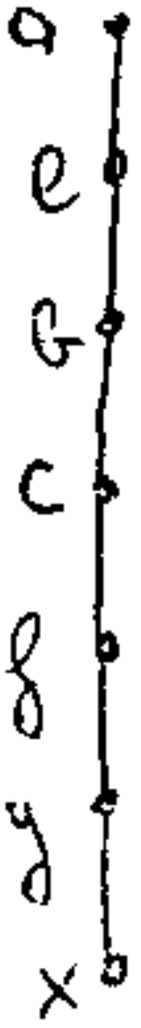
a) BFS:

DFS:



b) 01 Führe BFS durch

02 a ← true

03 For jeden Knoten v ~~[illegible]~~ do

04 a ← (Farbe(v) == schwarz) ~~[illegible]~~ and a

05 If a and (|V| = |E|+1)

06 then entscheidung = true

07 else entscheidung = false

=> alle Knoten [illegible]

=> mehr als einen Kreis

na ja

? P

c) Kosten von BFS: ~~ich glau~~ $O(|V|+|E|)$

Kosten von Zeile 02: $O(1)$

Kosten von Zeile 03/04: $O(|V|)$

~~—~~ " — 06/07: $O(1)$

? P

ingesamt: Kosten von BFS = $O(|V|+|E|)$

✓

```
package GeomShape;
public class Rectangle
{
    private double x1;      //x-Koordinate für linke Seite (kleiner)
    private double x2;      //  ---"---  rechte Seite (größer)
    private double y1;      //y-Koordinate für untere Seite (kleiner)
    private double y2;      //  ---"---  obere Seite (größer)

Rectangle ( )               //default-Konstruktor, Quadrat
  {
      x1 = y1 = 0.0;
      x2 = y2 = 1.0;
  }
Rectangle (double a, double b, double c, double d) //selbst-definierter Konstruktor
  { x1 = a;
    x2 = b;
    y1 = c;
    y2 = d;
  }
public void normalPos ( )   // Rechteck wird so definiert, daß eine Kante
                            // Kantenlänge jeweils links/rechts bzw. x
  {
    int dx = x2 - x1;       // Kantenlänge waagerecht
    int dy = y2 - y1;       // Kantenlänge senkrecht
    x1 = x1 - dx/2;
    x2 = x2 - dx/2;
```

```
    y1 = y1 + dy/2;
    y2 = y2 + dy/2;
}
public boolean isContained (Rectangle r) // Die niedrigeren Werte (x1,y1) von r
{                                        // müssen größer als die des Rechtecks sein, die anderen (x2,y2) kleiner
    return ((x1 < r.x1) && (x2 > r.x2) && (y1 < r.y1) && (y2 > r.y2));
}
public static void main (String[] args) // Test
{
  Rectangle recht1 = new Rectangle (-10,21,-32,50);
  Rectangle recht2 = new Rectangle();
  Rectangle recht3 = new Rectangle (1.2,3.9,-2.0,1.1);
  Rectangle recht2 = normalPos.recht3;
  boolean b1 = recht1.isContained(recht2);
  System.out.println (recht3.x1 + recht3.x2 + recht3.y1 + recht3.y2 + b1);
}
}
```