TECHNOLOGY AND ALGORITHMS: A SOLUTION TO BEE´S EXTINTION

Isabella Arango Restrepo
Universidad EAFIT
Colombia
iarangor1@eafit.edu.co

Juan David Rengifo Castro
Universidad EAFIT
Colombia
jdrengifoc@eafit.edu.co

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

ABSTRACT
The extinction of the human race is normally seen as sci-fi; however, this could be closer than expected. According to the IUCN (International Union for Conservation of Nature) "nearly one in ten wild bee species face extinction in Europe". This is something to pay close attention, due to these little pollinators are vital for earth's life. In fact, the bees are fundamental for plant's reproduction. In consequence, its extinction implies an authentic catastrophe, increasing the famine, the contamination, and the water reduction.

1. INTRODUCTION
It is not a secret that the human damage to earth is a big deal. The overpopulation and our self-destructive behavior have deteriorated the life on earth in almost all the aspects, and bees are not the exception. These little pollinators are an essential part of our life despite we are conscious or not. Sadly, the bee is in danger of extinction. According to BIP (Bee Informed Partnership) between 1988 and 2015 the 42.1% of the colonies died [1].

Keywords
Collision problems, algorithmic solution, extinction of bees.

ACM CLASSIFICATION Keywords
Computing methodologies → Artificial intelligence → Search methodologies → Discrete space search
Computing methodologies → Artificial intelligence → Planning and scheduling → Planning for deterministic actions
Computing methodologies → Modeling and simulation → Simulation types and techniques → Data assimilation
Computing methodologies → Modeling and simulation → Simulation types and techniques → Agent / discrete models

2. PROBLEM
The bee extinction is a major problem, due to this little animal is essential for plants reproduction. The extinction or decrease of its population avoid pollination, something that would increase the famine, the contamination and the water reduction. For this reason, this must be solved as soon as possible.

3. RELATED WORK
Collision detection is a very common problem that is applied in the programming of videogames, searches in multidimensional spaces, analysis of images and even in real life problems. To give an effective solution to this, different data structures are used to reduce the detection time of possible collisions.

3.1 Spatial hash.
A Spatial hash is a data structure that offers a faster way to solve a collision detection problem. It is a 2D o 3D table that consists of dividing the space into several cells and locating the objects that are in it according to their spatial coordinates. Then, these are placed in a hash table in such a way as to achieve an effective location of those found in the same frame.

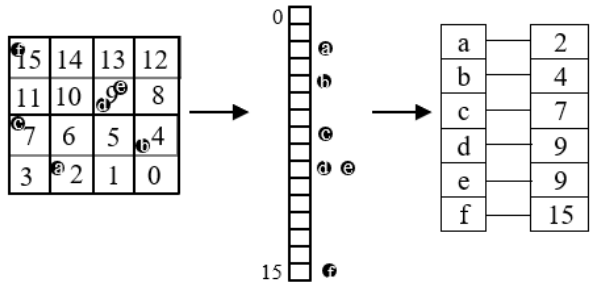

Figure 1: Spatial hash representation.

3.2 AABB tree.
This data structure is very useful and effective at searching objects collisions in an undetermined three-dimensional space. It consists of boxes that are aligned and represent the coordinates of each element [2]. Then, it is very easy to determine if they present an intersection, because it is enough to look if the following is met or not:

maxx1>minx2 & minx1<maxx2 & maxy1>miny1 & miny1<maxy2,

where maxx1 refers the x-coordinate of the upper right corner of object1, maxy1 to the y-coordinate of the same corner, minx1 and miny1 for the lower left corner and so on for the object2.

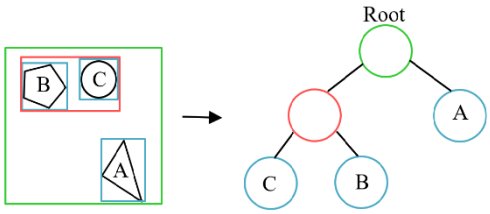

Figure 2: AABB tree representation.

3.3 Quadtree.
The quadtree is a type of spatial decomposition that consists of dividing the space recursively into four cells and when one reaches its maximum capacity, it is divided again into four cubes. They are too useful when detecting collisions, because it allows to evaluate only those objects that may collide, instead of reviewing the entire list of objects that are in space. For this reason, it takes less time to determine those elements that are very close.

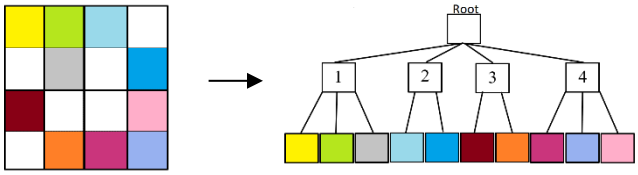

Figure 3: Quadtree representation.

3.4 R-Tree.

This data structure is very used in problems with multidimensional spaces, such as map locations. In addition, it is one of the most efficient methods to find objects that are within a space delimited by a rectangle, which allows searching in a specified area of the tree, instead of searching in each sector that composes the space. An advantage is that it allows to work and analyze large volumes of data without spending a lot of time [4].

