Informatica

Deel II: les 10

Bomen

Jan Lemeire

Informatica deel II

februari – mei 2012

Vrije Universiteit Brussel

De volgende revolutie

INTERNET!

Technologie

- Netwerk laat toe dat computers berichtjes zenden naar elkaar
- Adres van elke computer: ip-adres
  - 192.168.0.233
  - Windows-commando: *ipconfig*
  - Domeinnaam is een *alias* voor het nummer
- IP= Internet Protocol
- Bericht wordt opgedeeld in ip-pakketjes die hun weg naar de destinatie zoeken over het net
  - Oud telefoonnetwerk: je had de hele lijn voor je
  - Het zoeken van de weg: TCP-protocol (Transmission Control Protocol)
- **TCP/IP-protocol**

ARPANET

- Eisen aan digitaal communicatiesysteem:
  - Flexibel
  - Gedecentraliseerd, geen ‘baas’
  - Onafhankelijk: delen kunnen op zich werken
  - Zelf-regulerend
  - robustness and survivability
    - including the capability to withstand losses of large portions of the underlying networks (due to a nuclear attack)
    - pakketjes kunnen verloren gaan: fouten of ‘overlopen’ van de queues bij bottlenecks

Microsoft < 1995

- Niet geïnteresseerd (ik ook niet)
- Op CERN wordt Netscape Navigator ontwikkeld, de eerste browser
  - Om informatie (file) van een remote computer te visualiseren
- Netscape wordt de grootste speler
  - Is nu Firefox

Microsoft reageert

- Lanceert Internet Explorer in 1995
- Koopt hotmail op
  - En onlangs nog skype (maar dominante positie lijkt nu verloren)
- Verovert het internet
  - Door macht van Windows

IT-sector: the winner takes it all

- Windows, office, pdf, Google, facebook, …
- ⇔ andere sectoren kunnen er meerdere spelers zijn

Generic sorting

Arrays class:

Method Summary

`static void` **sort**`(int[] a)`
Sorts the specified array of ints into ascending numerical order.

`static void` **sort**`(int[] a, int fromIndex, int toIndex)`
Sorts the specified range of the specified array of ints into ascending numerical order.

`static void` **sort**`(Object[] a)`
Sorts the specified array of objects into ascending order, according to the *natural ordering* of its elements.

`static void` **sort**`(Object[] a, int fromIndex, int toIndex)`
Sorts the specified range of the specified array of objects into ascending order, according to the *natural ordering* of its elements.

`static void` **sort**(T[] a, `Comparator`<? super T> c)
Sorts the specified array of objects according to the order induced by the specified comparator.

`static void` **sort**(T[] a, int fromIndex, int toIndex, `Comparator`<? super T> c)
Sorts the specified range of the specified array of objects according to the order induced by the specified comparator.

Voor lijsten

Collections class

`static void` **`sort`**`(List list)`

Sorts the specified list into ascending order, according to the *natural ordering* of its elements.

`static void` **`sort`**`(List list, Comparator c)`

Sorts the specified list according to the order induced by the specified comparator.

“Natuurlijke” ordening

Interface Comparable

Method Summary

`int` **`compareTo`**`(Object o)`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Voorbeeld

```
public class Student implements Comparable<Student>{
	String voornaam, naam;
	int rolnummer, score;
	Vak[] vakken;
	int[] punten;

	Student(String voornaam, String naam, int rolnummer){
		this.voornaam=voornaam;
		this.naam=naam;
		this.rolnummer=rolnummer;
		vakken = new Vak[4];
	}
	@Override
	public int compareTo(Student student2) {
	   int ordeNaam = this.naam.compareTo(student2.naam);
	   if (ordeNaam == 0) // beide dezelfde naam
	      return this.voornaam.compareTo(student2.voornaam);
	   else
	       return ordeNaam;
	}
```

Specifieke ordening

java.util

Interface Comparator<T>

Method Summary

int	**compare**(T o1, T o2) Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.
boolean	**equals**(Object obj) Indicates whether some other object is "equal to" this comparator.

