

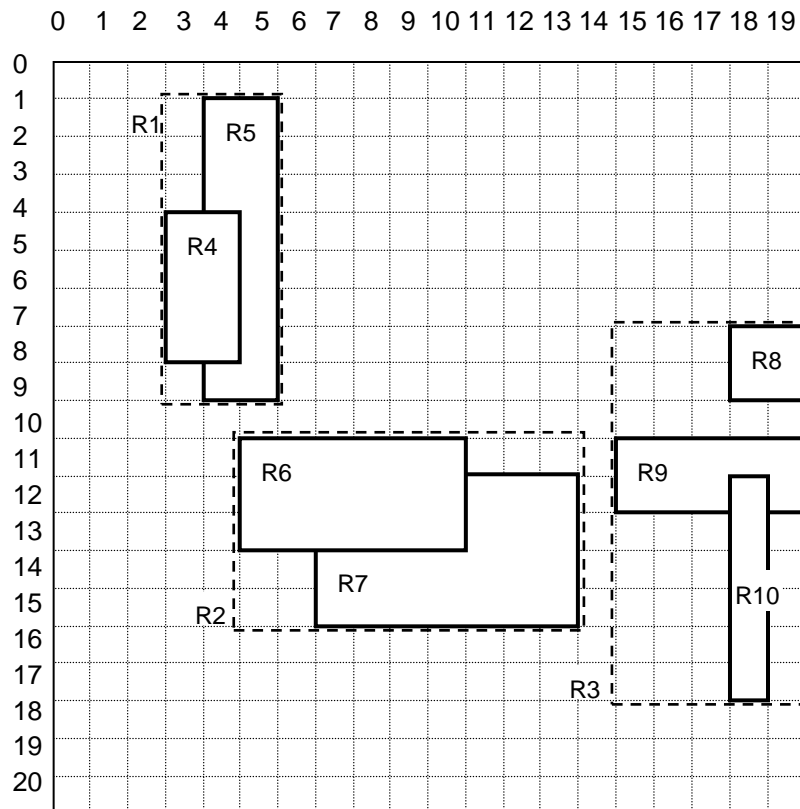
# Speicherstrukturen

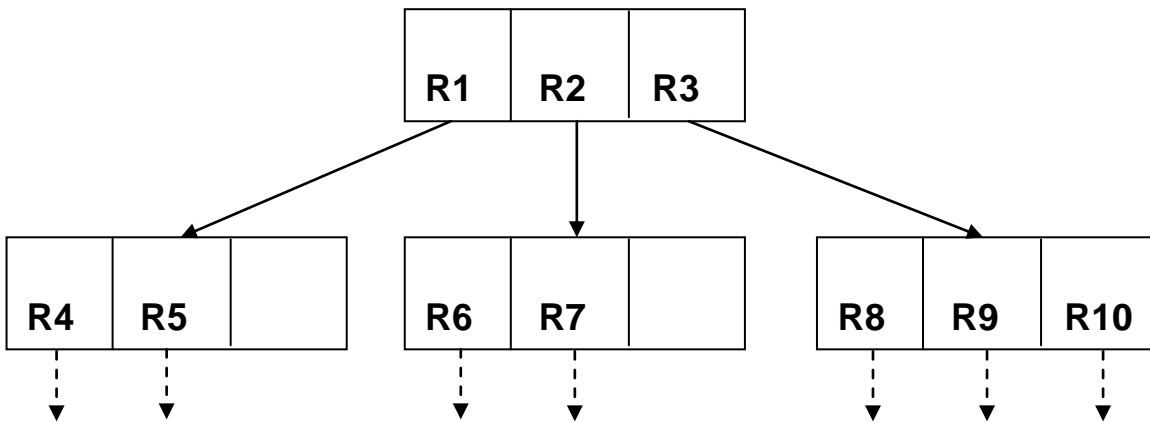
Institut für Datenbanken und Informationssysteme  
 Prof. Dr. M. Reichert, M. Predeschly, J. Kolb

