ALGORITHMIC CONTROL AND SUPERVISION FOR THE PREVENTION OF COLLISION BETWEEN ROBOTIC BEES

Hamilton Smith Gómez Osorio
Universidad EAFIT
Colombia
hsgomezo@eafit.edu.co

Santiago Isaza Cadavid
Universidad EAFIT
Colombia
sisazac@eafit.edu.co

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

ABSTRACT

Robotic bees are the future of agriculture in their process of production, that is why makes necessary having control and supervision of the exact location of these in order to achieve an optimal behavior.

KEYWORDS

Sorting and Searching, Data structures, Hashing, Data Management.

ACM CLASSIFICATION System

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

1. INTRODUCTION

Faced with the decline of the population of bees that is happening nowadays and the importance of these in the process of pollination and in the agricultural sector it is possible to say that there is a risk in agricultural crops, so it is necessary to find a solution to this problem. This is how the idea of creating robotic bees was born, which can help in this process and, to supervise them, and control their behavior, develop and implement and algorithm that prevents their collision.

2. PROBLEM

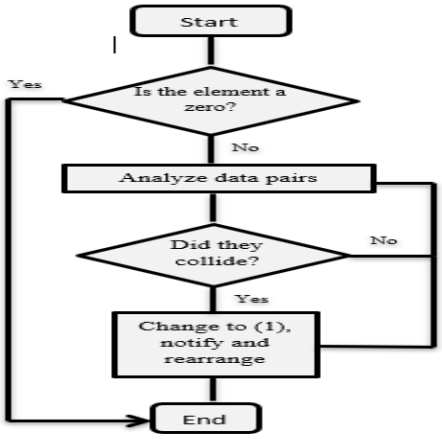
The implemented robotic bees in agriculture for the pollination process can collide if they are less than 100 meters from other bees, that is why is so important solve the problem in order to achieve and optimal behavior and an improvement in their processes.

3. RELATED WORKS

3.1 Radio Frequency Identification System (RFIS)

When there are several labels (used to store the information) and readers (read, change and verify the information on the label) in the same channel and signal transmission, a collision problem occurs due to mutual interference between the labels and the readers.

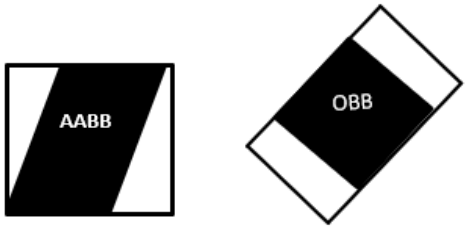
Solution: an anti-collision algorithm based on a matrix and a coding scheme. The decoded data is established in a matrix and then the reader is responsible for processing the data row by row, analyzed in pairs and finding a collision is replaced by a value (1), otherwise set a zero (0). After replacing the rows, the collisions are extracted and the following rows are analyzed until the algorithm finishes.



[1]

3.2 Bounded volumes to detect collisions

Very used in graphic computing when making videogames. It is based on the use of basic geometric shapes bounding more complex figures and using the intersection of these to determine when a collision occurs; thanks to the figures you have control of the objects when there is movement or perspective changes, a solution used in AABB (Axis-aligned bounding box) and OBB (Oriented bounding box).



[2]

3.3 Octree

To analyze collisions between multiple elements in a plane, this method is used, which subdivide a delimited space into rectangles of equal size and then divides the space again until it has a relatively smaller area in which is easier and faster to compare positions of one object and another, considering its volume and coordinates. [3]

3.4 Data structure spatial hashing

It consists of dividing a zone into cubes with a specific measure, considering the maximum and minimum value of coordinates, and then organizing the objects that are inscribed within them in a list of reference to the index of the box in which it is located. [4]

4. Structure to use: Spatial Hashing

4.2 Design criteria for the data structure

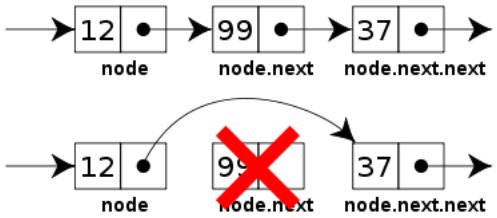
This structure is more efficient compared to the others presented because in the construction of the algorithm we can define the size of each cell so we do not need to analyze every single position and compare to the others (brute force) because the fact that two bees are in the same cell means that there is a collision risk. So, with this structure it is possible to save memory, avoiding any type of classification and query, as well as making the algorithm faster.