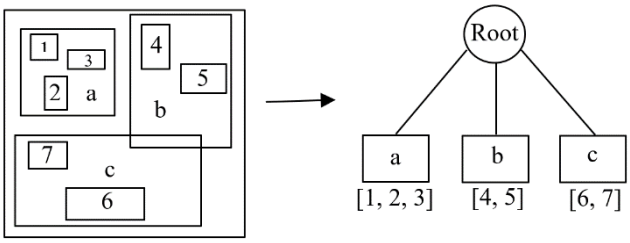

Figure 4: R-Tree representation.

4. ArrayList

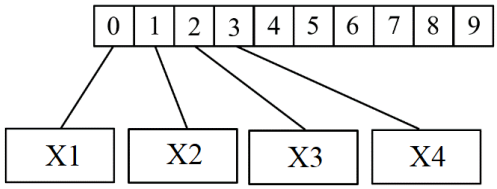

Figure 5: ArrayList of bee´s coordinates. A coordinate is double number that represent the latitude, longitude or height.

4.1 Operations of ArrayLists

The operations that the ArrayList of coordinates can do are:

a. Add in the last position:

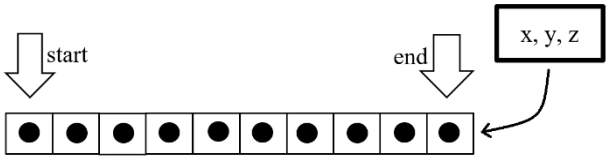

Figure 6: Add operation of ArrayLists.

b. Get:

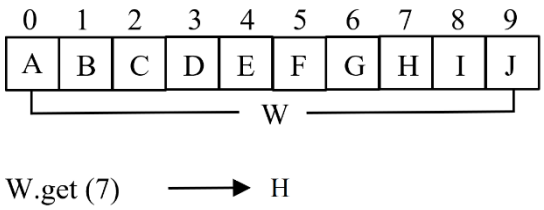

Figure 7: Get operation of an ArrayList.

4.2 Design criteria of ArrayLists

Choosing an appropriate data structure that helps to implement a functional and efficient code is very important at the time of thinking in a solution to solve an algorithmic problem. In this case, the ArrayList structure seems to be useful to solve collision problems due to its properties and the ease of working with it. For example, it allows to study different groups of data without knowing exactly their size or if it is going to increase or decrease. Is important to mention, that our method with the highest complexity needs the operation get(), and clearly array list have the lowest complexity in this operation against other data structures, such as a Linked list, stack, queue, or a binary three. Also, this data structure has complexities smaller or equals to O(n/2), which is efficient compared to the complexities of other structures.

4.3 Complexity analysis

Method	Complexity
Add	O(1)
Remove	O(n/2)
Get	O(1)

Table 1: Table to report complexity analysis.

4.4 Execution time

Operations	10 bees	100 bees	1000 bees	10000 bees
Add	0.8ms	0.86ms	10.8ms	97.9ms
Get	0ms	0ms	0ms	0ms

Table 2: Execution time of the operations Add and Get of ArrayLists for each data set. Time is in milliseconds.

4.5 Memory used

4.6 Result analysis

Finally, ArrayLists are a very efficient data structure to solve collision problems because they present very low time complexity compared to other structures and this can be seen in the execution time that it takes to analyze data groups of up to 10000 elements.

REFERENCES

1. LA VANGUARDIA. Available from https://www.lavanguardia.com/natural/20161005/41771284333/abeja-peligro-humanos.html (2016); accessed 24 August 2018.
2. Runestone. Available from http://interactivepython.org/runestone/static/pythoned/SortS earch/TransformacionDeClaves.html (2012); accessed 24 August 2018.
3. Azure from the trenches. Available from: https://www.azurefromthetrenches.com/introductory-guide- to-aabb-tree-collision-detection/ (2017); accessed 24 August 2018.
4. Wkipedia. Available form https://en.wikipedia.org/wiki/R- tree (2018); accessed 24 August 2018.
5. Evantotuts+. Available from https://gamedevelopment.tutsplus.com/tutorials/quick-tip- use-quadtrees-to-detect-likely-collisions-in-2d-space-- gamedev-374 (2018); accessed 24 August 2018.

6. Wikipedia. Available from https://en.wikipedia.org/wiki/Quadtree#Some_common_uses_of_quadtrees (2018); accessed 24 August 2018.
7. The IUCN Red List of Threatened Species. Available from http://www.iucnredlist.org/news/nearly-one-in-ten-wild-bee-species-face-extinction-in-europe-while-the-status-of-more-than-half-remains-unknown-iucn-report (2018); accessed 24 August 2018.
8. Big-O Cheat Sheet. Available from http://bigocheatsheet.com/ (2018); accessed 20 August 2018.
9. Information Technology Gems. Available from http://infotechgems.blogspot.com/2011/11/java-collections-performance-time.html (2018); accessed 20 August 2018.
10. Enrique Munguía. Available from http://www.enrique7mc.com/ (2018); accessed 22 August 2018.