Lösung für Übungsblatt 5

## Aufgabe 5-1: R-Baum

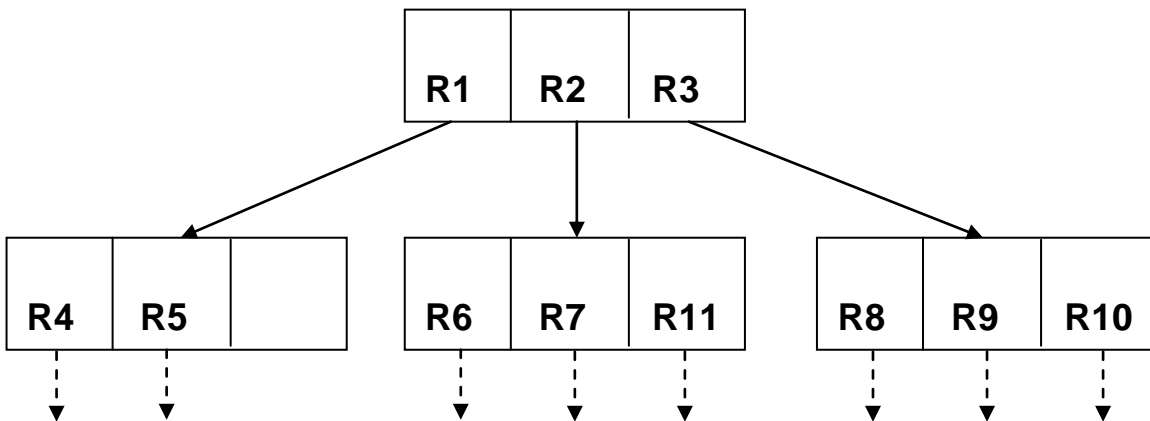
1.

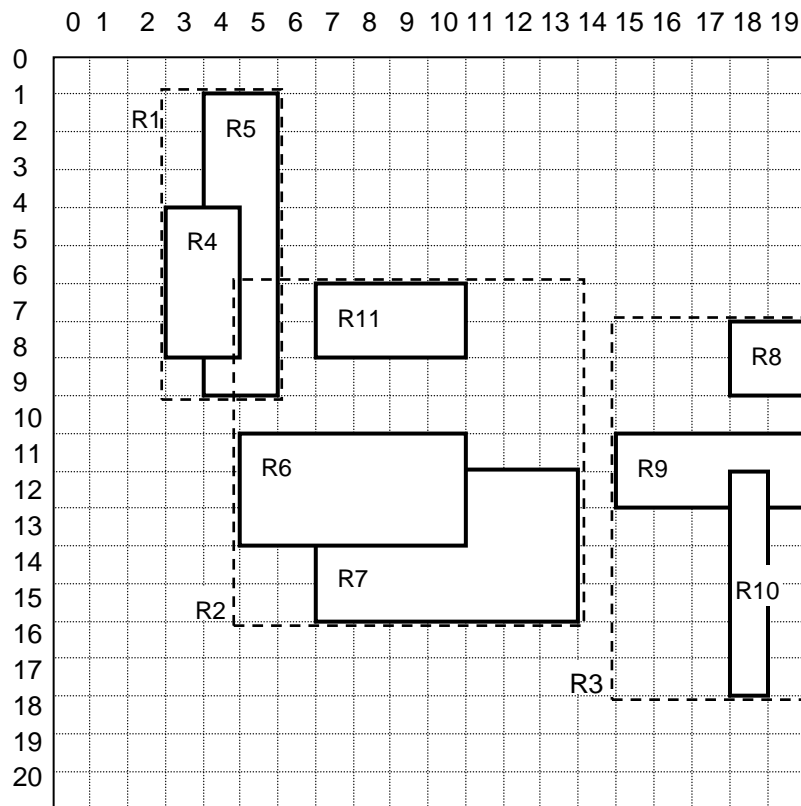




- Einfügen von R11(7, 6, 11, 8)
  1. Test: R11 in R1 → Vergrößerung von R1 auf (3, 1, 11, 9) um  $5 * 8 = 40$  Einheiten
  2. Test: R11 in R2 → Vergrößerung von R2 auf (5, 6, 14, 15) um  $4 * 9 = 36$  Einheiten
  3. Test: R11 in R3 → Vergrößerung von R3 auf (7, 6, 20, 17) um  $143 - 50 = 93$  Einheiten

Einfügen in R2:





- Einfügen von R12(13, 14, 15, 17)
  1. Test: R12 in R1 → Vergrößerung von R1 auf (3, 1, 15, 17) um  $192 - 24 = 168$  Einheiten
  2. Test: R12 in R2 → Vergrößerung von R2 auf (5, 6, 15, 17) um  $110 - 81 = 29$  Einheiten
  3. Test: R12 in R3 → Vergrößerung von R3 auf (13, 7, 20, 17) um  $70 - 50 = 20$  Einheiten