Voorbeeld

```
class ComparatorOnScore implements Comparator<Student>{
    @Override
    public int compare(Student student1, Student student2) {
        return student1.score - student2.score;
    }
}
```

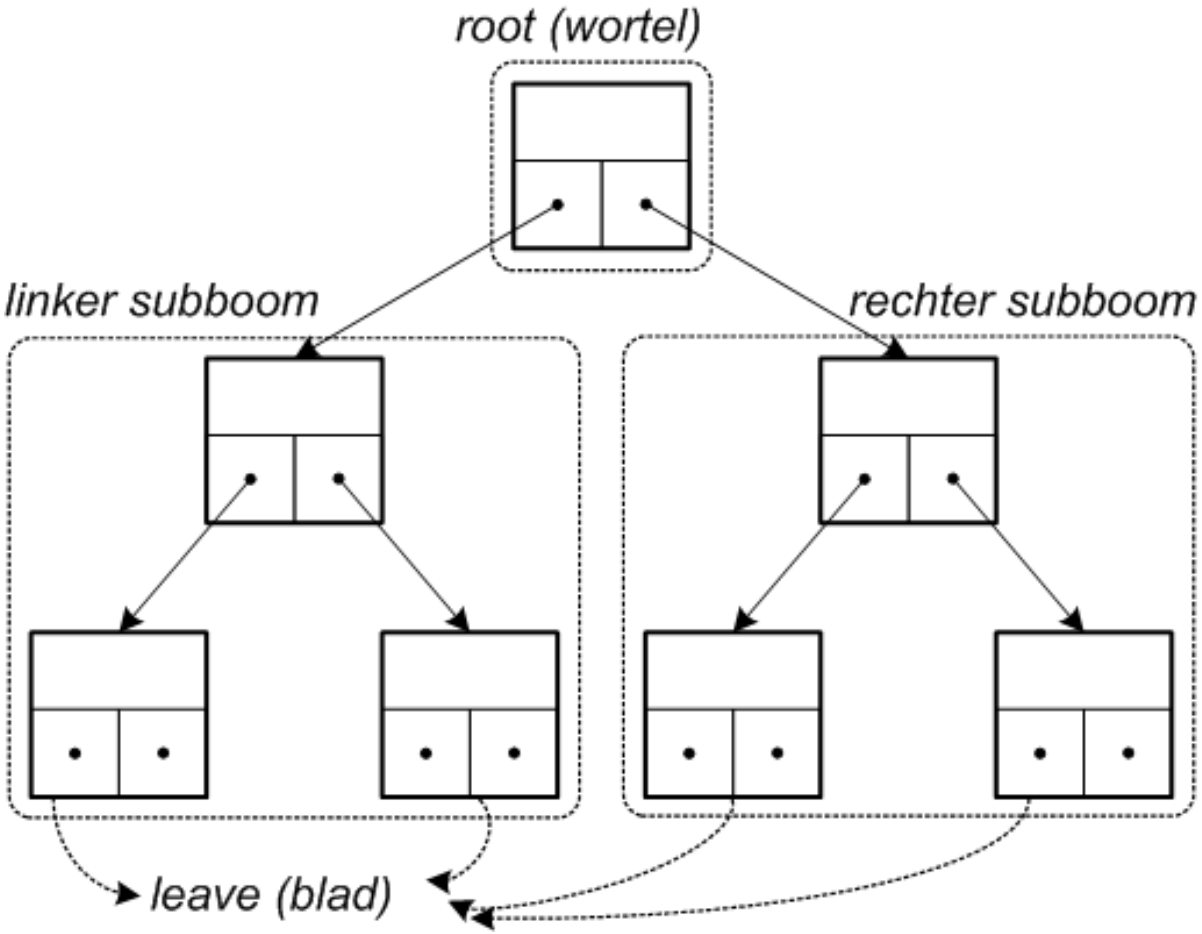
Sorteren gebeurt nu als volgt, met eigen comparator-object:

```
Arrays.sort(studenten, new ComparatorOnScore());
```

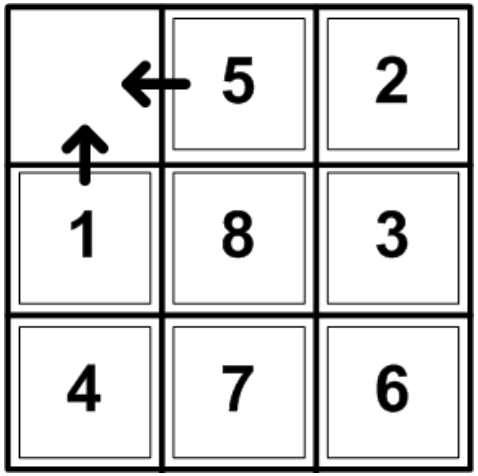
Bomen (hoofdstuk 7)

