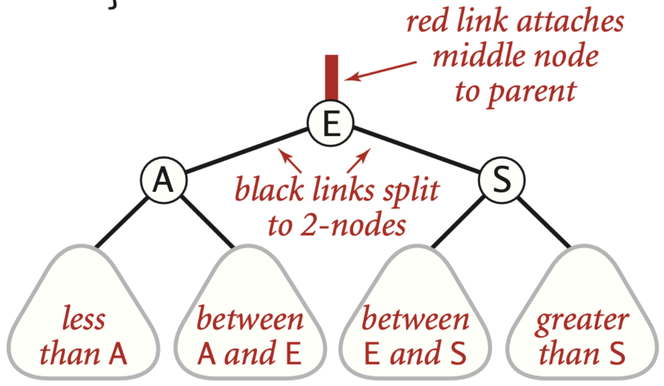

```
Node rotateLeft(Node h)
{
   Node x = h.right;
   h.right = x.left;
   x.left = h;
   x.color = h.color;
   h.color = RED;
   x.N = h.N;
   h.N = 1 + size(h.left)
           + size(h.right);
   return x;
}
```



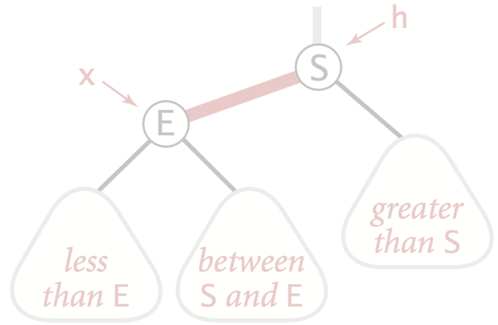
Left rotate (right link of h)



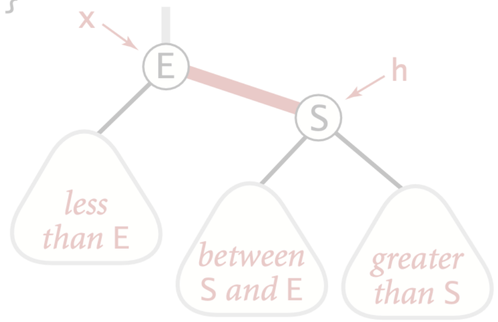

```
void flipColors(Node h)
{
   h.color = RED;
   h.left.color = BLACK;
   h.right.color = BLACK;
}
```



Flipping colors to split a 4-node

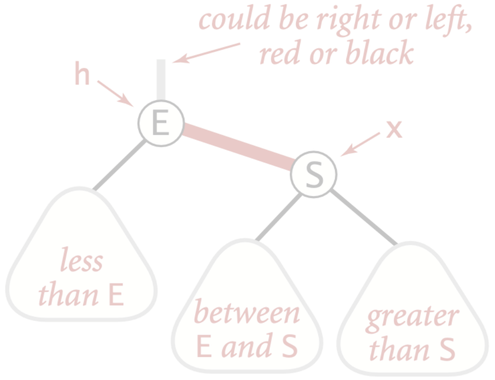



```
Node rotateRight(Node h)
{
   Node x = h.left;
   h.left = x.right;
   x.right = h;
   x.color = h.color;
   h.color = RED;
   x.N = h.N;
   h.N = 1 + size(h.left)
           + size(h.right);
   return x;
}
```

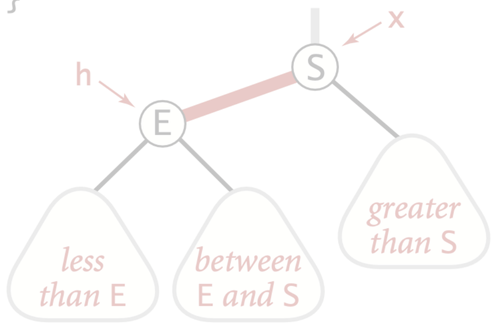


Right rotate (left link of h)

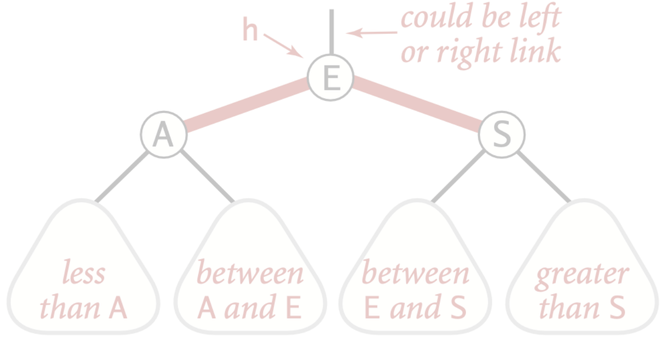
LLRB Operations



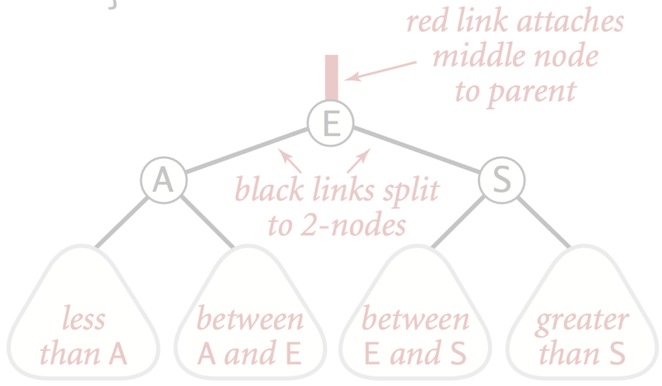

```
Node rotateLeft(Node h)
{
   Node x = h.right;
   h.right = x.left;
   x.left = h;
   x.color = h.color;
   h.color = RED;
   x.N = h.N;
   h.N = 1 + size(h.left)
           + size(h.right);
   return x;
}
```



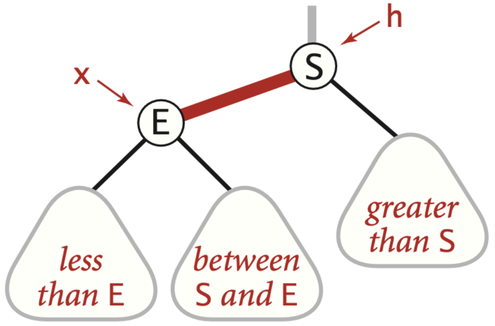
Left rotate (right link of h)



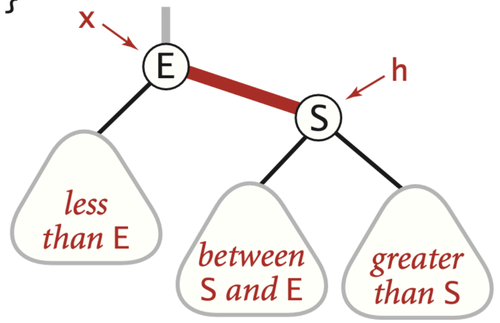

```
void flipColors(Node h)
{
   h.color = RED;
   h.left.color = BLACK;
   h.right.color = BLACK;
}
```



Flipping colors to split a 4-node




```
Node rotateRight(Node h)
{
   Node x = h.left;
   h.left = x.right;
   x.right = h;
   x.color = h.color;
   h.color = RED;
   x.N = h.N;
   h.N = 1 + size(h.left)
           + size(h.right);
   return x;
}
```



Right rotate (left link of h)

Putting it Together

Compare & Contrast: BST, Balanced Search Tree, and Red-Black BST