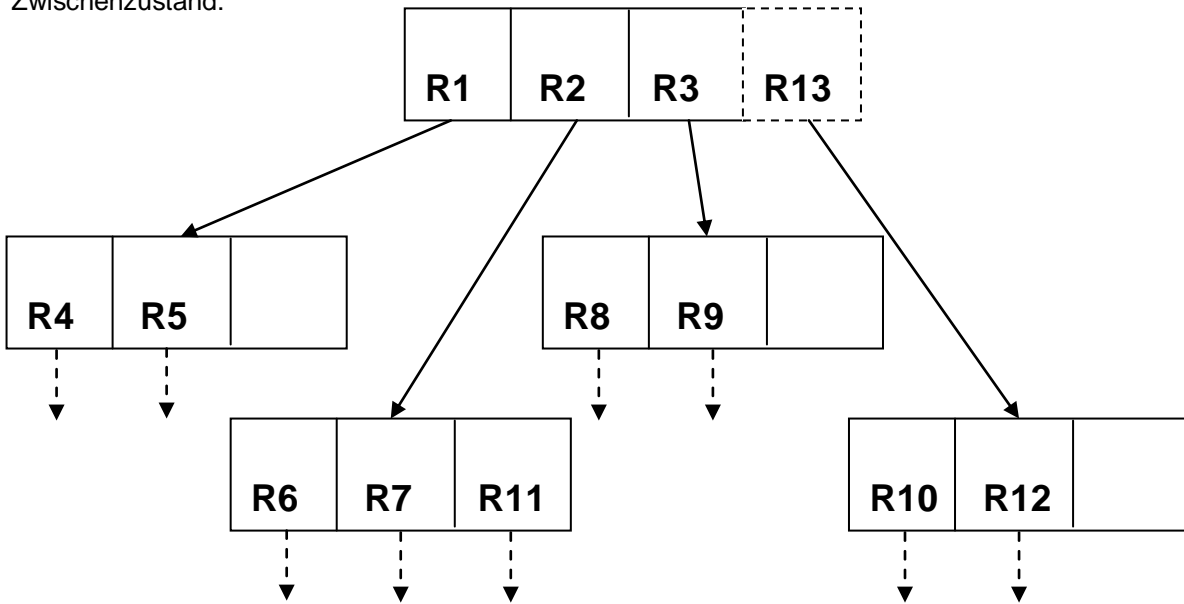
Temporäres Einfügen in R3 → Knoten-Split von (R8; R9; R10; R12) mit „Linear Cost Algorithm“:

	min(max(.))	max(min(.))	Diff.	normalisiert
x	<b>15 (= R12)</b>	<b>18 (= R10)</b>	<b>3</b>	<b>3 : 7 = 0,43</b>
y	<b>9 (= R8)</b>	<b>14 (= R12)</b>	<b>5</b>	<b>5 : 10 = 0,5</b>



Knotensplit: R9 kommt zu R8, R10 kommt zu R12

Zwischenzustand:



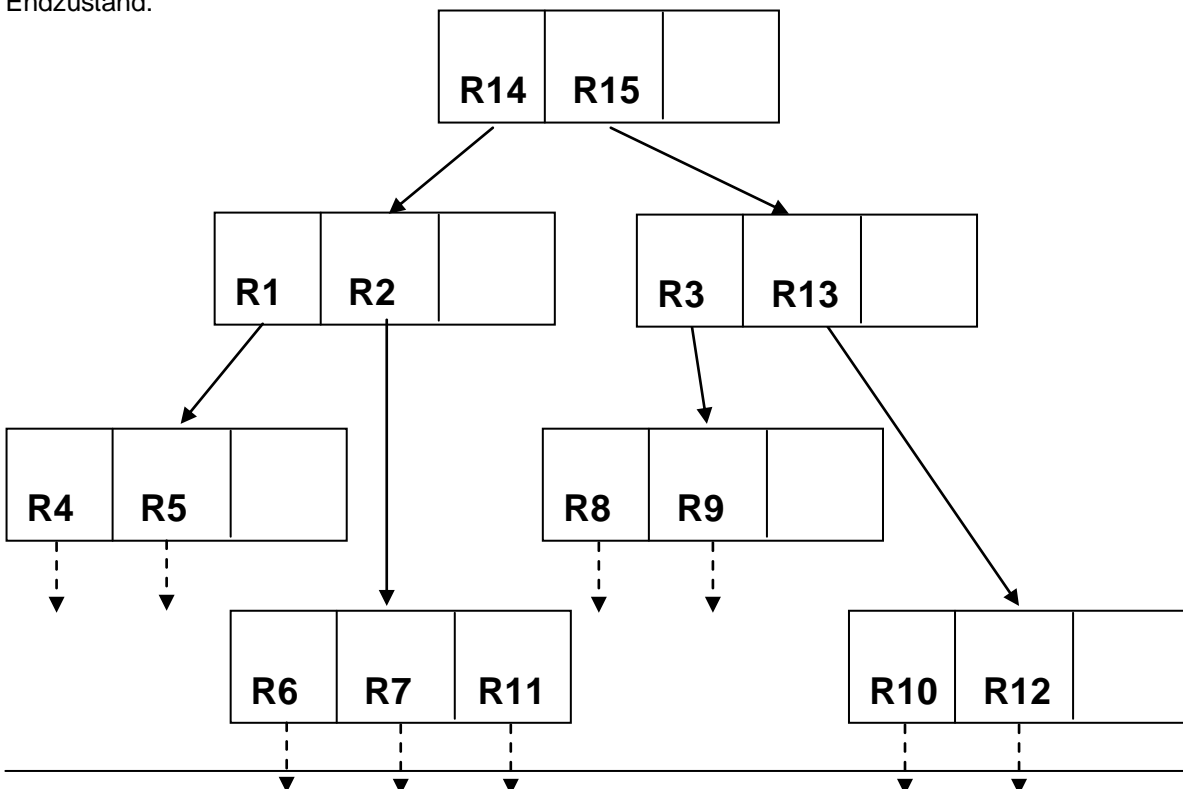
→ Knoten-Split von (R1; R2; R3; R13) mit „Linear Cost Algorithm“:

