DATA STRUCTURES FOR EARLY DETECTION OF BLIGHT PLAGUE ON COFFEE PLANTATIONS

Simón Flórez Silva
Universidad Eafit
Colombia
sflorezs1@eafit.edu.co

Adrián Alberto Gutiérrez Leal
Universidad Eafit
Colombia
aagutierrl@eafit.edu.co

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

NOTE: To have more information about the sections in this report, please read "*Guía para la realización del Proyecto Final de Estructura de Datos 1.*" For the final version of this report: 1. Detele all text in red. 2. Adjust spaces among words and paragraphs. 3. Change the color of all the texts to black. You should also keep in mind the meaning of the colors:

Black text = To complete for the 1st deliverable

Blue text = To complete for the 2nd deliverable

Violet text = To complete for the 3rd deliverable

ABSTRACT

In the following paper, we address the blight plague on coffee plantations, a problem that affects Colombian agriculture and therefore, its economy. In addition, lack of automation makes it harder to make an early diagnosis and, of course, makes it harder to control. As it was previously mentioned, the importance of this problem relapses on its consequences over the coffee exportation, from which thousands of Colombian families depend and, of course, the economy of the country as a whole. Some have already tried to treat this problem by implementing algorithms such as ID3 and C4.5 (which is a slightly better version of the latter) algorithms for decision trees.

KEYWORDS

• Mathematics of computing~Trees • Mathematics of computing~Graph algorithms • Information systems~Data access methods • Human-centered computing~Displays and imagers • Theory of computation~Recursive functions • Theory of computation~Data structures design and analysis • Hardware~Sensors and actuators • Hardware~Design modules and hierarchy • Software and its engineering~Interpreters • Software and its engineering~Virtual machines • Software and its engineering~Software performance • Software and its engineering~Object oriented languages • Software and its engineering~Abstract data types • Software and its engineering~Data types and structures • Software and its engineering~Classes and objects • Software and its engineering~Recursion

1. INTRODUCTION

In the current times, it becomes evident that automation can make some difficult processes a lot easier. That is the case for the early diagnosis and control of the blight plague on coffee plantations, a problem that has been harming productivity and incomes of Colombian coffee production during the last years. In this paper, the team will approach different algorithmic solutions and implement decision trees in order to detect this fungus as soon as possible.

2. PROBLEM

The main problem, from a software perspective, is to use Data Structures for the design of an algorithm based on decision trees to predict if a caturra coffee batch is infected with the blight plague. It's worth noting that it has an immense impact on Colombian economy, as coffee consists in one of Colombia's main exports and therefore, many people depend on it to sustain themselves.

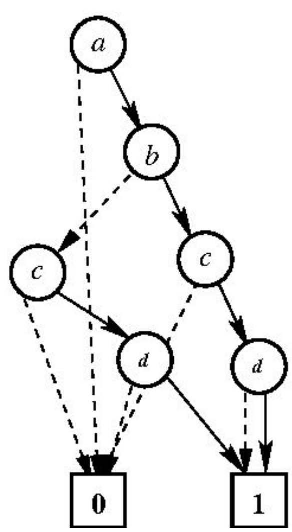
3. RELATED WORK

3.1 Fuzzy matching (Similar data matching)

According to Megter, Fuzzy matching is a technique used in computer-assisted translation as a special case of record linkage. It works with matches that may be less than 100% perfect when finding correspondences between segments of a text and entries in a database of previous translations. It usually operates at sentence-level segments, but some translation technology allows matching at a phrasal level. It is used when the translator is working with translation memory.

A solution example could be the n-gram algorithm, which is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n − 1)–order Markov model.

