CENG 201 Veri Yapıları 6: AVL Ağaçları Kodu

Öğr.Gör. Şevket Umut ÇAKIR

Pamukkale Üniversitesi

Hafta 6

Anahat

Düğüm Sınıfı

```
public class AVLNode<T> where T : IComparable
{
    public T element;
    public AVLNode<T> left;
    public AVLNode<T> right;
    public int height;
    public AVLNode (){ }
    public AVLNode (T element)
    {
        this.element = element;
    }
    public AVLNode (T element, AVLNode<T> left, AVLNode<T> right)
    {
        this.element = element;
        this.left = left;
        this.right = right;
        this.height = 0;
    }
}
```

Metodlar

```
private int height(AVLNode<T> node)
public bool isEmpty(AVLNode<T> node)
public T findMin()
AVLNode<T> findMin(AVLNode<T> node)
public T findMax()
AVLNode<T> findMax(AVLNode<T> node)
AVLNode<T> rotateWithLeftChild(AVLNode<T> k2)
AVLNode<T> doubleWithLeftChild(AVLNode<T> k3)
AVLNode<T> rotateWithRightChild(AVLNode<T> k1)
AVLNode<T> doubleWithRightChild(AVLNode<T> k1)
public void insert(T element)
private AVLNode<T> insert(AVLNode<T> node, T element)
private AVLNode<T> balance(AVLNode<T> node)
public void print()
void print(AVLNode<T> node, int max)
public void remove(T value)
AVLNode<T> remove(AVLNode<T> node, T value)
```

height ve isEmpty metodları

```
private int height(AVLNode<T> node)
{
  return node == null ? 0 : node.height;
}
public bool isEmpty(AVLNode<T> node)
{
  return node == null;
}
```

findMin metodu

```
public T findMin()
{
  if (isEmpty(root)) {
    throw new Exception ("AVL ağacı boş");
  }
  return findMin (root).element;
}
AVLNode<T> findMin(AVLNode<T> node)
{
  if (node == null)
    return null;
  while (node.left != null)
    node = node.left;
  return node;
}
```

findMax metodu

```java
public T findMax()
{
  if (isEmpty(root)) {
    throw new Exception ("AVL ağacı boş");
  }
  return findMax (root).element;
}
AVLNode<T> findMax(AVLNode<T> node)
{
  if (node == null)
    return null;
  while (node.right != null)
    node = node.right;
  return node;
}
```

Tekli Döndürme(Sola)

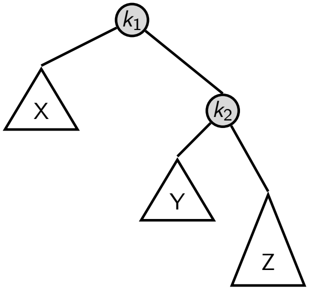


Figure: Durum 4'ün çözümü

Tekli Döndürme(Sola)

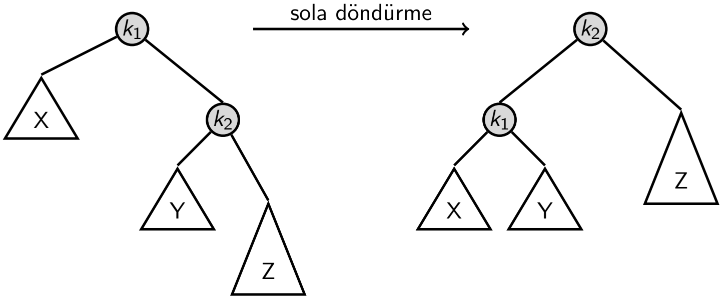


Figure: Durum 4'ün çözümü

Sola döndürme

```
/*
 *            k1             k2
 *           /  \           /  \
 *         T1   k2   ->   k1   T3
 *             /  \      /  \
 *            T2  T3    T1  T2
 */
AVLNode<T> rotateWithRightChild(AVLNode<T> k1)
{
  AVLNode<T> k2 = k1.right;
  k1.right = k2.left;
  k2.left = k1;
  k1.height = Math.Max (height (k1.left), height (k1.right)) + 1;
  k2.height = Math.Max (height (k2.right), k1.height) + 1;
  return k2;
}
```

Tekli Döndürme(Sağa)

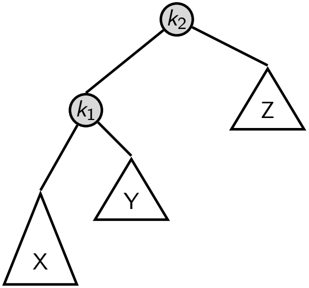


Figure: Durum 1'in çözümü

Tekli Döndürme(Sağa)

