Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

EMİRHAN KARAGÖZOĞLU
151044052

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

1.1 Problem Solution Approach

RedBlackTree classı BinarySearchWithRotate classından extend olmaktadır. BinarySearchWithRotate classı da BinaryTree classından extend olan BinarySearchTree classından extend olmaktadır. RedBlackTree classının add methodu bulunmaktadır. Bu method içerisinde helper moveBlackDown methodu bulunmaktadır.

Pseudocode of add method:

```
1.  if the item is equal to root.data
2.      The item is already in the tree; return false.
3.  else if the item is less than root.data
4.      if the left subtree is null
5.          Insert a new Red-Black node as the left subtree and color it red.
6.          Return true;
7.      else
8.          if both the left child and the right child are red
9.              Change the color of the children to black and change local root to red
10.         Recursively insert the item to the left subtree
11.         if the left child is now red
12.             if the left grandchild is now red
13.                 Change the color of the left child to black and change the root to red.
14.                 Rotate the local root right.
15.             else if  the right grandchil is now red
16.                 Rotate the left child left.
17.                 Change the color of left child to black and change the local root to red.
18.                 Rotate the local root right.
19. else
20.     Item greater than root.data; process the symmetry of what we do for the left tree
```

Pseudocode of moveBlackDown method:

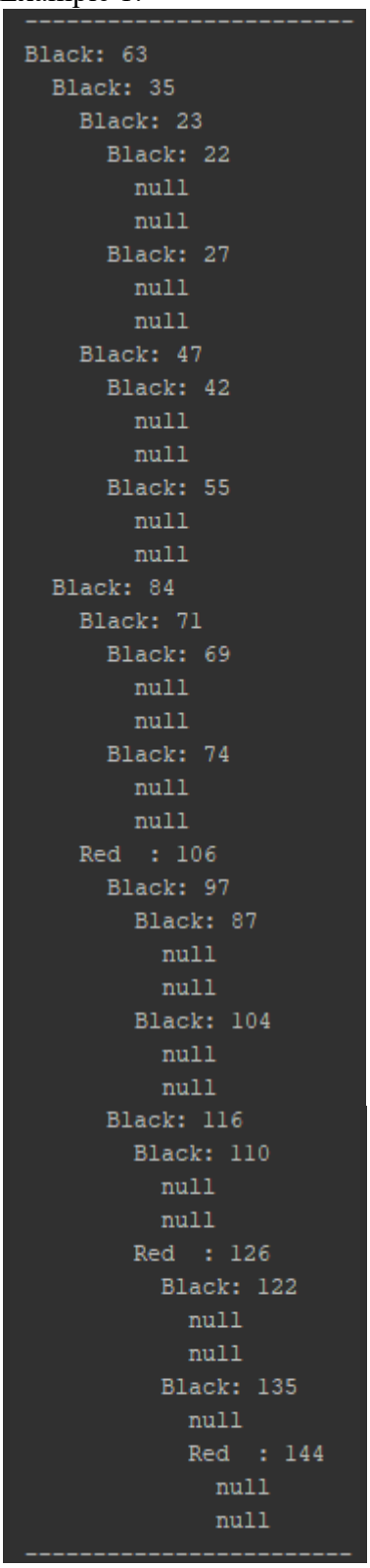
```
1.  if root is black and both of it's children are red
2.      make root red and make both chidren black
```

