

ALGORITMO PARA PREVENIR COLICIONES ENTRE ABEJAS ROBOTICAS

*Juan Camilo Guerrero Alarcon
Santiago Pulgarin Vasquez
Medellín, 6 de Noviembre de 2018*

QuadTree

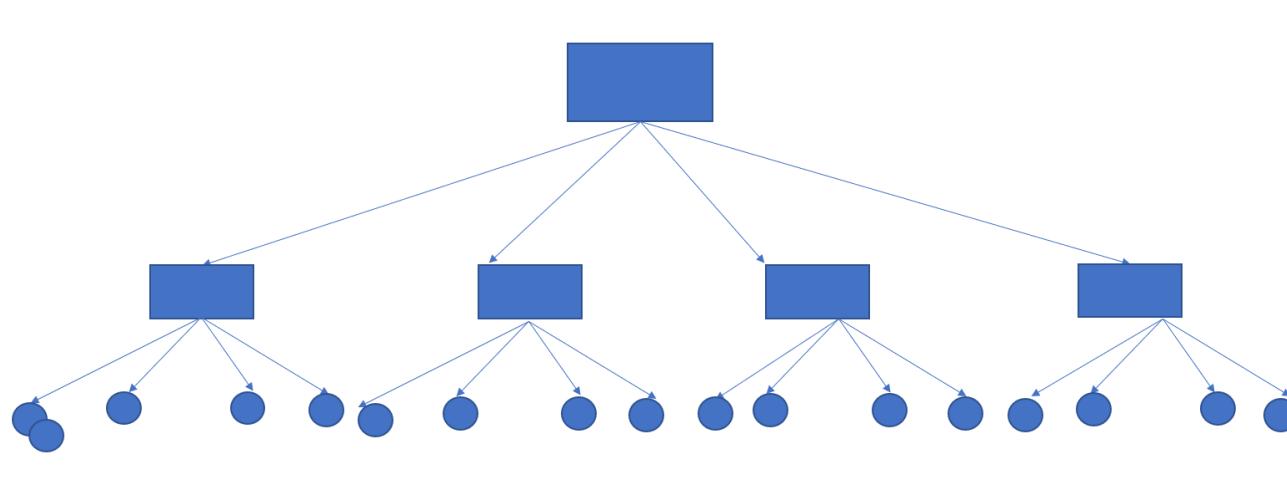


Gráfico 1: QuadTree de abejas. Una abeja es una clase que contiene coordenada “x” y coordenada “y”

Operaciones de la Estructura de Datos

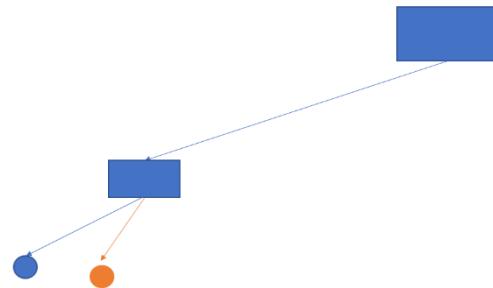


Gráfico 2: Operación de insertado
de una estructura de datos

Clase	Complejidad
Abeja	$O(1)$
AgregarAbejas	$O(n)$
Boundary	$O(1)$
Main	$O(1)$
QuadTree	$O(n)$

Tabla 1: Complejidad de las operaciones
de la estructura de datos

Criterios de Diseño de la Estructura de Datos

- La razón por la cual escogimos la estructura de datos QuadTree es debido a que es una de las más usadas cuando se tratan problemas de ubicación espacial, más conocidas y muy acertada para tipos de problemas como este, además de su excelente implementación que facilita el análisis de los datos.