Implementation approach

In the Java Programming language, we seek to implement an optimal solution with the Spatial Hashing data structure, which, using cubes with a volume of 10m3, delimiting the territorial extension of the municipality of Bello, inscribing the robotic bees located there, classifying them in a list and then return those bees that are in the same cube and recognize the ones that at risk of collision.

4.1 Operaciones de la estructura de datos

Diseñen las operaciones de la estructura de datos para solucionar el problema eficientemente. Incluyan una imagen explicando cada operación



Gráfica 2: Imagen de una operación de borrado de una lista encadenada

4.3 Análisis de Complejidad

Calculen la complejidad de las operaciones de la estructura de datos para el peor de los casos. Vean un ejemplo para reportarla:

Mètodo	Complejidad
Bùsqueda Fonètica	O(1)
Imprimir bùsqueda fonètica	O(m)
Insertar palabra busqueda fonètica	O(1)
Bùsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir bùsqueda	O(s)

Tabla 1: Tabla para reportar la complejidad

4.4 Tiempos de Ejecución

Calculen, (I) el tiempo de ejecución y (II) la memoria usada para las operaciones de la estructura de datos, para el Conjunto de Datos que está en el ZIP

Tomen 100 veces el tiempo de ejecución y memoria de ejecución, para cada conjunto de datos y para cada operación de la estructura de datos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Creación	10 sg	20 sg	5 sg
Operación 1	12 sg	10 sg	35 sg
Operación 2	15 sg	21 sg	35 sg
Operación n	12 sg	24 sg	35 sg

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

4.5 Memoria

Mencionar la memoria que consume el programa para los conjuntos de datos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Consumo de memoria	10 MB	20 MB	5 MB

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

4.6 Análisis de los resultados

Expliquen los resultados obtenidos. Hagan una gráfica con los datos obtenidos, como por ejemplo:

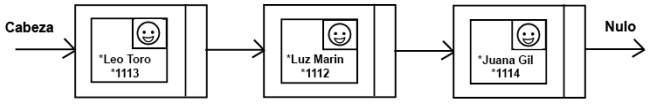
Tabla de valores durante la ejecución

Estructuras de autocompletado	LinkedList	Arrays	HashMap
Espacio en el Heap	60MB	175MB	384MB
Tiempo creación	1.16 - 1.34 s	0.82 - 1.1 s	2.23 - 2.6 s
Tiempo búsqueda ("a")	0.31 - 0.39 s	0.37 - 0.7 s	0.22 - 0.28 s
Tiempo búsqueda ("zyzzyvas")	0.088 ms	0.038 ms	0.06 ms
Búsqueda ("aerobacteriologically")	0.077 ms	0.041 ms	0.058 ms
Tiempo búsqueda todas las palabras	6.1 - 8.02 s	4.07 - 5.19 s	4.79 - 5.8 s

Table 4: Análisis de los resultados obtenidos con la implementación de la estructura de datos

5. TÍTULO DE LA SOLUCIÓN FINAL DISEÑADA

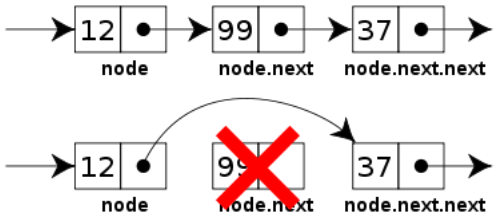
Implementen una estructura de datos para solucionar finalmente el problema y grafíquenla. Además, pruébenla con los datos que están en la carpeta de Conjunto de Datos del .ZIP



Gráfica 3: Lista simplemente encadenada de personas. Una persona es una clase que contiene nombre, cédula y foto

5.1 Operaciones de la estructura de datos

Diseñen las operaciones de la estructura de datos para solucionar finalmente el problema. Incluyan una imagen explicando cada operación



Gráfica 4: Imagen de una operación de borrado de una lista encadenada

5.2 Criterios de diseño de la estructura de datos

Expliquen con criterios objetivos, por qué diseñaron así la estructura de datos. Criterios objetivos son, por ejemplo, la eficiencia en tiempo y memoria. Criterios no objetivos y que rebajan la nota son: “me enfermé”, “fue la primera que encontré”, “la hice el último día”, etc. Recuerden: este es el numeral que más vale en la evaluación con 40%