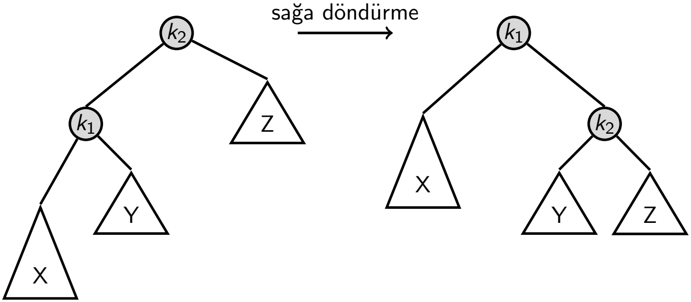


Figure: Durum 1'in çözümü

Sağa döndürme

```
/*
 *          k2            k1
 *         /  \          /  \
 *       k1   T3   ->  T1   k2
 *      /  \               /  \
 *    T1   T2             T2   T3
 */
AVLNode<T> rotateWithLeftChild(AVLNode<T> k2)
{
  AVLNode<T> k1 = k2.left;
  k2.left = k1.right;
  k1.right = k2;
  k2.height = Math.Max (height (k2.left), height (k2.right)) + 1;
  k1.height = Math.Max (height (k1.left), k2.height) + 1;
  return k1;
}
```

Çift Döndürme(sağ, sol)

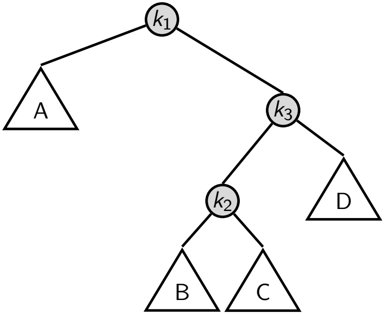


Figure: Durum 3'ün çözümü

Çift Döndürme(sağ, sol)

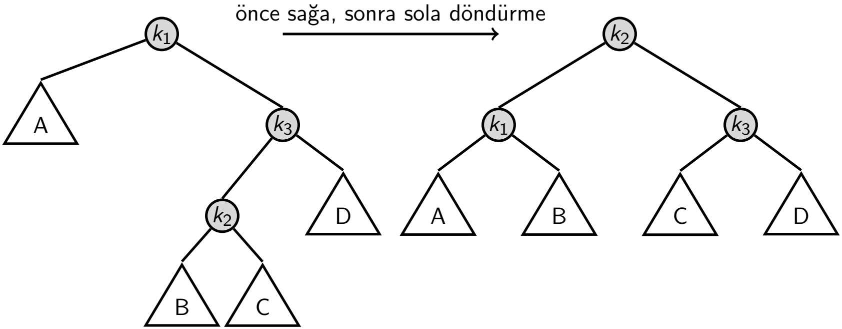


Figure: Durum 3'ün çözümü

Önce sağa sonra sola döndürme

```
/*
 *            k3                k1
 *           /  \             /      \
 *         k2   T4          k2       k3
 *        /  \       ->    /  \     /  \
 *      T1   k1          T1   T2  T3   T4
 *           /  \
 *         T2   T3
 */
AVLNode<T> doubleWithLeftChild(AVLNode<T> k3)
{
  k3.left = rotateWithRightChild (k3.left);
  return rotateWithLeftChild (k3);
}
```

Çift Döndürme(sol, sağ)

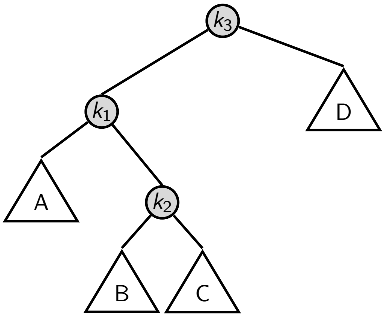


Figure: Durum 2'nin çözümü

Çift Döndürme(sol, sağ)