Consumo de Tiempo y Memoria

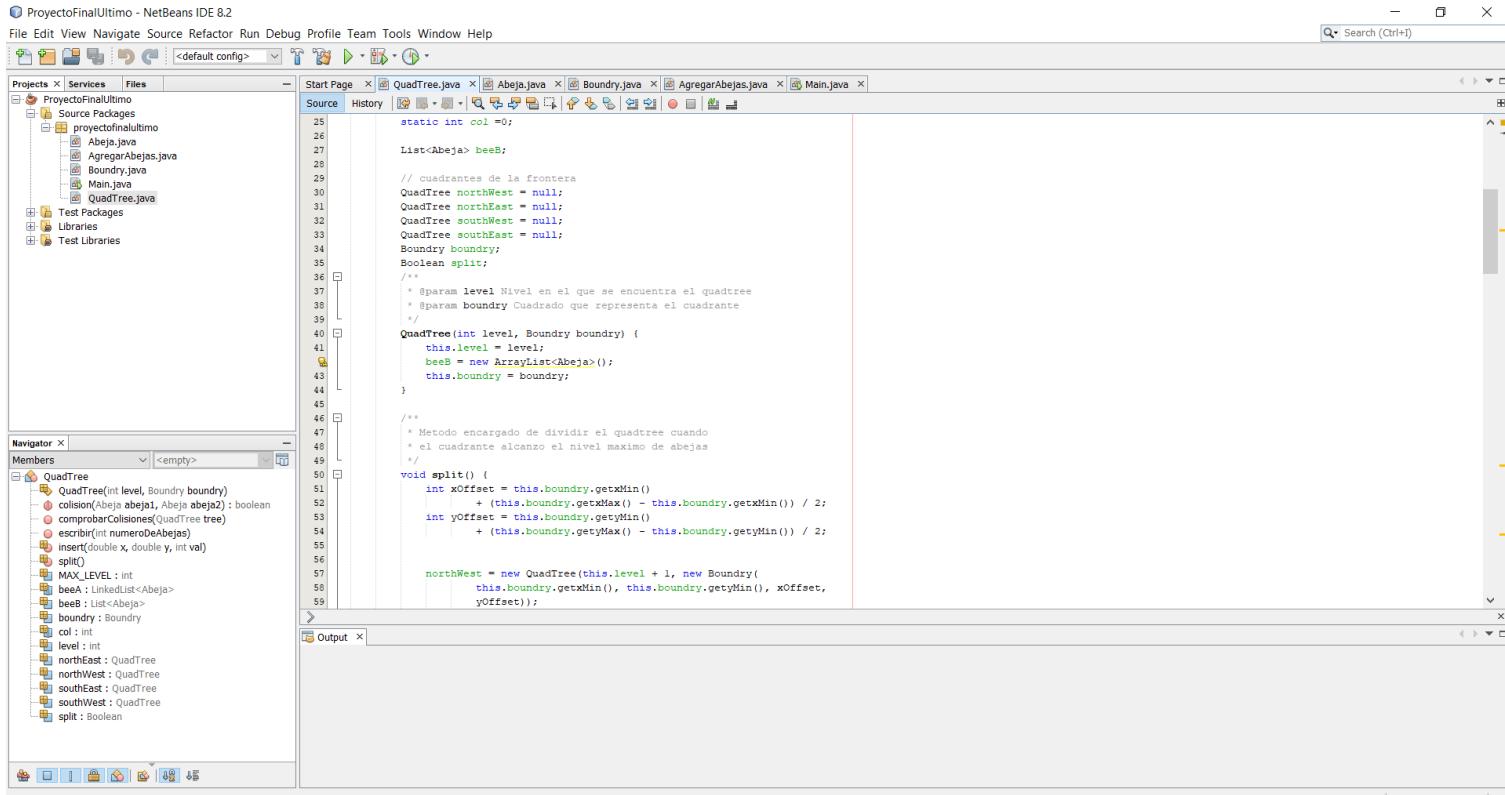
	CONJUNTO DE DATOS 1 (10)	CONJUNTO DE DATOS 2 (100)	CONJUNTO DE DATOS 3 (1000)	CONJUNTO DE DATOS 4 (100000)	CONJUNTO DE DATOS 5 (1000000)
TIEMPO QUE TARDÓ	0,001 ms	0,8 ms	9 ms	100 ms	10000 ms
Consumo de memoria	90 MB	90 MB	90 MB	120 MB	450 MB

