

ALGORITMO PARA PREVENIR COLICIONES ENTRE ABEJAS ROBOTICAS

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

Estructuras de Datos Diseñada

Tamaño Del Arreglo = 6



Abeja 1 + Longitud + Latitud + Altitud	Abeja 2 + Longitud + Latitud + Altitud	Abeja 3 + Longitud + Latitud + Altitud	Abeja 4 + Longitud + Latitud + Altitud	Abeja 5 + Longitud + Latitud + Altitud	Abeja 6 + Longitud + Latitud + Altitud
---	---	---	---	---	---

0

1

2

3

4

5

contiene

Operaciones de la Estructura de Datos



Gráfica 2: Imagen de una operación Add de un ArrayList.

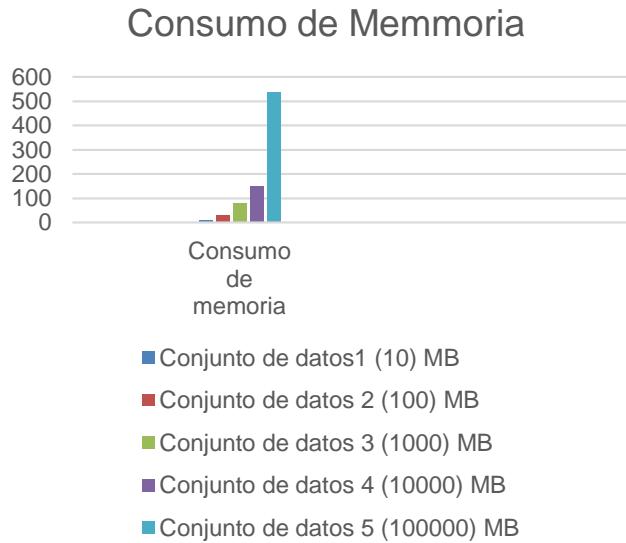
Tabla 1: Tabla para reportar la complejidad

Metodo	Complejidad
distancia()	$O(n)$
leerArchivo	$O(n)$
detectarColisiones()	$O(n^2)$
guardarArchivo()	$O(n)$
main()	$O(1)$

Criterios de Diseño de la Estructura de Datos

La razón por la cual escogimos la estructura de datos ArrayList es debido a que es una de las más usadas, más conocidas y fácil de implementar a tipos de problemas como este, además de su excelente implementación que facilita el análisis de los datos, aunque este tipo de estructura no es muy eficiente y nos genera un déficit en memoria y en tiempo de procesamiento lo que genera poca efectividad para grandes cantidades de datos.

Consumo de Tiempo y Memoria



	Mejor Tiempo (segundos)	Peor Tiempo (segundos)	Tiempo Promedio (segundos)	Mejor Memoria (MB)	Peor Memoria (MB)	Memoria Promedio (MB)
Conjunto de datos1 (10)	0	0	0	7	11	9
Conjunto de datos 2 (100)	0,001	0,003	0,001683168	20	40	30
Conjunto de datos 3 (1000)	0,011	0,015	0,013138614	60	100	80
Conjunto de datos 4 (10000)	0,319	0,39	0,327653465	100	200	150
Conjunto de datos 5 (100000)	18,46	20,786	19,54	400	680	540



Inserten sus propias gráficas y sus explicaciones

Software Desarrollado

The screenshot shows a Java code editor with the file 'PrevencionColisiones.java' open. The code implements an algorithm to detect collisions between bees based on their coordinates. It includes methods for reading input from a file, detecting collisions, and saving the results to a file.

```
Start Page X PrevencionColisiones.java X
Source History | 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
arregloDeAbejas[index+j] = abeja; //Constante
lineaActual = br.readLine(); //Constante
}
catch(IOException ioe) {
    System.out.println("Error leyendo el archivo de entrada"); //Constante
}
return arregloDeAbejas; //Constante
}

/**
 * Algoritmo para prevenir colisiones (genera muchas respuestas repetidas)
 *
 * @param arregloDeAbejas Un arreglo con coordenadas de las abejas
 * @return una lista definida con arreglos con las abejas que tienen riesgo de colision
 */
public static ArrayList<Point3D> detectarColisiones(Point3D[] arregloDeAbejas){
    ArrayList<Point3D> abejasConRiesgoDeColision = new ArrayList(); //Constante
    for (int i = 0; i < arregloDeAbejas.length ; ++i) //Constante + T(n-1)*Constante
        for(int j = i+1; j < arregloDeAbejas.length; ++j) //((Constante + T(n-1)*Constante)*n^2
            if (distancia(arregloDeAbejas[i], arregloDeAbejas[j]) <= 100){ //Constante*n^2
                abejasConRiesgoDeColision.add(arregloDeAbejas[i]); //Constante*n
                abejasConRiesgoDeColision.add(arregloDeAbejas[j]); //Constante*n
            }
    return abejasConRiesgoDeColision; //Constante
}

/**
 * Metodo para escribir un archivo con la respuesta
 *
 * @param abejasConRiesgoDeColision Lista definida con arreglos con las abejas con riesgo de colision
 * @param numeroDeAbejas Numero de abejas del conjunto de datos original
 */
public static void guardarArchivo(ArrayList<Point3D> abejasConRiesgoDeColision, int numeroDeAbejas){
    final String nombreDelArchivo = "C:/Users/Consumidor/Downloads/Univeral/abejas4/abejas.txt"; //Constante
}

Output - EjemploProyectoFinal (run) X
run
El algoritmo tomo un tiempo de: 19116 ms
BUILD SUCCESSFUL (total time: 1 minute 1 second)
```