Sequences	Unordered 4-grams	Combinations
aabc	abc	*abc*
cabc	abc	*abc*
abcd	abcd	*abcd*
aacd	acd	*acd*

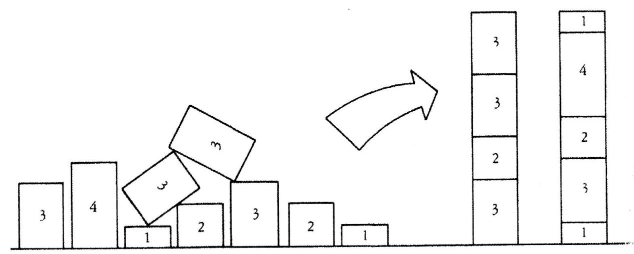


Example for the n-gram algorithm

3.2 Partition Problem

Following Naumenko, F., Partition Problem consists of deciding whether a given set of positive integers with count of N can be partitioned into K subsets such that the sum of the numbers in each subset is equal. In actual practice, the exact equality of the sums is usually unattainable. A common example of its application would be the distribution of a set of independent works, each with different execution time, in several concurrent threads. If each thread runs on a separate core, we would like to be confident in the uniform load of each core in the CPU node.

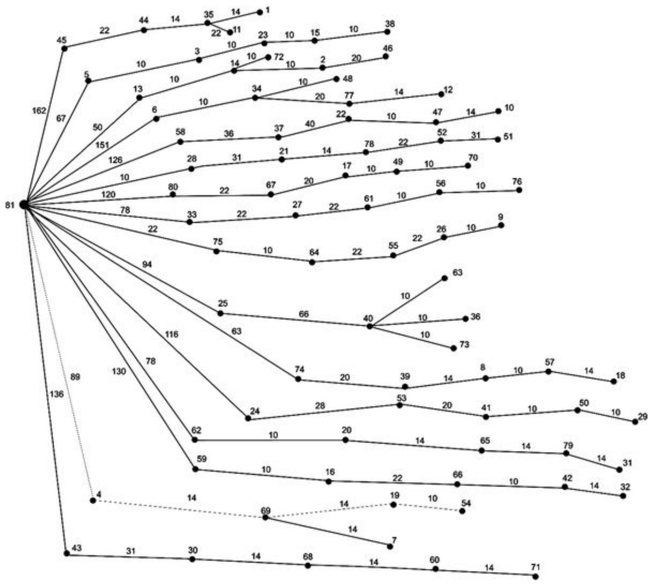
There is an easy and well-known algorithmic approach for this problem, that could actually be solved by children, for this, the algorithm iterates through the numbers in descending order, assigning each of them to whichever subset has the smaller sum. This approach has a running time of O(n log n).



3.3 Capacitated minimum spanning tree

Capacitated minimum spanning tree is a minimal cost spanning tree of a graph that has a designated root node “r” and satisfies the capacity constraint “c”. The capacity constraint ensures that all subtrees (maximal subgraphs connected to the root by a single edge) incident on the root node r have no more than c nodes. If the tree nodes have weights, then the capacity constraint may be interpreted as follows: the sum of weights in any subtree should be no greater than c. The edges connecting the subgraphs to the root node are called gates.

For its solution, there is the Esau-Williams heuristic, which finds suboptimal CMST that are very close to the exact solutions, but on average EW produces better results than many other heuristics. Initially, all nodes are connected to the root ***r*** (star graph) and the network's cost is ; each of these edges is a gate. At each iteration, it seeks the closest neighbor a_j for every node in G - r and evaluate the tradeoff function: $t(a_i) = g_i - c_{ij}$. Then, it looks for the greatest $t(a_i)$ among the positive tradeoffs and, if the resulting subtree does not violate the capacity constraints, removes the gate g_i connecting the i-th subtree to a_i by an edge c_{ij}. We repeat the iterations until we can not make any further improvements to the tree.

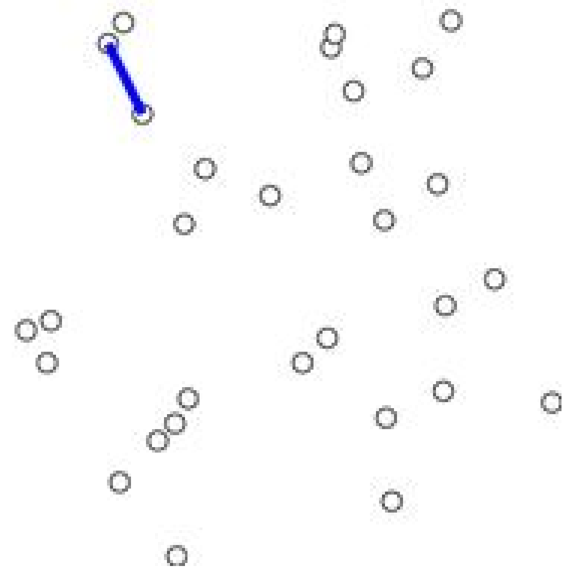


Example for the Essau-Williams Heuristic

3.4 **Minimum spanning tree**

A minimum spanning tree (MST) is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge weights is as small as possible. More generally, any edge-weighted undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components.