Tabla 2: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

	Conjunto de datos1 (10)	CONJUNTO DE DATOS 2 (100)	CONJUNTO DE DATOS 3 (1000)	CONJUNTO DE DATOS 4 (100000)	CONJUNTO DE DATOS 5 (1000000)
Consumo de memoria	90 MB	90 MB	90 MB	120 MB	450 MB
Consumo de memoria	90 MB	90 MB	90 MB	120 MB	450 MB

Tabla 3: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

Software Desarrollado



The screenshot shows the NetBeans IDE 8.2 interface with the following details:

- Project:** ProyectoFinalUltimo
- Source Packages:** proyectofinalultimo (containing Abeja.java, AgregarAbejas.java, Boundary.java, Main.java, and QuadTree.java).
- Source Editor:** The main window displays the `QuadTree.java` file. The code implements a QuadTree data structure for managing bees. It includes methods for collision detection, inserting bees, and splitting quadrants. The code uses Java annotations like `@Override` and `@NotNull`.
- Navigator:** Shows the class hierarchy and member variables for `QuadTree`.
- Output:** An empty output pane.

```
static int col =0;
List<Abeja> beeB;
// cuadrantes de la frontera
QuadTree northWest = null;
QuadTree northEast = null;
QuadTree southWest = null;
QuadTree southEast = null;
Boundary boundary;
Boolean split;
/**
 * Guarda level Nivel en el que se encuentra el quadtree
 * Guarda boundary Cuadrado que representa el cuadrante
 */
QuadTree(int level, Boundary boundary) {
    this.level = level;
    beeB = new ArrayList<Abeja>();
    this.boundary = boundary;
}

/**
 * Metodo encargado de dividir el quadtree cuando
 * el cuadrante alcanzo el nivel maximo de abejas
 */
void split() {
    int xOffset = this.boundary.getMinX()
        + (this.boundary.getMaxX() - this.boundary.getMinX()) / 2;
    int yOffset = this.boundary.getMinY()
        + (this.boundary.getMaxY() - this.boundary.getMinY()) / 2;

    northWest = new QuadTree(this.level + 1, new Boundary(
        this.boundary.getMinX(), this.boundary.getMinY(), xOffset,
        yOffset));
}
```

Gráfico 4: Estructura de datos QuadTree

Inspira Crea Transforma



Matriz en tres dimensiones

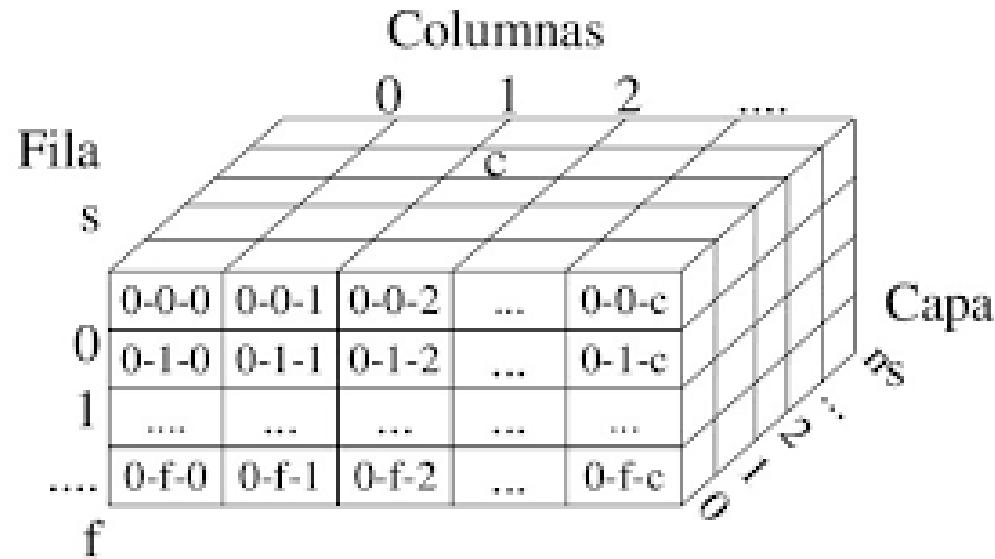


Gráfico 1: Matriz 3D de abejas. Una abeja es una clase que contiene coordenada “x”, coordenada “y” y coordenada z