1.2 Test Cases

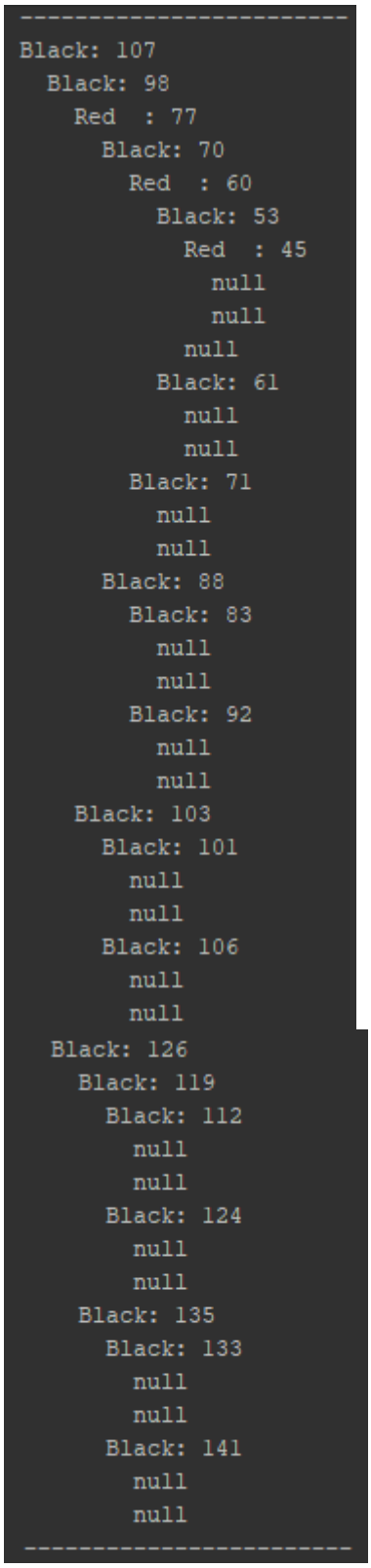
6 yüksekliğinde worst Red-Black tree oluşturmak için sıralı olarak 22 adet ekleme yapılmalıdır. Yükseklik kavramı için net bir tanım yoktur. Bazı kaynaklarda ağacın yükseliği hesaplanır root 0 kabul edilirken bazı kaynaklarda ise 1 kabul edilmektedir. Ben ağacı oluştururken root u 0 kabul ederek 6 yüksekliğinde bir ağaç oluşturdum ve bunun içinde minimum 22 ekleme yapılmaktadır. (Eğer root 1 olarak kabul edilse idi 14 ekleme yeterli olucaktı.) Oluşturulan bu RedBlack treenin worst olması için maksimum sayıda rotate yapması gerekmektedir. Sıralı olarak ekleme yapıldığında bu durum sağlanmaktadır. İlk örnekte küçükten büyüğe (sağa yatık) ikinci örnekte ise büyükten küçüğe (sola yatık) random ağaçlar oluşturdum. Her ekleme işlemi yapıldığında RedBlack Tree ekrana bastırıldı. Tüm ağacın bu şekilde çıktısı çok uzun olacağı için rapora sadece ağaçların son hallerinin screenshot'ını koyacağım. Step-step ekleme adımlarını görmek için Q1 main çıktısına bakabilirsiniz. Her defa çalıştırıldığında aynı mantıkla farklı sayılarla RedBlack Tree oluşturulur.

1.3 Running Commands and Results

Example 1:



Example 2:



2 binarySearch method

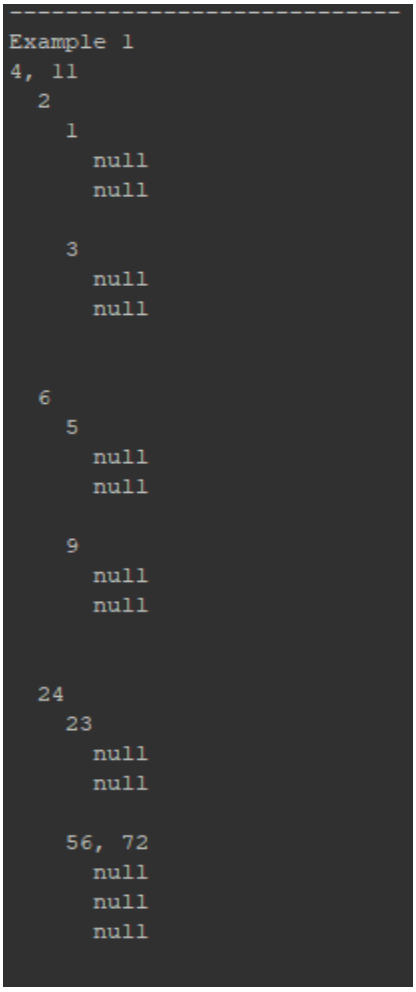
2.1 Problem Solution Approach

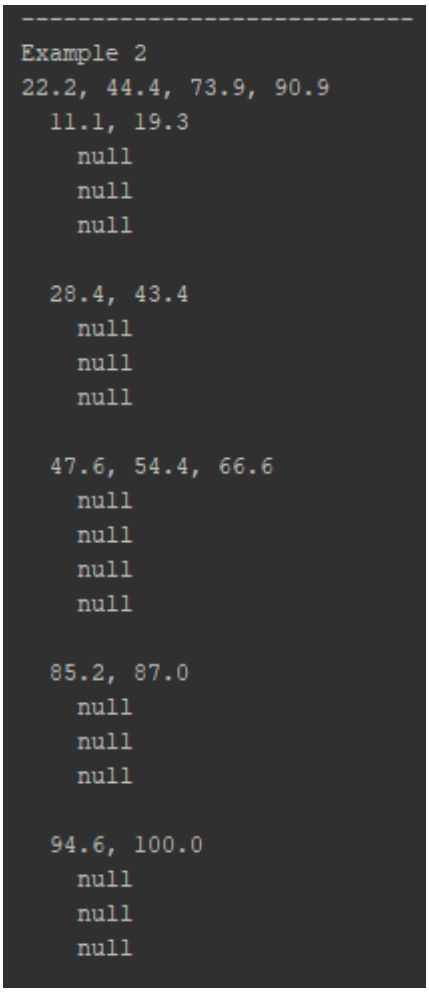
BTree class'ı SearchTree interface'ini impliment etmektedir. Bu partta BTree classının eksik olan binarySearch methodu impliment edildi. Bu method Btree'nin her bir node'undaki arrayler içerinde eleman aramak için yazılmış private helper bir methoddur. Add methodunun içinde bulunan insert methodunda kullanılmaktadır. Normal BinarySearch den farklı olarak aranan eleman bulunamadığında array üzerinde uygun bir index return etmektedir(normalde -1 return edilir). Bu index belirlenirken sorted arrayın bozulmaması göz önünde bulundurulur.

2.2 Test Cases

binarySearch methodunu add methodu kullandığı için belirli eklemeler yaparak BTree yi gözlemlersek binarySearch methonunun çalıştığını görebiliriz. BinarySearch methodu ekleme yaparken daha önce bu elemanın olup olmadığını kontrol etmekle kalmayıp eleman yok ise ona uygun bir index return etmektedir. Buda BTree nin her bir node'unun sıralı olmasını sağlar. Test ederken 3 order'a sahip integer bir BTree ve 5 order'a sahip bir double BTree oluşturuldu ve eleman eklenerek ekrana bastırıldı. Çıktılardan görülebileceği üzere BTree lerin her node'u sıralı haldedir. Bu da BinarySearch methodunun doğru çalıştığının bir göstergesidir.

2.3 Running Commands and Results





3 Project 9.5 in book

3.1 Problem Solution Approach

AVLTree classı BinarySearchWithRotate classından extend olmaktadır. BinarySearchWithRotate classı da BinaryTree classından extend olan BinarySearchTree classından extend olmaktadır. AVLTree classının AVLNode isimli BinaryTree nin Inner Node classından extend olan bir classı vardır. AVLTree'nin public add ve delete, private helper rebalanceLeft, rebalanceRight, decrementBalance, incrementBalance, isAVLTree ve height methodları vardır. AVLTree'nin BinaryTree alan constructor'ında gelen BinaryTree bir AVLTree değilse exception atılır.