Then, for this problem, one of its possible solutions could be Prim's (also known as Jarník's) algorithm, a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph, by finding a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.



Representation of Prim's Algorithm

4. Decision Tree of Coffee Data

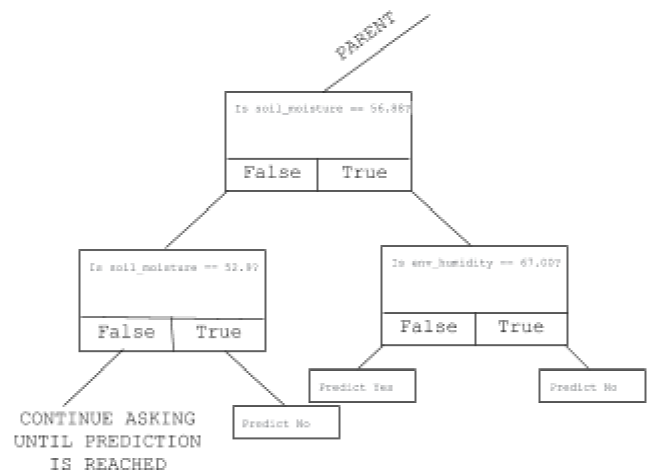


Figure 1: Tree of Coffee Data. Each node represents a type of data which will generate a hierarchy based on the type and value of the data to determine as quick as possible if the plants are sick.

4.1 Operations of the data structure

The traverse method, “traverses” each node of the Tree and counts them:

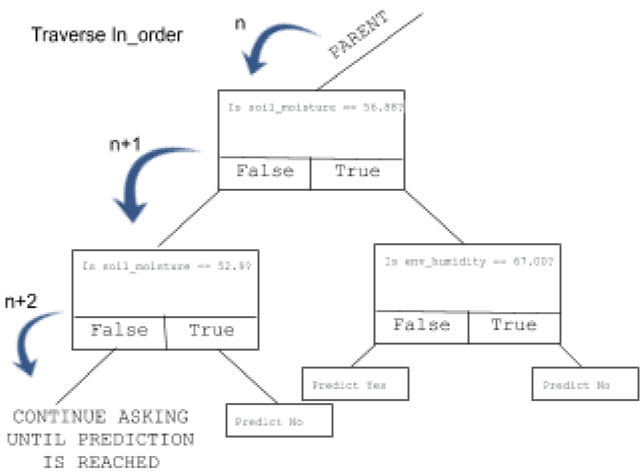


Figure 2: Traverse

The find method performs the task of searching each node for a specific input data:

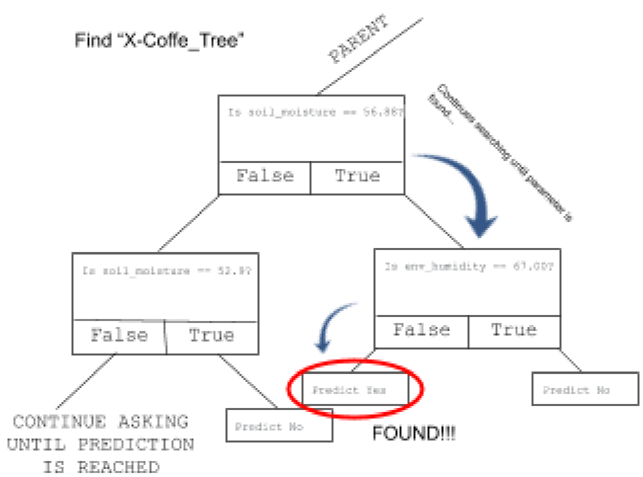


Figure 3: Find

As for the remove method, it searches for a node and deletes it based on a specific input data.

