Entrega 2: Proyecto de colisión de abejas

Isaias Labrador Sánchez
Universidad Eafit
Colombia
ilabradors@eafit.edu.co

Manuel García Jiménez
Universidad Eafit
Colombia
megarciaj@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

A: PALABRAS CLAVES DE AUTOR

Quadtree, colisión de abejas, estructuras de datos.

B: PALABRAS CLAVES DE LA ACM

Theory of computation → Design and analysis of algorithms→ Data structures design and analysis→Sorting and searching

D: GRAFICA

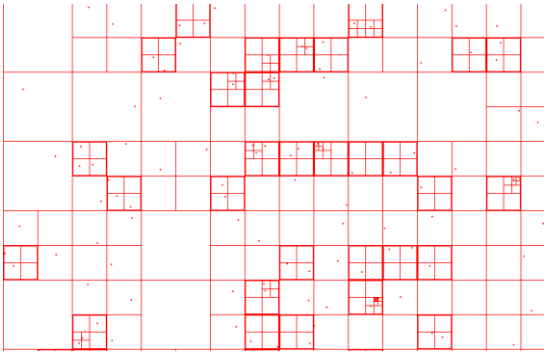

Figura: Quadtree, cada cuadro es un nodo que puede o no contener abejas, cada punto es una abeja.

E: EXPLICACIÓN DE CADA OPERACIÓN:

Cuadrado (Constructor):
Recibe una coordenada x y otra y, además de un largo ancho. Se usa más que todo para la creación de nodos del árbol, pues cada vez que entra una nueva abeja a la estructura de datos se crea un nuevo "cuadtree" a partir del rectángulo que representa el área que era originalmente de la única abeja que lo habitaba.

Quadtree (constructor):
Crea un Quadtree. También puede ser utilizado para crear un nodo del árbol original.

Quadtree.show:
Recorre el árbol hacia abajo metiéndose por todas las ramas.

Drawing (Constructor):
Crea ArrayLists de posiciones de abejas.

Quadtree.insert:
Mete a la abeja en la posición correcta del árbol, asegurándose de crear todos los nodos necesarios para hacerlo.

createBees:
Crea un ArrayList de abejas a partir de ArrayLists con sus posiciones en x y en y.

Abeja (Constructor):
Crea una abeja teniendo en cuenta que esta tiene una forma circular.

Circulo (Constructor):
Crea un circulo con centro y radio dados como parámetros.

Quadtree.query:
Crea un ArrayList que contiene a todas las abejas que están en la estructura de datos.

Drawing.enviar:
Genera la gráfica del quadtree. Grafica los rectángulos que representan los nodos del árbol de blanco y a las abejas de verde.

F: CALCULO DE COMPLEJIDADES:

Complejidades:	
Cuadrado	O(1)
Quadtree	O(1)
Quadtree.show()	O(log(n))
Quadtree.insert()	O(log(n))
Quadtree.createBees()	O(n)
Circulo	O(1)
Abeja	O(1)
Quadtree.query()	O(n)
Drawing.enviar()	O(n)

G: EXPLICACIÓN ESCONJENCIA DE ESTRUCTURA DE DATOS:

Esta estructura de datos fue escogida porque permite interpretar las posiciones de objetos como coordenadas de dos dimensiones, lo cual lo hace apropiado para este tipo de problemas donde las posiciones en un plano de cosas son relevantes.

También fue escogida porque al ubicar las hojas según posiciones, estas quedan ubicadas de manera que, si están cerca geométricamente, es muy probable que estén en la misma rama. Esto hace que buscar abejas cercanas a la abeja actual sea O(log(n)).

K: JAVADOC:

file:///D:/Manuel/Documents/semestre%203/Datos%20y%20algoritmos%201/proyecto%20final/Proyecto_Datos/Proyecto_Datos/dist/javadoc/allclasses-frame.html

IMPORTANTE: CITAS:

El código utilizado para esta solución del problema está basado en el código y seudocódigo expuesto en las siguientes fuentes:

Shiffman(2018), Quadtree (Version 1.0), [source code], https://github.com/CodingTrain/website/tree/master/CodingChallenges/CC_98_1_QuadTree

Wikipedia (2018), Pseudo code (Version 1.0) [Pseudo code], https://en.wikipedia.org/wiki/Quadtree#Pseudo_code