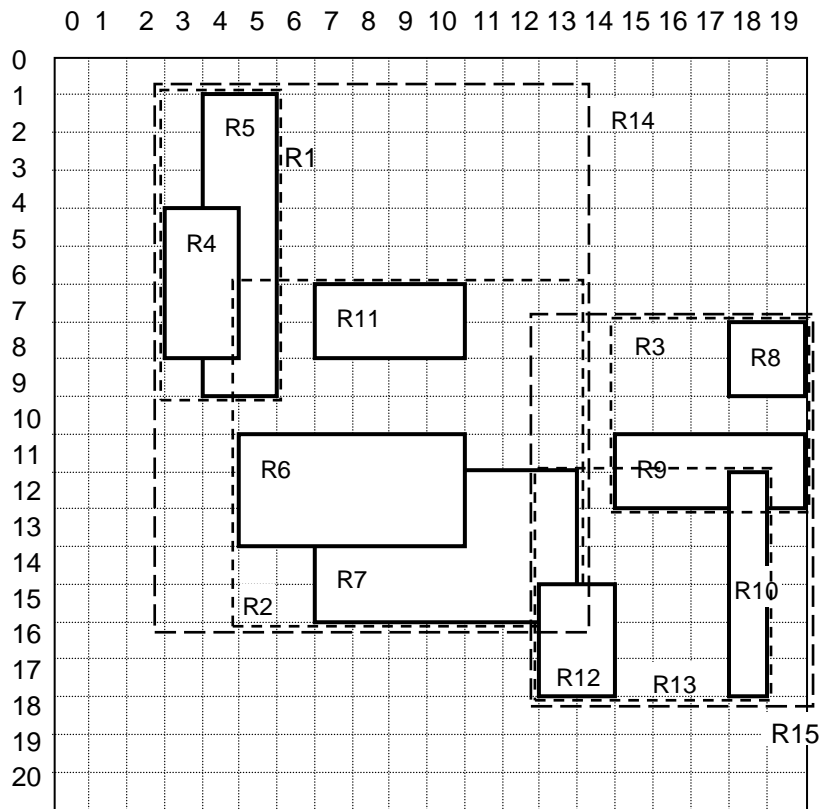
	min(max(.))	max(min(.))	Diff.	normalisiert
x	6 (= R1)	15 (= R3)	9	9 : 17 = 0,53
y	9 (= R1)	11 (= R13)	2	2 : 16 = 0,125