Pseudocode of add method:

```
1.  if the root is null
2.      Create a new tree with the item at the root and return true.
3.  else if the item is equal to root.data
4.      The item is already in the tree; return false.
5.  else if item is less than root.data
6.      Recursively insert the item in the left subtree.
7.      if the height of the left subtree has increased
8.          decrement balance.
9.          if balance is zero, reset increase to false.
10.         if balance is less than -1
11.             reset increase to false.
12.             perform a rebalanceLeft.
13. else if item is greater than root.data
14.     Process the symmetry of what we do for the left tree
```

Pseudocode of delete method:

```
1.  Clean all tree.
2.  Add all element of old tree except the element to be deleted.
```

Pseudocode of rebalanceLeft method:

```
1.  if the left subtree has a positive balance (Left-Right case)
2.      if the left-left subtree has a negative balance
3.          set the left subtree balance to -1.
4.          set the left-right subtree balance to 0.
5.          set the local root balance to 0.
6.      else if left-left subtree has a positive balance
7.          set the left subtree balance to 0.
8.          set the left-right subtree balance to 0.
9.          set the local root balance to +1.
10.     else
11.         set the left subtree balance to 0.
12.         set the local root balance to 0.
13.     rotate the left subtree left.
14. else (Left-Left case)
15.     set the left subtree balance to 0.
16.     set the local root balance to 0.
17.  rotate the local root right
```

Pseudocode of rebalanceRight method:

1. if the right subtree has a positive balance (Right-Left case)
2. if the right-right subtree has a negative balance
3. set the right subtree balance to 0.
4. set the right-left subtree balance to 0.
5. set the local root balance to -1.
6. else if right-right subtree has a positive balance
7. set the right subtree balance to +1.
8. set the right-left subtree balance to 0.
9. set the local root balance to 0.
10. else
11. set the right subtree balance to 0.
12. set the local root balance to 0.
13. rotate the right subtree right.
14. else (Right-Right case)
15. set the right subtree balance to 0.
16. set the local root balance to 0.
17. rotate the local root right

Pseudocode of decrementBalance method:

1. Decrement balance
2. if balance is 0
3. set increase 0

Pseudocode of incrementBalance method:

1. Increment balance
2. if balance is 0
3. set increase 0

Pseudocode of isAVLTree method:

1. if root is null; return true.
2. if the difference between the height of the left and right tree is lower than 2 and left subtree and right subtree is an AVL tree; return true.
3. else; return false.

Pseudocode of height method:

1. if root is null; return 0.
2. return max of right and left subtree's height + 1.

3.2 Test Cases

İlk olarak bir BinarySearchTree objesi oluşturulup içine AVLTree kurallarına uymayacak şekilde ekleme yapıldı. Daha sonra bir AVLTree objesi bu BinarySearchTree(BST is a Binary Tree) objesi ile oluşturuldu. Exception firlatıldığı gösterildi. Daha sonra normal bir AVLTree objesi oluşturulup bir kaç eleman eklendi ve oluşan ağaç ekrana bastırıldı. Son olarak da oluşan bu AVLTree den bir eleman silindi ve ağaç ekrana bastırıldı. add ve delete methodların içerisinde rebalanceLeft, rebalanceRight, incrimentBalance, decrementBalance methodları ; constructor'ın içerisinde isAVLTree ve height methodları kullanıldığı için add, delete ve constructorun çalışmasının gösterilmesi bu methodlarında çalıştığını gösterir.

3.3 Running Commands and Results

```
------------------------
Binary Tree
11
  2
    null
    null
  13
    12
      null
      null
    15
      null
      16
        null
        null

This Binary Tree is not AVL Tree!
------------------------
AVL Tree
-1: 48
  -1: 32
    -1: 23
      -1: 19
        0: 11
          null
          null
        null
      0: 28
        null
        null
    -1: 45
      0: 37
        null
        null
      null
  1: 76
    0: 63
      null
      null
    -1: 98
      0: 84
        null
        null
      null

------------------------
32 was deleted
0: 48
  0: 28
    0: 19
      0: 11
        null
        null
      0: 23
        null
        null
    -1: 45
      0: 37
        null
        null
      null
  1: 76
    0: 63
      null
      null
    -1: 98
      0: 84
        null
        null
      null
```