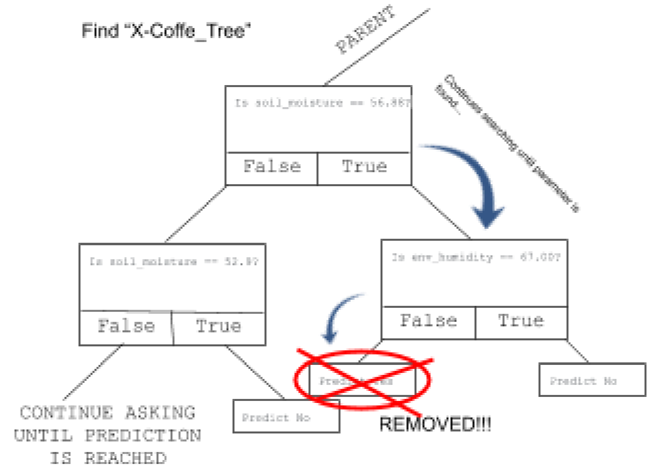


Figure 4: Remove

4.2 Design criteria of the data structure

The design of this data structure was based on the implementation of the Decision Tree data structure, from which branches are generated and so, create a hierarchy of the data collected from a coffee tree, using the importance of the type of data and whether or not its values are outside the reviewed parameters to decide as fast as possible if the tree is infected or if it's healthy. For that matter, one of the best ways to decide is by implementing a decision tree, for it considers every outcome based on the data it has and allows to take the best possible decision for a problem for which solution would be otherwise uncertain.

It's worth to add that these Trees are also pretty easy to implement, due to them being based on simple math and provide a visual representation of the data, which makes even easier the decision taking process.

4.3 Complexity analysis

The complexity for each operation can be seen in the next table:

Method	Complexity
Remove	O(n)
Traverse (Post-Order)	O(n)
Traverse (Pre-Order)	O(n)
Find	O(n)
Full Algorithm (Create Tree)	O(v · nlogn)

Table 1: Table to report complexity analysis

4.4 Execution time

	Data set test 1	Data set test 2	Data set test 3
Full Algorithm (Create Tree)	35 sec	32 sec	38 sec
Remove	0.097 ms	0.073 ms	0.088 ms
Traverse	0.077 ms	0.053 ms	0.074 ms
Find	0.094 ms	0.035 ms	0.108 ms

Table 2: Execution time of the operations of the data structure for each data set.

4.5 Memory used

	Data set 1	Data set 2	Data set 3
Memory Consumption	10 MB	9.9 MB	10.7 MB

Table 3: Memory used for each data set.

4.6 Result analysis

For the data structure we designed, the worst case scenario of its runtime would be O(v · nlogn), still it works relatively fast considering the amount of data that it has to manage. Some functions such as removing, finding or traversing take as little as O(n) to fulfill their respective tasks, being this, of course, an approximation of the worst case scenario, forgiven a rather lucky value, it could take O(1) to run, which is a very optimistic, even ideal and yet possible runtime for these algorithmic solutions.

Values during execution	
Task	**Performance**
Memory Space	10 MB
Create Tree	32 sec. - 38 sec. 35 sec. in average
Remove	0.073 ms. - 0.097 ms. 0.085 ms. in average
Traverse	0.053 ms. - 0.077 ms. 0.065 ms. in average
Find	0.035 ms. - 0.108 ms. 0.072 ms. in average

Table 4: Analysis of the results

5. Title of the last data structure designed

Design a data structure to solve the problem and make a figure explaining it. Do not use figures from the Internet.

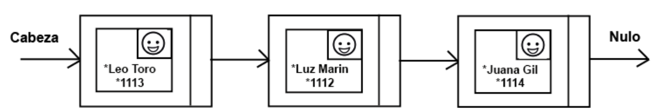


Figure 1: Linked List of persons. Una person is a class that contains a name, id number and photo.

5.1 Operations of the data structure

Design the oepration of the data structure to solve the problem efficiently. Include one figure to explain each operation.

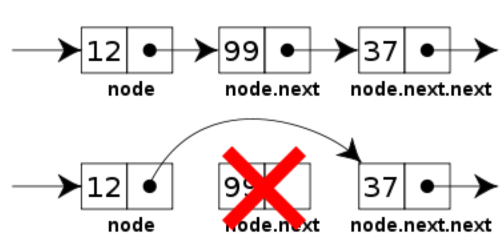


Figure 2: Delete operation of a Linked List.