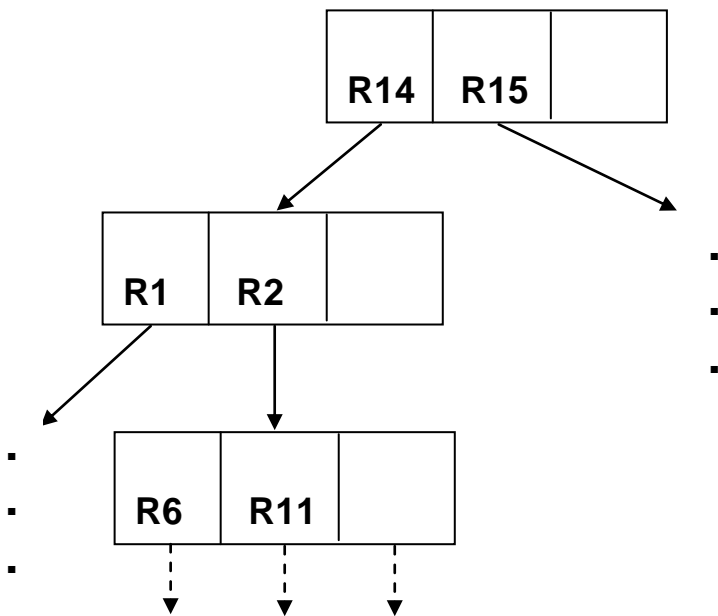
Knotensplit: R2 kommt zu R1, R13 kommt zu R3

Endzustand:





2. Löschen von R7



## Aufgabe 5-2: Konvexe Polygone

```
package CP;

import java.io.*;
import java.awt.Polygon;

public class ConvexPolygon extends java.awt.Polygon
{

    public ConvexPolygon(int xpoints[], int ypoints[], int npoints)
    {
        super(xpoints, ypoints, npoints);
    }

    // Algorithmus Clockwise aus Skript: Die drei Punkte werden durch
    // den Index des mittleren Punkts im Punktearray des Polygons
    // identifiziert!
    // Funktioniert so nur fuer die Ueberpruefung dauf Konvexität

    boolean clockwise(int startIndex)
    {
        int sIMinusOne = (startIndex > 0) ? (startIndex - 1) : (npoints - 1);
        int sIPlusOne = (startIndex < (npoints-1)) ? (startIndex + 1) : 0;

        int dx1, dx2, dy1, dy2;

        dx1 = xpoints[startIndex] - xpoints[sIMinusOne];
        dy1 = ypoints[startIndex] - ypoints[sIMinusOne];

        dx2 = xpoints[sIPlusOne] - xpoints[startIndex];
        dy2 = ypoints[sIPlusOne] - ypoints[startIndex];

        if ((dy1 * dx2) >= (dy2 * dx1)) return true;
        else return false;
    }

    // Pruefung, ob das Polygon konvex ist:
    // Falls clockwise fuer irgendwelche drei
    // aufeinanderfolgenden Punkte false liefert,
    // so ist das Polygon NICHT konvex!

    boolean checkConvexProperty()
    {
        int i;
        boolean convex = true;

        for (i = 0; i < npoints; i++)
        {
            if (! (clockwise(i))) convex = false;
        }

        return convex;
    }
}
```

```

// Pruefung, ob das Polygon cp im aktuellen
// enthalten ist:
// Fuer jeden Punkt des Polygons cp wird geprueft,
// ob im aktuellen Polygon enthalten ist.

boolean contains(ConvexPolygon cp)
{
    int i;

    boolean cont = true;

    for (i = 0; i < cp.npoints; i++)
    {
        if (!(this.contains(cp.xpoints[i], cp.ypoints[i])))
            cont = false;
    }

    return cont;
}

//Main-Methode

public static void main(String[] args) {

    int cp1nPoints = 5;
    int cp1xPoints[] = {4,1,5,17,11};
    int cp1yPoints[] = {4,12,16,13,4};

    ConvexPolygon cp1 = new ConvexPolygon(cp1xPoints, cp1yPoints,
                                         cp1nPoints);

    int cp2nPoints = 3;
    int cp2xPoints[] = {5,8,10};
    int cp2yPoints[] = {7,13,6};

    ConvexPolygon cp2 = new ConvexPolygon(cp2xPoints, cp2yPoints,
                                         cp2nPoints);

    if (cp1.checkConvexProperty())
        System.out.println("\n Das Polygon CP1 ist convex!");
    else
        System.out.println("\n Das Polygon CP1 ist NICHT convex!");

    if (cp2.checkConvexProperty())
        System.out.println("\n Das Polygon CP2 ist convex!");
    else
        System.out.println("\n Das Polygon CP2 ist NICHT convex!");

    if (cp1.contains(cp2))
        System.out.println("\n Das Polygon CP2 ist in CP1 enthalten!");

    if (cp2.contains(cp1))
        System.out.println("\n Das Polygon CP1 ist in CP2 enthalten!");

}
}

```