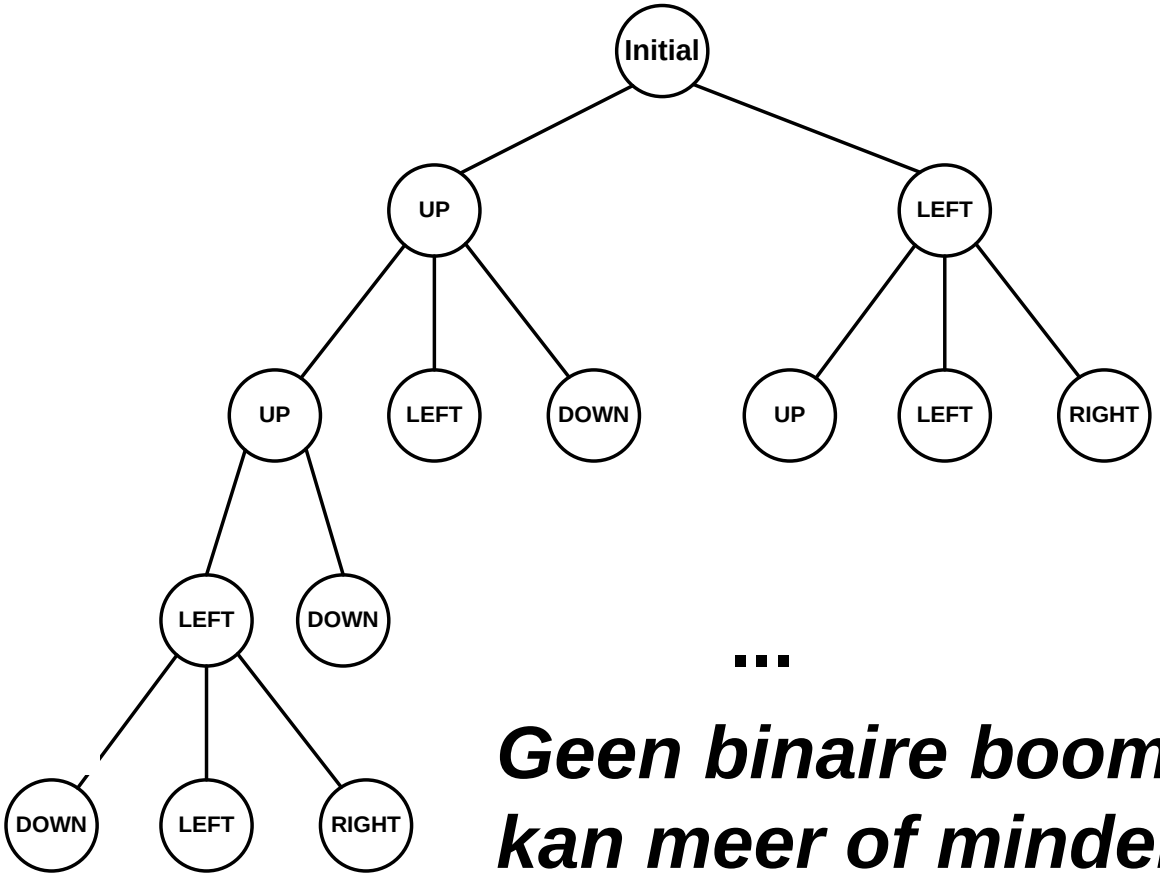
- Arrays
- Linked Lists
- Flexibele structuur om door te lopen

Binaire boom



Zoeken = aanmaken boom (les 7)