Operaciones de la Estructura de Datos

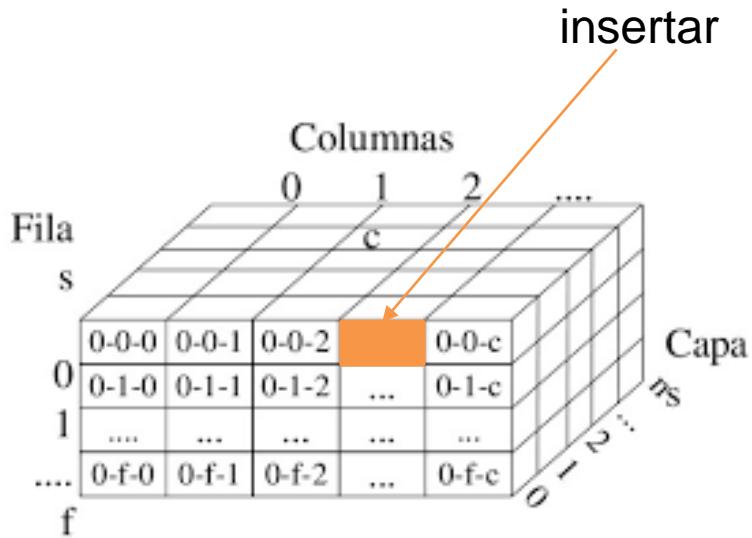


Gráfico 2: Operación de insertado de una estructura de datos

Clase	Complejidad
Detectar Colisiones	$O(n \log (n))$

Tabla 1: Complejidad de las operaciones de la estructura de datos

Criterios de Diseño de la Estructura de Datos

- La razón por la cual escogimos la estructura de datos Matriz tridimensional es debido a que es una de las más usadas cuando se tratan problemas de ubicación espacial, más conocidas y muy acertada para tipos de problemas como este, además de su excelente implementación que facilita el análisis de los datos.
- Una de las principales razones por la que decidimos implementar esta estructura de datos es por su facil acceso O(1)