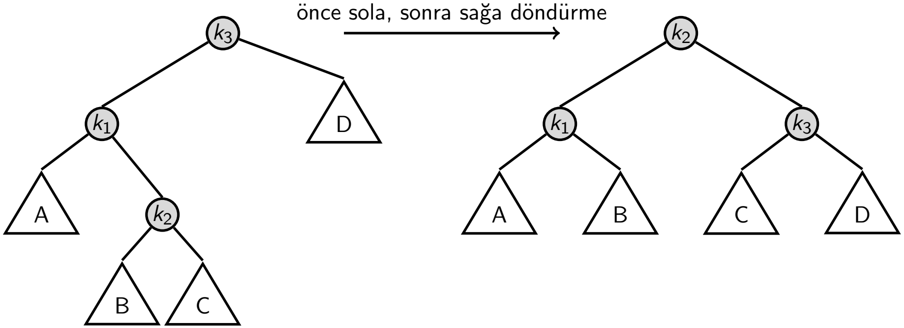


Figure: Durum 2'nin çözümü

Önce sola sonra sağa döndürme

```
/*
*            k1               k3
*           /  \            /      \
*         T1   k2   ->   k1        k2
*             /  \       /  \    /  \
*            k3   T4    T1   T2  T3   T4
*           /  \
*         T2   T3
*/
AVLNode<T> doubleWithRightChild(AVLNode<T> k1)
{
  k1.right = rotateWithLeftChild (k1.right);
  return rotateWithLeftChild (k1);
}
```

Ekleme

```
public void insert(T element)
{
  root = insert (root, element);
}
private AVLNode<T> insert(AVLNode<T> node, T element)
{
  if (node == null)
    return new AVLNode<T> (element);
  int compareResult = node.element.CompareTo (element);
  if (compareResult < 0)
    node.right = insert (node.right, element);
  else if (compareResult > 0)
    node.left = insert (node.left, element);
  return balance (node);
}
```

Dengeyi sağlama

```java
private AVLNode<T> balance(AVLNode<T> node)
{
  if (node == null)
    return node;
  int balanceFactor = height (node.right) - height (node.left);
  if (balanceFactor > 1) {
    if (height (node.right.right) >= height (node.right.left))
      node = rotateWithRightChild (node);//Sağa döndür
    else
      node = doubleWithRightChild (node);//Önce sol, sonra sağa döndür
  } else if (balanceFactor < -1) {
    if (height (node.left.left) >= height (node.left.right))
      node = rotateWithLeftChild (node);
    else
      node = doubleWithLeftChild (node);
  }
  node.height = Math.Max (height (node.left), height (node.right)) + 1;
  return node;
}
```

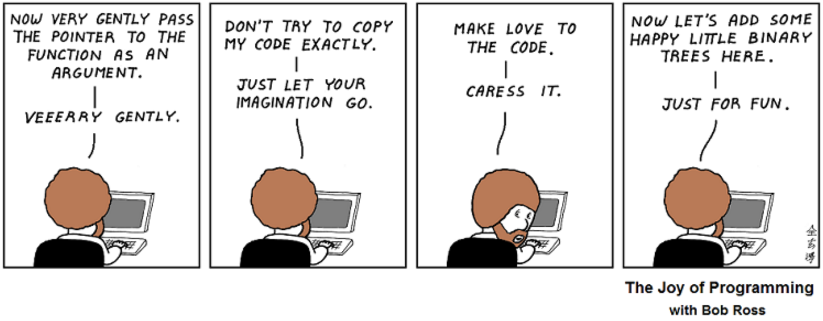
Yazdırma

```
public void print()
{
  print (root, root.height);
}
void print(AVLNode<T> node, int max)
{
  if (node == null)
    return;
  int s=max-height(node);
  for (int i = 0; i < s; i++) {
    Console.Write ("|  ");
  }
  Console.WriteLine ("|{1,-2}",new String(' ',s), node.element);
  print (node.left, max);
  print (node.right, max);
}
```

Silme

```
public void remove(T value)
{
  root = remove (root, value);
}
AVLNode<T> remove(AVLNode<T> node, T value)
{
  if (node == null)
    return node;
  int compareResult = value.CompareTo (node.element);
  if (compareResult < 0)
    node.left = remove (node.left, value);
  else if (compareResult > 0)
    node.right = remove (node.right, value);
  else if (node.left != null && node.right != null) {//iki çocuk
    node.element = findMin (node.right).element;
    node.right = remove (node.right, node.element);
  } else
    node = node.left != null ? node.left : node.right;
  return balance (node);

}
```


NOW VERY GENTLY PASS THE POINTER TO THE FUNCTION AS AN ARGUMENT.
VEEERRY GENTLY.
DON'T TRY TO COPY MY CODE EXACTLY.
JUST LET YOUR IMAGINATION GO.
MAKE LOVE TO THE CODE.
CARESS IT.
NOW LET'S ADD SOME HAPPY LITTLE BINARY TREES HERE.
JUST FOR FUN.
The Joy of Programming
with Bob Ross

1

[1] abstrusegoose