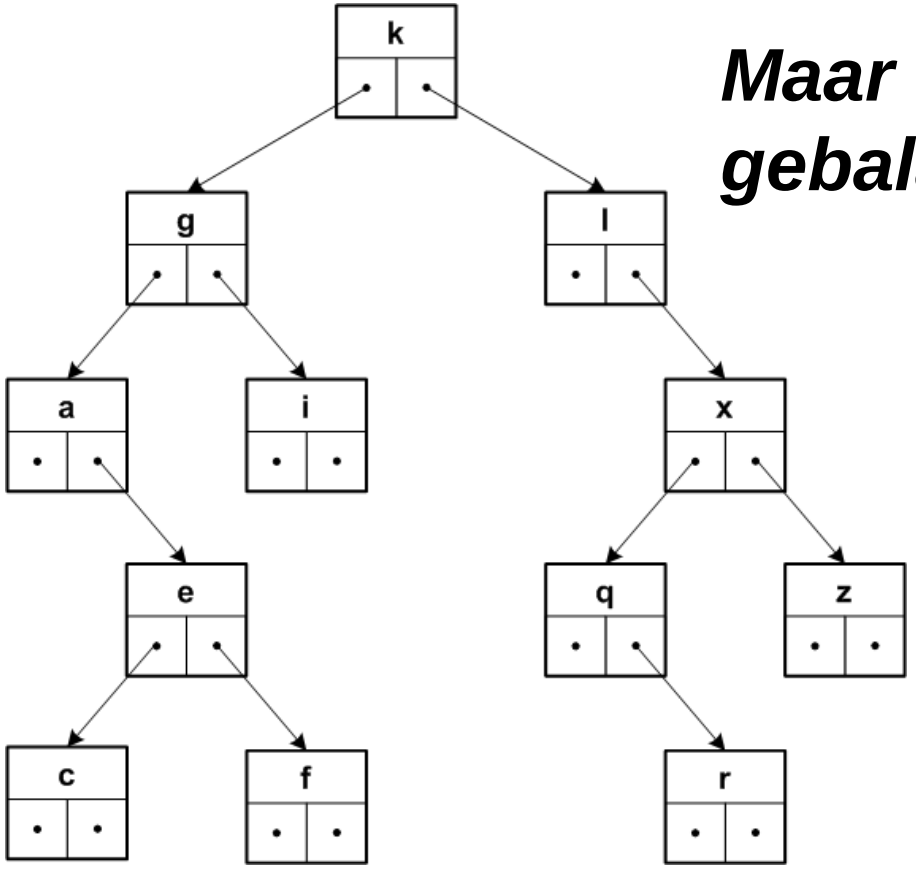


Oplossing puzzel zoeken

...

Geen binaire boom, kan meer of minder takken hebben

Binaire, geordende boom



Maar niet 100% gebalanceerd…

Elke node is een object

```
class Node<T>{
        T data;
        Node<T> left, right;

        Node(T data){
                this.data = data;
                this.left = null;
                this.right = null;
        }
}
```

Boom object

```
public class BinaryTree<T> {
      Node<T> root;
      Comparator<T> comparator;

      public BinaryTree(Comparator<T> comparator){
            this.comparator = comparator;
            root = null;
      }
}
```

Belangrijke operaties

1. Zoeken element
2. Toevoegen element
3. Printen boom
4. Verwijderen element
5. ‘Balanceren’ van de boom

1. Zoeken element

```
public boolean contains(T object){
        return find(root, object);
}
private boolean find(Node<T> current, T object){
        if (current == null)
                return false;
        else if (comparator.compare(current.data, object) == 0)
                return true;
        else if (comparator.compare(current.data, object) < 0)
                return find(current.left, object);
        else
                return find(current.right, object);
}
```

- Tijd <= diepte boom
- Idealiter: $log_2 n$
  - Behalve als ongebalanceerd

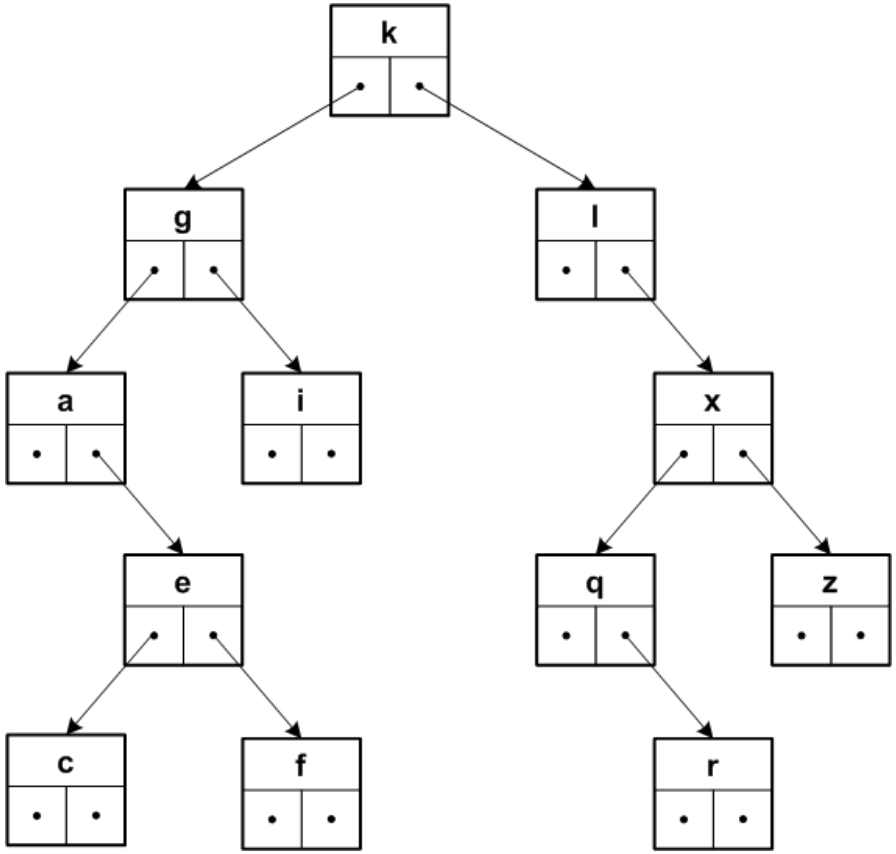
2. Toevoegen element

```
public void add(T object){
        Node<T> newNode = new Node<T>(object);
        if (root == null)
                root = newNode;
        else
                add(root, newNode);
}
private void add(Node<T> current, Node<T> newNode){
        if (comparator.compare(newNode.data, current.data) < 0){
                // moet links komen
                if (current.left == null)
                        current.left = newNode;
                else
                        add(current.left, newNode);
        } else {
                // rechts
                if (current.right == null)
                        current.right = newNode;
                else
                        add(current.right, newNode);
        }
}
```

