Bee collision detection

Juan Iguaran
Universidad Eafit
Colombia
jciguaran@eafit.edu.co

Juan Andres arroyave
Universidad Eafit
Colombia
jaaroyavs@eafit.edu.co

Gustavo Lopez
Universidad Eafit
Colombia
glopezg@eafit.edu.co

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

SUMMARY

. The solution proposed to solve the problem and detect the collisions was to use an array of objects, those objects of type abjea, like the arrayList. The data of a text file will be read. Each bee will have 3 coordinate data: latitude, longitude, and meters above sea level. That said, we create the arrayList, with a split, that will separate the elements each time you see a comma ",".

Already in the method that will detect the collisions an algorithm was created, which compared two bees by mathematical equations, and if the difference was less than or equal to 100, it prints the bees that are at risk of collision, this, doing it through an aninated cycle .

The results yielded the data of the bee that is in collision with another bee.

It was concluded that, in the case of 10 bees, there are 2 possibilities of collision, and that if the problem of intersecting is not fixed, they could pollinate the same flower and leave another flower without pollinating, or even collide with each other.

5. Detection of collision of bees by arrayList

	0	1	2
0	**lat**	**l**ongi	**msn**
1	**lat**	**longi**	**msn**
2	**lat**	**longi**	**msn**

Graph 1: arrayListt of bees. A bee is a class that contains latitude, longitude and height above sea level.

5.1 Data structure oprerations

$$\sqrt{((0.lat - 1.lat)*111111)^2 + ((0.longi - 1.longi)*111111)^2 + (0.msn - 1.msn)^2}$$

Graph 2: Image of an operation to calculate distance between a bee in position 0 and the other in position 1 in an array.

5.2 Design criteria of the data structure.

We used an arrayList mainly because of its ease of entering an element, because to enter an arrayList its complexity is 0 (1) in the worst case, and that is what we need to be able to calculate the distance of the obejas, power access the element. In contrast to an arrayList is a list, a queue, a LinkedList or a Hash table, whose complexity to access an element is O (n).

Another factor by which we determine to create an arrayList, was because it is one of the structures that occupy less space in memory, because its complexity is O (n), like a table of Hash, or a list or a queue .

As a last important factor, there is the facility to access and manage arrayList, inserting and managing elements in an arrayList allowed us to develop the project in the best way.

5.3 Analysis of Complexity

Method	complexity
Insert element in the arrayList.	O(n)
Search an element	O(1)
Calculate the distance of two bees	O(n^2*e^3)

Table 1: Table to report complexity

5.4 Execution Times

	10 bees	100 bees	1000 bees	10000 bees	100000 bees	1000000 bees
creation	10 ms	8 ms	64 ms	94 ms	173 ms	1723ms
distance calculation	1 ms	16 ms	266 ms	322 ms		

Table 2: Execution times of data structure operations with different data sets

5.5 Memory used

Number of bees	Consume in KiloBytes
10	90552
100	46392
1000	293904
10000	293904

Table 3: Memory consumption of the data structure with different data sets

5.6 Result analysis

Memory with ArrayLists to collitions	Memory with HashTable to collitions	Memory using 3D Matrices to collitions
O(N2*e^3)	O(n*log(n)*e^3)	O(n+k*e^3)

Time with ArrayList to read all the elements in the dataset.	Time with hashTables to read all the dataset.	Time using 3D matrices to read and save
O(N)	O(N)	O(N)

Table 4: Table of values during execution

6. CONCLUSIONS

We can conclude that the data structure that we used is usedfull. Why? the answer won´t be short and we have to explain it being the most rigorous at possible

This data structure stop at the item indicated by i.

This is an advantage because it will stop at the element we indicate, quickly and effectively.

This data structure does not consume a lot of memory compared with anothers, and the time it takes to execute the code is acceptable.

The solution here raised with Arraylist saves more memory purchased with structures likeoctree, quadtree, spaciated hash,etc are very good solutions because these doesn´t compare all with all elements.

6.1 Future works

We would like to use our data structure specially in how to optimize our code, we think that there may be ways to make our code more efficient in terms of time. And at the same time, it can serve not only to detect collisions of abjeas, but to detect different types of collisions, even for simple games.

We believe that this Project could have O(log(n)) as the smallest complexity,so we will work in this code to make it better.

ACKNOWLEDGEMENTS

We are really grateful with the teaching asistants because they helped to make this project easier,without them we think that this project would be imposible.

To the program Ser Pilo Paga, without this program we won´t be here.

To our families, because they helped everyday with their love, their support giving us the energy to continue with this Project, and with this hard semester, with them everything started.

And finally to Juan José Iguarán who taught us how to work with arraylist, because when the Project started we were really confused because we didn´t understand good what do we had to do.