Graduate Algorithms

CS673-2016F-07

Red/Black Trees

David Galles

Department of Computer Science
University of San Francisco
07-0: Binary Search Trees

• Binary Trees
• For each node n, (value stored at node n) > (value stored in left subtree)
• For each node n, (value stored at node n) < (value stored in right subtree)
07-1: Example Binary Search Trees
07-2: Example Binary Search Trees

- Examples:
  - Finding an element
  - Inserting an element
  - Deleting an element
07-3: Running Times

- Best-Case upper limit on the time for insert/delete/find of an element for a BST with \( n \) elements?
- Worst-Case upper limit on the time for insert/delete/find for a BST with \( n \) elements?
- Expected upper limit on the time for insert/delete/find for a BST with \( n \) elements?
  - What would we mean by “expected” in this instance?
07-4: Running Times

- Best-Case upper limit on the time for insert/delete/find of an element for a BST with $n$ elements?
  - $O(\lg n)$, if the tree is balanced
- Worst-Case upper limit on the time for insert/delete/find for a BST with $n$ elements?
  - $O(n)$, if the tree is a list
- Expected upper limit on the time for insert/delete/find for a BST with $n$ elements?
  - $O(\lg n)$, if elements are inserted in random order
Balanced BSTs

- We can guarantee $O(\lg n)$ running time for insert/find/delete if we can guarantee the tree is balanced.

- Several methods for guaranteeing a balanced tree:
  - AVL trees & Red-Black trees are the most common.
  - We’ll look at Red-Black Trees.
• Red-Black Trees as Binary Search trees, with “Null Leaves”
  • Examples of BSTs with “Null Leaves”
  • (Null leaves are mostly a notational convenience)
Red-Black Trees

Red-Black Trees are Binary Search trees, with “Null Leaves”, and the following properties:

- Every Node is either Red or Black
- (Root is Black) <Not strictly required>
- Each null “leaf” is Black
- If a node is red, both children are black
- For each node, all paths from the node to descendant leaves contain the same number of black nodes
07-8: Red-Black Trees
Example Red-Black tree ("Null Leaves" left out for clarity)
07-10: Red-Black Trees

- In a Red-Black tree, what is the greatest possible difference in the length of the path from the root to two different leaves?
- What is the largest height of a Red-Black tree that contains \( n \) elements?
Let $bh(X)$ be the "Black Height" of a node – the number of black nodes on a path from that node to a leaf (not including the node itself)

The subtree rooted at any node $X$ has at least $2^{bh(X)} - 1$ internal (non-leaf) nodes

- Proof by induction (on board)
07-12: Tree Rotations

Right Rotate

Left Rotate
07-13: Tree Insertions

- Always insert red nodes
- Which property would be violated by inserting a red node?
Tree Insertions

- Always insert red nodes
- Which property would be violated by inserting a red node?
  - Could have a red node with a red child
- Fix using tree rotations
To fix a red node with red child:

- Case 1: Uncle is red
- Case 2: Uncle is black, Inserted node is right child of parent, and parent is a left child of Grandparent (or node is left child, parent is right child)
- Case 3: Uncle is black, Node is left child of parent, parent is left child of Grandparent (or node is right child, parent is right child)
07-16: Case 1

- Red Uncle
07-17: Case 1

- Red Uncle
Case 2

- Black Uncle / parent child different handedness
Case 3

- Black Uncle / parent child same handedness
07-20: **Case 3**

- Black Uncle / parent child same handedness

![Diagram showing relationships between A, B, C, and dotted lines for α, β, γ, δ.](attachment:image.png)
07-21: Deleting nodes

- Deleting nodes
  - Delete nodes just like in standard BST
  - Which properties could be violated by deleting a red node?
    - Each node red or black
    - Black Root
    - Each red node has 2 black children
    - Black path length to leaves same for each node
Deleting nodes

- Delete nodes just like in standard BST
- Which properties could be violated by deleting a red node?
  - None!
Deleting nodes

- Delete nodes just like in standard BST
- Which properties could be violated by deleting a black node?
  - Each node red or black
  - Black Root
  - Each red node has 2 black children
  - Black path length to leaves same for each node
07-24: Deleting Nodes

• Deleting black node
  • If the child of the deleted node is red ... (show example on board)
07-25: Deleting Nodes

- Deleting black node
  - If the child of the deleted node is black
    - Make the child “doubly black”
    - Push “extra blackness” up the tree until it can be removed by a rotation
X is “doubly black” node, X is a left child

Case 4:
- X’s sibling W is black, and W’s right child is red
- Can remove “double-blackness” of X with a single rotation
07-27: Deleting Nodes

Doubly Black

A

B

C

D

E

α β

γ δ ε ζ
07-28: Deleting Nodes

Diagram of a tree structure with nodes labeled D, B, C, E, A, α, β, γ, δ, ε, and ζ.
07-29: Deleting Nodes

Doubly Black

A

B

C

D

E

α

β

γ

δ

ε

ζ
07-30: Deleting Nodes
• X is “doubly black” node, X is a left child
  • Case 3:
    • X’s sibling W is black, and W’s left child is red, and right child is black
    • Single rotation to get to previous case
Deleting Nodes

Doubly Black

A

B

C

D

E

α β

γ δ

ε ζ
07-33: Deleting Nodes

Doubly Black

A
  / 
 α β

B

C
  / 
 γ

D
  / 
 δ

E
  / 
 ε

ζ
07-34: Deleting Nodes

Doubly Black

A

α β

B

C

γ δ

D

E

ε ζ
07-35: Deleting Nodes

Doubly Black

A
α β

B

C
γ

D
δ

E
ε ζ
07-36: Deleting Nodes

- X is “doubly black” node, X is a left child
  - Case 2:
    - X’s sibling W is black, and both of W’s children are black
    - Push “Blackness” of X and W to parent
07-37: Deleting Nodes

Doubly Black
07-38: Deleting Nodes

A
\(\alpha\) \(\beta\)

B

C
\(\gamma\) \(\delta\)

D

E
\(\varepsilon\) \(\zeta\)
07-39: Deleting Nodes

Doubly Black

A: \(\alpha\) \(\beta\) B: 
C: \(\gamma\) \(\delta\) D: 
E: \(\varepsilon\) \(\zeta\)
07-40: Deleting Nodes

Doubly Black

A
α
β
C
γ
δ
E
ε
ζ
B
D
07-41: Deleting Nodes

- X is “doubly black” node, X is a left child
  - Case 2:
    - X’s sibling W is Red
    - Do a rotation, to make W black. Then one of the other cases will apply.
07-42: Deleting Nodes

Doubly Black

A

α β

B

C

γ δ

D

E

ε ζ
Deleting Nodes

Doubly Black
07-44: Deleting Nodes

- Need to include symmetric cases
  - In all of the previous examples, swap left/right
  - (Go over at least one example)