Volgorde van belang!

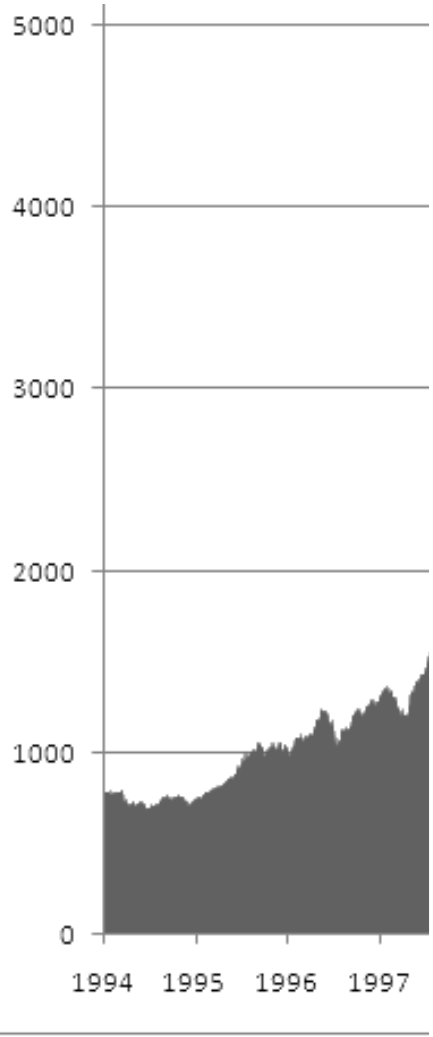
Wat als…

- {k, l, g, i, x, q, r, a, e, f, c, z}
- {a, c, e, f, g, i, k, l, q, r, x, z}
- {i, z, q, l, k, f, e, g, c, x, a, r}



De internetrevolutie

Technologie-index van USA: Nasdaq



De internetbubbel

- Internetzeepbel
- Dotcom-crisis
- Investeren!
- Leken steken hun geld in aandelenfondsen en aandelenclubs om de boot niet te missen
- Bubbel <= Hebzucht!

Informatica II: les 10

Jan Lemeire

De economie zou totaal veranderen => dotcom

- Internet: ongekende commerciële mogelijkheden
  - *Anders communiceren*
  - *Anders kopen*
  - Ook kerstbomen via het internet…
- Belangrijkste: aandacht (hits/leden/…)
  - "*Get large or get lost*"
  - agressieve marktpenetratie door middel van het uitbouwen van netwerken.
- extreme verliezen in het begin werden gezien als slechts investeringen

3. Printen van boom

```
public void preOrder(Node<T> node){
        if (node != null){
                System.out.print(node.data+", ");
                print(node.left);
                print(node.right);
        }
}
```

{k, g, a, e, c, f, i, l, x, q, r, z}

```
public void inOrder(Node<T> node){
        if (node != null){
                print(node.left);
                System.out.print(node.data+", ");
                print(node.right);
        }
}
```