5.2 Design criteria of the data structure

Explain objetive criteria that you considered to design the data structure. Examples of objective criteria are efficiency in time and space. Non-objective criteria will lower your grade. Examples of non-objetive criteria are: "I was sick", "it was the first data structure that I found on the Internet", "I did it on the last day before deadline", etc. Remember: This is 40% of the project grade.

5.3 Complexity analysis

Derive the complexity of each operation of the data structure for the worst case and best case, As an example, this is a way to report the complexity analysis:

Mètodo	Complejidad
Bùsqueda 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 bùsqueda	O(s)

Table 5: Table to report complexity analysis

5.4 Execution time

Measure (I) execution time and (II) memory used by the operations of the data structure, for the data set found in the .ZIP file.

Measure the execution time and memory used 100 for each data set and for each operation of the data structure. Report the average values.

	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

Table 6: Execution time of the operations of the data structure for each data set.

5.5 Memory used

Report the memory used for each data set

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

Table 7: Memory used for each operation of the data structure and for each data set data sets.

5.6 Result analysis

Explain the results obtained. As an example, compare different implementation of the data structure and report the comparison in a table or graph.

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 8: Analysis of the results

6. CONCLUSIONS

To write the conclusions, proceed in the following way. 1. Write a paragraph with a summary, the most important issued of the report. 2. In another paragraph, explain the most important results that you obtained with the last data structure you designed. 3. Compare your first solution with the last solution. 4. At last, explain future work, a future continuation of this project. You can also mention in the conclusions, technical problems that you had during the development of the data structure and its implementation and you solved them.

6.1 Future work

Answer, what would you like to improve in the future? How would you like to improve your data structure and its implementation?

ACKNOWLEDGEMENTS

Identify the kind of acknowledgment you want to write: for a person or for an institution. Consider the following guidelines: 1. Name of teacher is not mentioned because he is an author. 2. You should not mention websites of authors of articles that you have not contacted. 3. You should mention students, teachers from other courses that helped you.

internet ni autores de artículo leídos con quienes no se han contactado. 3. Los nombres que sí van son quienes ayudaron, compañeros del curso o docentes de otros cursos.

As an example: This research was supported/partially supported by [Name of Foundation, Grant maker, Donor].

We thank for assistance with [particular technique, methodology] to [Name Surname, position, institution name] for comments that greatly improved the manuscript.

REFERENCES

1. Fischer, G. and Nakakoji, K. Amplifying designers' creativity with domainoriented design environments. in Dartnall, T. ed. Artificial Intelligence and Creativity: An Interdisciplinary Approach, Kluwer Academic Publishers, Dordrecht, 1994, 343-364.

2. Megter: Fuzzy Matching Algorithms To Help Data Scientists Match Similar Data, 2016. Retrieved August 10, 2019, from Data Science Central: https://www.datasciencecentral.com/profiles/blogs/fuzzy-matching-algorithms-to-help-data-scientists-match-similar

3. Naumenko, F: Fast and Practically 'perfect' Partition Problem Solution, 2018. Retrieved in August 10, 2019, from Code Project: https://www.codeproject.com/Articles/1265125/Fast-and-Practically-perfect-Partition-Problem-Sol

4. Kurai R., Minato S., Zeugmann T.: Unordered N-gram Representation Based on Zero-suppressed BDDs for Text Mining and Classification, 2007. Retrieved in August 17, 2019, from: http://bit.ly/2Zk4I2x

5. Wikipedia contributors. "Capacitated minimum spanning tree." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 9 Feb. 2019. Web. Retrieved in August 10, 2019, form: http://bit.ly/2KDfLLY

6. Wikipedia contributors. "Minimum spanning tree." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 11 Aug. 2019. Web. 17 Aug. 2019. Retrieved in August 10, 2019, form: http://bit.ly/2Hev1N0

7. Kritikos, M. & Ioannou, G. J Oper Res Soc (2017) 68: 1223. https://doi.org/10.1057/s41274-016-0146-7