ENGINEERING METHOD

MEMBERS:

Gabriel Suarez -A00368589SIS
Alejandro Varela -A00369019TEL & SIS
Luis Alfonso Murcia Hernández-A00369008 TEL

TEACHER:

Anibal Sosa Aguirre

ALGORITHMS AND DATA STRUCTURES

ICESI UNIVERSITY
2021-2

Problematic Context: During the semester we have approached different subjects oriented to the graphs, of which we were asked to carry out a final project based on them. So, a program is requested that implements different algorithms and solves a specific problem using everything seen in the last weeks of the semester.

Solution development: We decided to rely on the engineering method to solve this problem. By searching various sources, it was possible to realize that 7 steps are needed for an efficient problem solving.

- **Phase 1**: Identification of the Problem: Although it may not seem like it, it is the most important phase of the engineering method, a good problem must be defined to know what they are asking for and based on this, begin to develop creative, effective solutions and, above all, innovative.
- **Phase 2:** Gathering the Necessary Information: It is necessary to know the number of stores that offer the same service, the feasibility of incorporating the store into the market and the location.
- **Phase 3**: Search for Creative Solutions: Allows brainstorming
- **Phase 4**: Transition from the Formulation of Ideas to the Preliminary Designs: Collaborates in the discarding of ideas in the previous step and deepening of the most viable ones.
- **Phase 5**: Evaluation and Selection of the Best Solution: After an exhaustive analysis, the true solution is chosen.
- **Phase 6**: Preparation of Reports and Specifications: The most important aspects of the project are documented.
- **Phase 7**: Design Implementation: We proceed to the experimental stage to carry out the idea of the 6 previous steps.

In order to be precise in the process of developing this method, each step will be clearly explained:

Phase 1: Problem Identification

Before the imminent closing of the 2021-2 semester, Professor Anibal Sosa wanted to test his students of Algorithms and Data Structures, with a project based on the theory and practice of graphs, which, as is well known, is a data structure that allows the symbolic representation of the constituent elements of a system or set, through graphic diagrams. For this reason, he proposed a series of **conditions** to limit the students and guide them along the best possible path to find a solution.

- **Develop** two versions of Graph (your solution must work without problem with both versions, that is, the program must allow the change of the implementation used at any time and work well regardless of the one being used). Each graph must be developed from the TAD to the automatic unit tests.
- **Make** and document each of the phases of the engineering method for solving the problem.
- **Properly** document the analysis and design phases with the requirements specification document, the TAD design, class and object diagrams, and the design of the automatic unit test cases.
- **Manage** a graphical user interface that allows to use the functionalities that respond to the requirements of the problem.
- **Implement** all the graph algorithms seen in class even if they are not used in your project to solve the problem.

Phase 2: Gathering the necessary information

Because it has not yet been strictly defined what is going to work, it is necessary to know the concepts, tools and applications of the theory that is going to be applied to the project, for that reason a glossary has been made of everything essential to do this most enjoyable project.

GitHub: Which is a collaborative software development platform to host projects using the Git version control system. It hosts a code repository and provides you with very useful tools for teamwork within a project. Besides that, you can contribute to improve the software of others. A very important tool to work with all the people who will support in the creation of the project. Besides that, it provides us with certain useful features, such as:

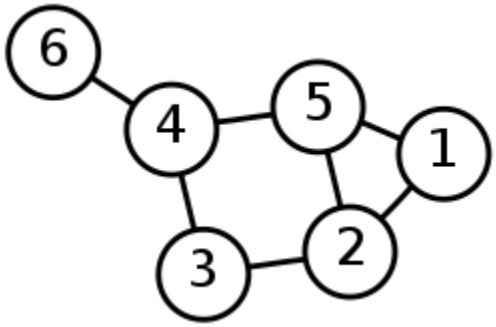
- A wiki for the maintenance of the different versions of the pages.
- A problem tracking system that allows your team members to detail a problem with your software or a suggestion they want to make.
- A code review tool, where you can add annotations at any point in a file and discuss certain changes made in a specific commit.
- A branch viewer where you can compare the progress made in the different branches of our repository.

Among the concepts that can be found in the implementation phase, we have some such as:

TAD: An abstract data type (hereinafter TDA) is a set of data or objects to which operations are associated. The TDA provides an interface with which it is possible to perform the allowed operations, abstracting from the way in which these operations are implemented. This means that the same ADT can be implemented using different data structures and provide the same functionality.

Graph: A graph is a set of objects called vertices or nodes joined by links called edges or arcs, which allow representing binary relationships between elements of a set. Typically, a graph is represented graphically as a set of points (vertices or nodes) joined by lines (edges or arcs).

From a practical point of view, graphs allow us to study the interrelationships between units that interact with each other. For example, a computer network can be represented and studied by means of a graph, in which the vertices represent terminals, and the edges represent connections (which, in turn, can be cables or wireless connections).



Path algorithms:

DFS: A deep search (DFS) is a search algorithm for which it traverses the nodes of a graph. Its operation consists of expanding each of the nodes that it locates on a recurring basis (from the parent node to the child node). When there are no more nodes to visit on that path, it returns to the predecessor node, so that it repeats the same process with each of the node's neighbors. It should be noted that if the node is found before going through all the nodes, the search ends.

BFS: A width search (BFS) is a search algorithm for which it goes through the nodes of a graph, starting at the root (choosing some node as the root element in the case of a graph), and then exploring all the neighbors of this node. Then, for each of the neighbors, their respective adjacent neighbors are scanned, and so on until the entire graph is traversed. It should be noted that if the node is found before going through all the nodes, the search ends.