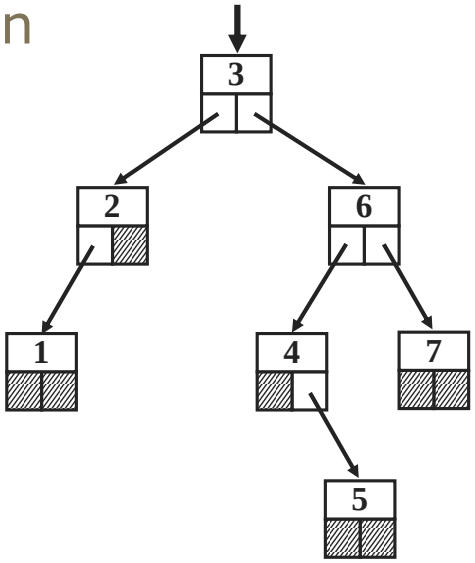
{a, c, e, f, g, i, k, l, q, r, x, z}

```
public void postOrder(Node<T> node){
        if (node != null){
                print(node.left);
                print(node.right);
                System.out.print(node.data+", ");
        }
}
```

{c, f, e, a, i, g, r, q, z, x, l, k}

4. Verwijderen van element

- 4 mogelijkheden:
  - Element is root
  - Element is niet in boom
  - Element heeft 0 of 1 subboom
  - Element heeft 2 subbomen



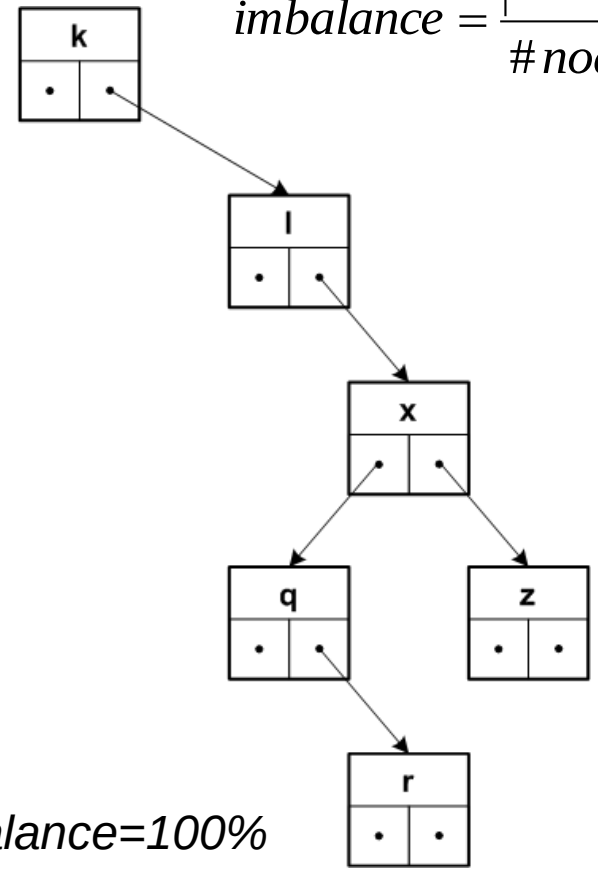
3 different situations :
a) 2 successors :
3, 6
b) 0 or 1 successor :
1, 2, 4, 5, 7
c) not present :
8

```
public boolean remove(T object){
        if (comparator.compare(root.data, object)== 0){
                root = createSubtree(root);
                return true;
        } else
                return findNodeAndRemove(root, object);
}

private boolean findNodeAndRemove(Node<T> current, T object){
        if (current == null)
                return false;

        if (current.left != null && comparator.compare(current.left.data,
object)== 0){
                current.left = createSubtree(current.left);
                return true;

        } else if (current.right != null
                 && comparator.compare(current.right.data, object)== 0){

                current.left = createSubtree(current.right);
                return true;

        } else if (comparator.compare(current.data, object) < 0)
                return findNodeAndRemove(current.left, object);
        else
                return findNodeAndRemove(current.right, object);
}
```