Consumo de Tiempo y Memoria

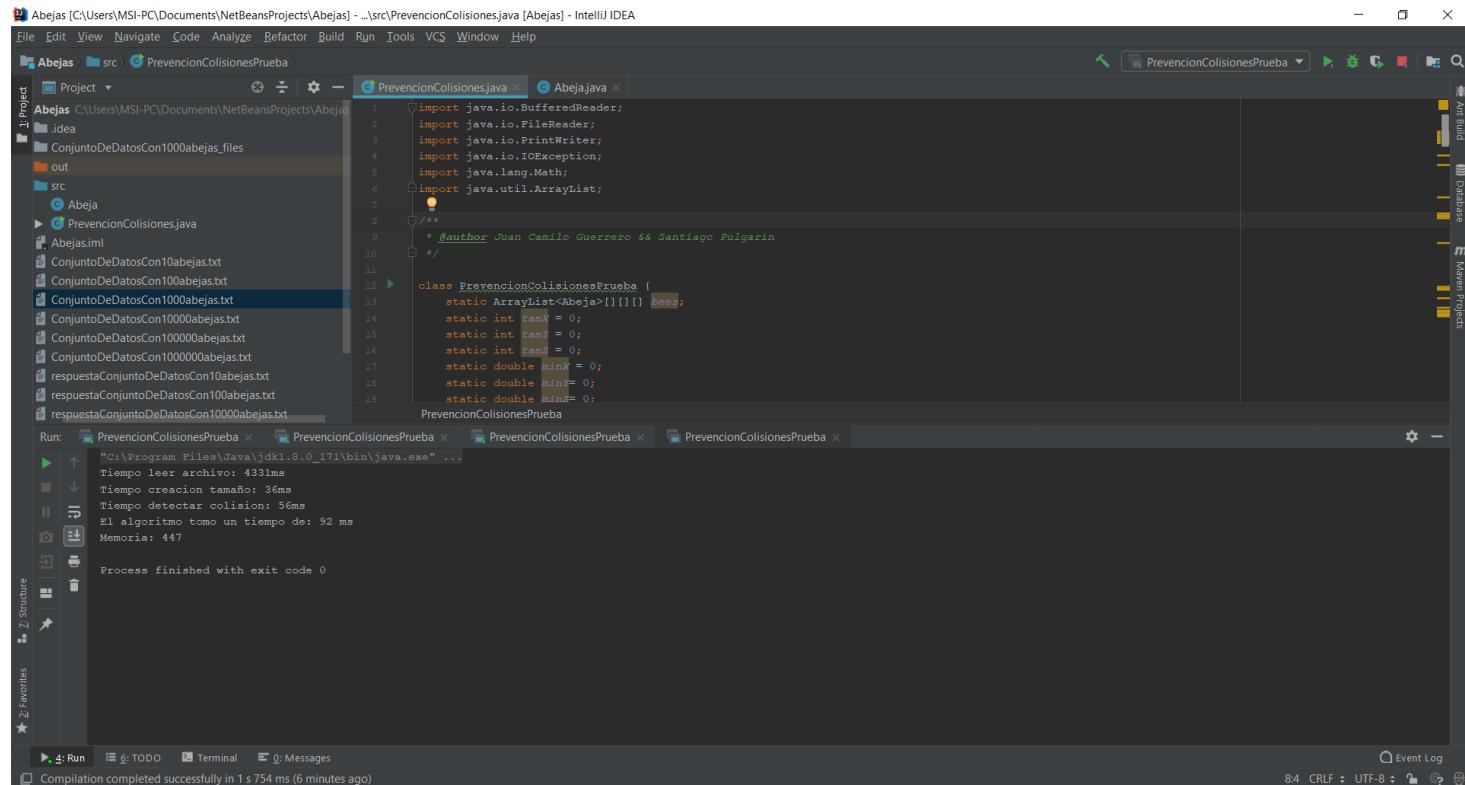
	CONJUNTO DE DATOS 1 (10)	CONJUNTO DE DATOS 2 (100)	CONJUNTO DE DATOS 3 (1000)	CONJUNTO DE DATOS 4 (10000)	CONJUNTO DE DATOS (100000)	DE 5	CONJUNTO DE DATOS 6 (1000000)
TIEMPO QUE TARDO	0 ms	20 ms	52 ms	78 ms	436 ms		756 ms

Tabla 2: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

	Conjunto de datos1 (10)	CONJUNTO DE DATOS 2 (100)	CONJUNTO DE DATOS 3 (1000)	CONJUNTO DE DATOS 4 (10000)	CONJUNTO DE DATOS 5 (100000)	CONJUNTO DE DATOS 6 (1000000)
Consumo de memoria	123 MB	123 MB	123 MB	123 MB	155 MB	500 MB

Tabla 3: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

Software Desarrollado



The screenshot shows the IntelliJ IDEA interface with the project 'Abejas' open. The code editor displays 'PrevencionColisiones.java' containing Java code for collision detection between bees. The code includes imports for BufferedReader, FileReader, PrintWriter, IOException, and ArrayList, along with class definitions and static variables. The run tab shows the execution results of the program, which includes timing metrics for reading files, creating the tree, detecting collisions, and the total algorithm time.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.PrintWriter;
import java.io.IOException;
import java.lang.Math;
import java.util.ArrayList;

/*
 * @author Juan Camilo Guerrero & Santiago Pulgarin
 */

class PrevencionColisionesPrueba {
    static ArrayList<Abeja>[][][] bees;
    static int tamX = 0;
    static int tamY = 0;
    static int tamZ = 0;
    static double minX = 0;
    static double minY = 0;
    static double minZ = 0;

    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader("ConjuntoDeDatosCon10abejas.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                String[] coordinates = line.split(",");
                double x = Double.parseDouble(coordinates[0]);
                double y = Double.parseDouble(coordinates[1]);
                double z = Double.parseDouble(coordinates[2]);
                bees[tamX][tamY][tamZ] = new Abeja(x, y, z);
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void detectCollision() {
        // Implementation of collision detection logic
    }

    public static void printResults() {
        // Implementation of result printing logic
    }
}
```

Gráfico 4: Estructura de datos QuadTree