Minimum weight roads:

Dijkstra's algorithm: It is an algorithm for determining the shortest path, given an origin vertex, towards the rest of the vertices in a graph that has weights on each edge. The underlying idea in this algorithm consists of exploring all the shortest paths that start from the origin vertex and that lead to all the other vertices; when the shortest path is obtained from the origin vertex to the rest of the vertices that make up the graph, the algorithm stops. It is a specialization of the uniform cost search and, as such, it does not work in graphs with negative cost edges (by always choosing the node with the smallest distance, nodes that in future iterations would lower the overall cost may be excluded from the search of the road when passing a negative cost edge).

Floyd-Warshall algorithm:

The Warshall algorithm is an example of a Boolean algorithm. Starting from an initial table composed of 0`s (there is no initial correspondence in the graph) and 1`s (there is a correspondence, called “arrow”, between nodes), it obtains a new matrix called “Transitive Closure Matrix” in the one that shows all the possible unions between nodes, directly or indirectly. That is, if from "A" to "B" there is no "arrow", it is possible that there is from "A" to "C" and then from "C" to "B". Then, this result will be dumped in the final matrix. Floyd's algorithm is very similar, but it works with weighted graphs. That is, the value of the "arrow" that we represent in the matrix can be any real or infinite number. Infinity marks that there is no union between the nodes. This time, the result will be a matrix where the minimum distances between nodes will be represented, selecting the most convenient paths according to their weighting ("weight"). For example, if from “A” to “B” there are 36 (km), but from “A” to “C” there are 2 (km) and from “C” to “B” there are 10 (km), the algorithm it will finally return that from "A" to "B" there are 12 (km).

Minimum Cover Tree

Prim algorithm: It is an algorithm belonging to the theory of graphs to find a minimal covered tree in a connected graph, not directed and whose edges are labeled. In other words, the algorithm finds a subset of edges that make up a tree with all vertices, where the total weight of all edges in the tree is the minimum possible. If the graph is not connected, then the algorithm will find the minimum covering tree for one of the connected components that make up said non-connected graph.

Kruskal algorithm: It is an algorithm of graph theory to find a minimal covered tree in a connected and weighted graph. That is, it looks for a subset of edges that, forming a tree, include all vertices and where the value of the sum of all the edges of the tree is the minimum. If the graph is not connected, then it looks for a minimal expanded forest (a minimal expanded tree for each connected component)

Phase 3: Search for creative solutions

Because it is necessary to have several proposals from all the members to be able to analyze what is the solution that is going to be carried out, we decided to **brainstorm ideas** that were not so elaborate and thought in **just 5 minutes**.

Alejandro Varela:

Idea 1: An application that will find the best route for garbage collection in a not so big city.

Idea 2: A maze-like game where the player needs to go through several doors, with each door having a different theme, with a number of attempts and with the aim of finding the end.

Gabriel Suarez:

Idea 1: A Snake and Ladder modeled with graphs, where if you step on a snake, you go down, and if you touch a ladder, you go up.

Idea 2: A game similar to minesweeper but modeled with graphs where the player must complete the square without clicking on any mine.

Luis Murcia:

Idea 1: A simple maze where the person could see the game crosswise to find the exit.

Idea 2: An application that will help you find the best way for a mail truck.

Phase 4: Transition from Idea Formulation to Draft Designs

We need to find the best solution to solve the problem we are developing, for this reason it is necessary to review each of the proposals of all the members who contributed their grain of sand. We began by analyzing each of the proposed solutions, and in most it was possible to see how they were searching for efficient paths or games that were modeled in a simple way using graphs.

Due to the wide complexity of an application focused on obtaining the best route in a city and due to the large number of streets and highways that there may be the garbage collection system and the correspondence system are **instantly discarded**. We also have that despite the fact that the famous minesweeper game can be modeled with graphs, it is not very clear how it is possible to embrace all the necessary algorithms to make the game in a complete and efficient way, so it is **not considered** to resort to phase number 5.

Among the ideas that are not discarded, we have Alejandro's idea **number 2**, which is **based** on a system that allows calculating the best route to reach the end of the maze and also will be the number of attempts that the user will give to finish the game. On the other hand, the ideas of **Gabriel (idea 1)** and **Luis (idea 1)** are based on a traditional game system, without any changes.

In this way we are left with three **(3)** possible **ideas** of reformulation to enter the next phase:

Phase 5: Evaluation and selection of the best solution

Alejandro Varela (Idea 1):

- **Labyrinth with doors**

As it is a non-traditional game system, it is a very good option for the public to try something new.

Gabriel Suárez (Idea 2):

- **Snake and Ladder**

A game quite loved by many and hated by those who found it difficult to win.

Luis Murcia (Idea 3):

- **Labyrinth seen from above**

A traditional maze that we all know.

Evaluation criteria:

Criterion 1: Application innovation

- [2] Innovative
- [1] Little Innovative
- [0] Nothing innovative

Criterion 2: Compliance with the requirements in the development of the problem

- [2] Meets all requirements
- [1] Meets some of the requirements
- [0] Does not meet any of the requirements

Criterion 3: Ease of use of the application

- [3] Easy
- [2] Normal
- [1] Difficult

	Criterion 1	Criterion 2	Criterion 3	Total
Idea 1	2	2	3	7
Idea 2	0	1	3	4
Idea 3	0	2	3	5

Thanks to these evaluation criteria, we agree that the best **solution** to carry out is **one**, and it makes sense, it complies with most of the guidelines that were made from the first stage

Phase 6: Preparation of reports and specifications

You need to analyze the documents in the doc's folder. of this same project.

Phase 7: Design implementation

Our project design implementation is mounted on GitHub with a enlace to the repository.