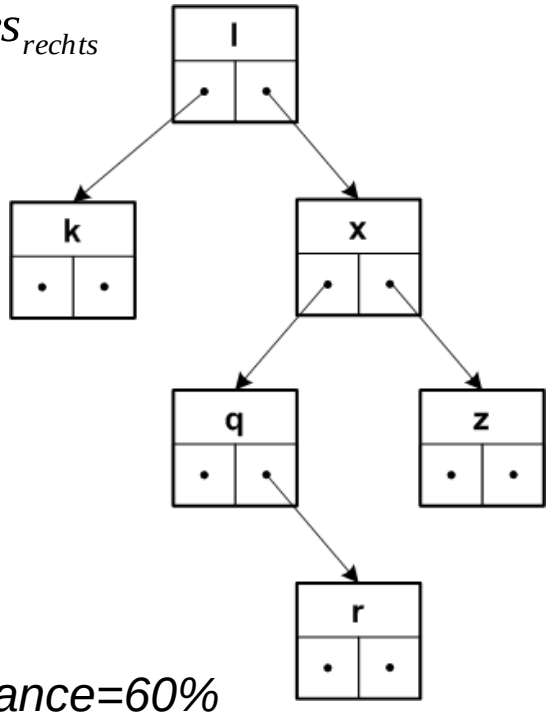
```
private Node<T> createSubtree(Node<T> current){
        if (current.left == null){
        // current.right is the only subtree that we should consider
                return current.right;
        } else if (current.right == null){
        // current.left is the only subtree that we should consider
                return current.left;
        } else {
                // promote rightmost node of left subtree
                if (current.left.right == null){
                        current.left.right = current.right;
                        return current.left;
                } else {
                   Node<T> rightMostNode = pickRightMostNode(current.left);
                        // he is new root of subtree
                        rightMostNode.right = current.right;
                        rightMostNode.left = current.left;
                        return rightMostNode;
                }
        }
}
private Node<T> pickRightMostNode(Node<T> current){
        // we expect current.right not to be null
        if (current.right.right != null){
                return pickRightMostNode(current.right);
        } else {
                Node<T> rightMostNode = current.right;
                current.right = rightMostNode.left;
                rightMostNode.left = null;
                return rightMostNode;
        }
}
```

5. Balanceren van de boom

$$imbalance = \frac{|\#nodes_{links} - \#nodes_{rechts}|}{\#nodes_{links} + \#nodes_{rechts}}$$