5.3 Análisis de la Complejidad

Calculen la complejidad de las operaciones de la nueva estructura de datos para el peor de los casos. Vean un ejemplo para reportarla:

Método	Complejidad
Búsqueda Fonètica	O(1)
Imprimir búsqueda fonètica	O(m)
Insertar palabra busqueda fonètica	O(1)
Búsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir búsqueda	O(s)

Tabla 5: Tabla para reportar la complejidad

5.4 Tiempos de Ejecución

Calculen, (I) el tiempo de ejecución y (II) la memoria usada para las operaciones de la nueva estructura de datos, para el Conjunto de Datos que está en el ZIP. Explicar el tiempo para varios ejemplos

Tomen 100 veces el tiempo de ejecución y memoria de ejecución, para cada conjunto de datos y para cada operación de la estructura de datos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Creación	10 sg	20 sg	5 sg
Operación 1	12 sg	10 sg	35 sg
Operación 2	15 sg	21 sg	35 sg
Operación n	12 sg	24 sg	35 sg

Tabla 6: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

5.5 Memoria

Mencionar la memoria que consume el programa para los conjuntos de datos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Consumo de memoria	10 MB	20 MB	5 MB

Tabla 7: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

5.6 Análisis de los resultados

Expliquen los resultados obtenidos. Hagan una gráfica con los datos obtenidos, como por ejemplo:

Tabla de valores durante la ejecución

Estructuras de autocompletado	LinkedList	Arrays	HashMap
Espacio en el Heap	60MB	175MB	384MB
Tiempo creación	1.16 - 1.34 s	0.82 - 1.1 s	2.23 - 2.6 s
Tiempo búsqueda ("a")	0.31 - 0.39 s	0.37 - 0.7 s	0.22 - 0.28 s
Tiempo búsqueda ("zyzzyvas")	0.088 ms	0.038 ms	0.06 ms
Búsqueda ("aerobacteriologically")	0.077 ms	0.041 ms	0.058 ms
Tiempo búsqueda todas las palabras	6.1 - 8.02 s	4.07 - 5.19 s	4.79 - 5.8 s

Tabla 8: Tabla de valores durante la ejecución

6. CONCLUSIONES

Para escribirlas, procedan de la siguiente forma: 1. En un párrafo escriban un resumen de lo más importante que hablaron en el reporte. 2. En otro expliquen los resultados más importantes, por ejemplo, los que se obtuvieron con la solución final. 3.Luego, comparen la primera solución que hicieron con los trabajos relacionados y la solución final. 4. Por último, expliquen los trabajos futuros para una posible continuación de este Proyecto. Aquí también pueden mencionar los problemas que tuvieron durante el desarrollo del proyecto

6.1 Trabajos futuros

Respondan ¿Qué les gustaría mejorar en el futuro? ¿Qué les gustaría mejorar estructura de datos o a la implementación?

ACKNOWLEDGEMENTS

We are especially grateful to the national education project Ser Pilo Paga of the Government of Colombia and to the Fundación scholarship of Universidad EAFIT for financially support our education.

We would also like to thank Daniel Mesa, instructor of the Data Structures and Algorithms I course at Universidad EAFIT, for his advice in the learning process, for his contributions in class and his guidance in the realization of this project

REFERENCES

[1]. Liu, B. and Su, X. An Anti-Collision Algorithm for RFID Based on an Array And Encoding Scheme. Information, 2018, 2078-2489. Retrieved August 25, 2018 from Eafit University: https://bit.ly/2PEzPhx

[2]. Dinas, S. and Bañón J. M. A literature review pf bounding volumes hierarchy focused on collision detection. Ingeniería Competitiva, 2015, 49-62. Retrieved August 25, 2018, from Eafit University: https://bit.ly/2BMI9sD

[3]. Nevala, E. Introduction to Octrees. GameDev.net, 2018 https://gamedev.net/articles/programming/general-and-gameplay-programming/introduction-to-octrees-r3529/. Accessed September 23, 2018.

[4]. Spatial hashing implementation for fast 2D collisions. The mind of Conkerjo, 2013. https://conkerjo.wordpress.com/2009/06/13/spatial-hashing-implementation-for-fast-2d-collisions/. Accessed September 23, 2018.

[5]. How to efficiently remove duplicate collision pairs in spatial hash grid? Stack Overflow, 2015. https://stackoverflow.com/questions/31124702/how-to-efficiently-remove-duplicate-collision-pairs-in-spatial-hash-grid. Accessed September 23, 2018.