Week 3 ALG2

Sander van Leeuwen

Rick Veens

public class Opdracht

{

private SpelObject[] objArray;

public Opdracht()

{

objArray = new SpelObject[8];

objArray[0] = new SpelObject(900, 100);

objArray[1] = new SpelObject(100, 100);

objArray[2] = new SpelObject(50, 750);

objArray[3] = new SpelObject(110, 90);

objArray[4] = new SpelObject(950, 50);

objArray[5] = new SpelObject(60, 800);

objArray[6] = new SpelObject(40, 800);

objArray[7] = new SpelObject(700, 850);

// root node, parent is null

Node rootNode;

rootNode = sortArray(0, 7, 0, null);

printNodes(rootNode);

}

private void printNodes(Node node) {

if (node != null) {

if (node instanceof SplitNode) {

printNodes(((SplitNode)node).getLinkerKind());

printNodes(((SplitNode)node).getRechterKind());

} else

((EndNode)node).printSpelObject();

}

}

private void swap(int first, int second)

{

SpelObject temp = objArray[first];

objArray[first] = objArray[second];

objArray[second] = temp;

}

public Node sortArray(int left, int right, int dimensionIndex, Node parentNode)

{

Node node = null; // dit wordt onze return value

if (right - left <= 0) // array is 1 groot

node = new EndNode(parentNode, objArray[left]); // endnode, deze gaat op return en we gaan weer terug

else if ( (right - left) == 1) // array is 2 groot

{ // ook nog splitnode

if (objArray[right].getPosition(dimensionIndex) < objArray[left].getPosition(dimensionIndex))

swap(left, right);

node = new SplitNode(parentNode);

EndNode leftNode = new EndNode(node, objArray[left]);

EndNode rightNode = new EndNode(node, objArray[right]);

((SplitNode)node).setLinkerKind(leftNode);

((SplitNode)node).setRechterKind(rightNode);

}

else // splitnode

{

int mediaanIndex = partition(left, right, dimensionIndex);

// dimension een omhoog (of weer 0)

if (dimensionIndex < SpelObject.DIMENSION -1)

++dimensionIndex;

else

dimensionIndex = 0;

node = new SplitNode(parentNode); // node is nieuwe parent

// linkerhelft: left ... <= mediaan

((SplitNode)node).setLinkerKind(sortArray(left, mediaanIndex, dimensionIndex, node));

// rechterhelft: mediaan >= ...

((SplitNode)node).setRechterKind(sortArray(mediaanIndex+1, right, dimensionIndex, node));

}

return node; // node komt terug bij de vorige en wordt ingesteld als linker of rechter kind, of hij komt weer terug bij de eerste call

}

private int partition(int initialLeft, int initialRight, int dimensionIndex)

{

//Bepalen van de mediaan, door middel van de mediaan van 3

int left = initialLeft;

int right = initialRight;

int middle = (left+right) / 2;

if(objArray[left].getPosition(dimensionIndex) > objArray[middle].getPosition(dimensionIndex))

swap(left, middle);

if(objArray[middle].getPosition(dimensionIndex) > objArray[right].getPosition(dimensionIndex))

swap(middle, right);

if(objArray[left].getPosition(dimensionIndex) > objArray[middle].getPosition(dimensionIndex))

swap(left, middle);

swap(right-1, middle);

double mediaanWaarde = objArray[right-1].getPosition(dimensionIndex);

left++;

right -= 2;

//Einde bepalen mediaan van 3

while (left <= right)

{

// ga door totdat je een waarde vind die hoger is dan de mediaan waarde

while (objArray[left].getPosition(dimensionIndex) < mediaanWaarde)

left++;

// ga door totdat je een waarde vind die lager is dan de mediaan waarde

while (mediaanWaarde < objArray[right].getPosition(dimensionIndex))

right--;

// zolang i en j elkaar niet kruisen,

if (left <= right) {

swap(left, right);

++left;

--right;

}

}

swap(left, initialRight-1);

return left;

}

}