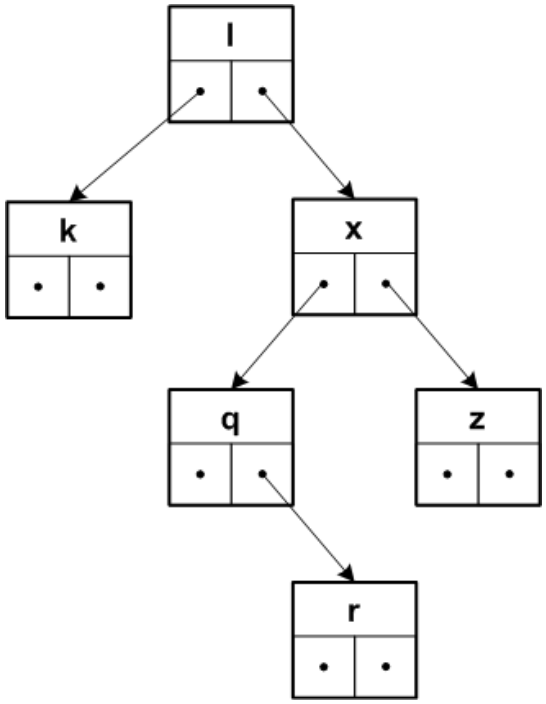
imbalance=100%

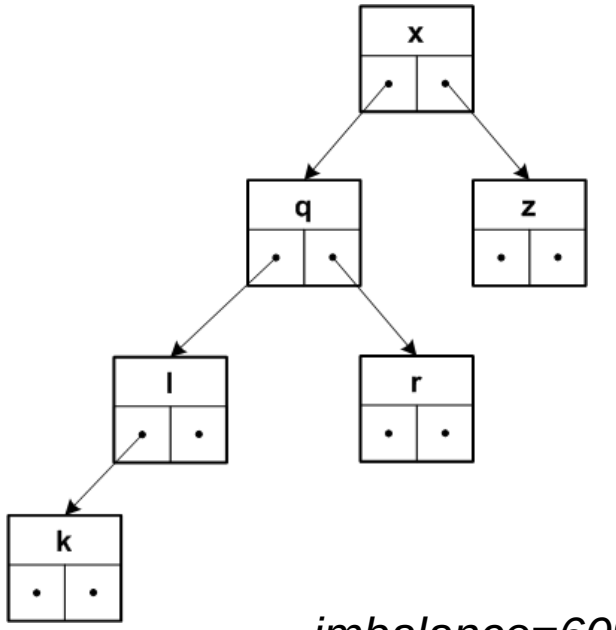


imbalance=60%

Promotie node 'l' tot root

Balanceren stap 2

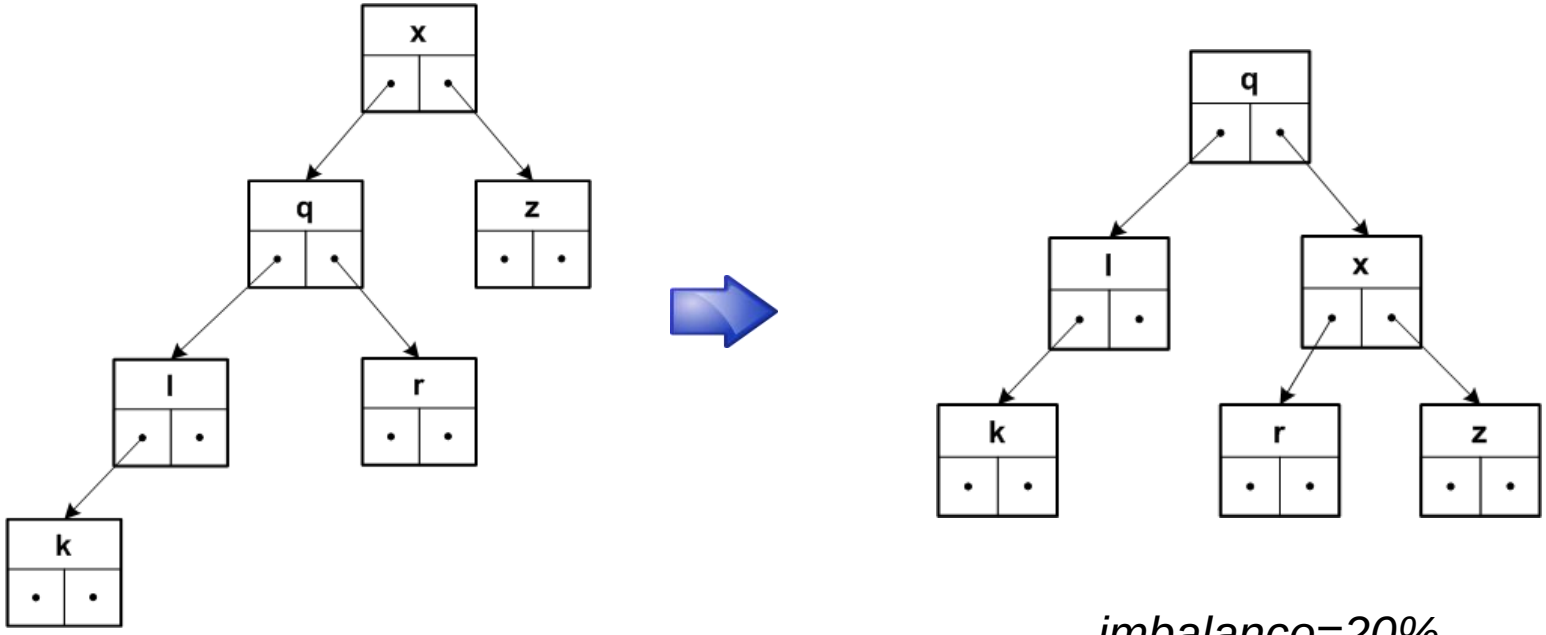




imbalance=60%

Promotie node 'x' tot root

Balanceren stap 3



imbalance=20%

Promotie node 'q' tot root

Java applets

- Kan je toevoegen aan html-webpagina's

```
<applet code="be.ac.vub.ir.util.JAppletWithPanel"
  archive="../causalLearningWithKde.jar"
  height="360" width="900">
  <param name="Panel"
  value="be.ac.vub.ir.statistics.estimators.KDE2DPanel"
  > </applet>
```

- Steek al je files in een jar-file
  - In Eclipse: File menu -> export -> Java -> Jar file
- Je browser heeft een java-console waarin de System output komt (System.out & System.err)

JApplet

- De browser communiceert met de applet via
  - **`public void init( )`** initializes variables and objects
    - Voeg GUI componenten toe aan contentpane:

      `getContentPane().add()`
  - **`public void paint( Graphics g )`** to draw screen
  - **`public void start()`** called when the applet is visible.
    - It is called after the init method and each time the applet is revisited in a Web page.
  - **`public void stop()`** called when applet is no longer visible
  - **`public void destroy()`** when hosting window is closed (exit).
- *Overschrijf methodes als nodig*

Voorbeeld

```
import javax.swing.JApplet;

public class CannonDemoApplet extends JApplet {
     public void init(){
          String opt = getParameter("option");
          if (opt != null && opt.equals("ammo"))
              getContentPane().add(new CannonPanel(true));
          else
              getContentPane().add(new CannonPanel(false));
     }
}
```