Integrative task I
Computing and Discrete Structures I
Development of the Engineering Method

Members:

- Juan Jose Diaz A00381098
- Ana Sofía Londoño A00380882
- Mateo Silva A00382277

1. Implementing the engineering method:

Step 1 - Identification of the problem.

Context of the problem

A recognized delivery company named “Rappiando” requires the creation of a system that helps the employees know the most efficient way to get from one place to another. Due to the heavy vehicular traffic in the city of Santiago de Cali, there is a lot of uncertainty when it comes to selecting a route when driving around the city. This system is specialized in analysis just south of Santiago de Cali, as it is the range that the company can make deliveries in the city. The system must have 2 functionalities: showing the shortest path between 2 points in the city of Santiago de Cali, and another that shows a route that can cover the largest number of points in the city of Santiago de Cali to be able to promote new products of the company.

Problem definition

The delivery company “Rappiando” has requested a program that allows choosing the most efficient way to get around in the south of the city of Santiago de Cali.

Definition of needs

- The system must have a graphic interface that presents 2 options to select. Also, it needs to show the map of the South of Cali and its most busiest points.

- The system must have the option of calculating the fastest path between two points in the south of Santiago de Cali.

    - The system must show graphically the route in the map of the fastest route according to the points given by the user.

- The system must have the option of calculating the most optimal path that the company should take to distribute brochures throughout the south of Cali, taking into account that having infinite distributors.

    - The system must show graphically in the map this route.

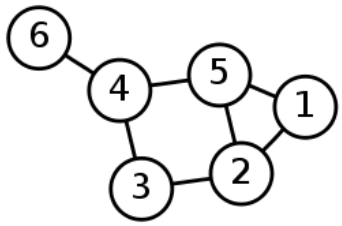
Step 2 - Information gathering:

To address the problem of the delivery system from a software engineering point of view, we are first going to raise some important concepts that will be relevant when implementing the project.

We have to find data structures and algorithms which allow us to manage the operations that can be made in a map and store all its information. Also algorithms that can find that path between two points in the city and an algorithm that helps us find the minimal spanning tree on the south of Santiago de Cali. In order to present options to develop the project we need to establish some definitions and concepts:

Definitions

Graph theory:

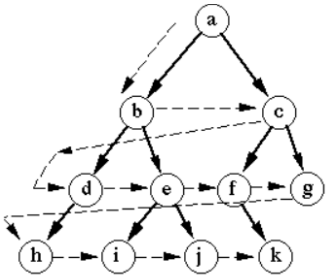


Graph: Graphs are discrete structures consisting of vertices and edges that connect those vertices together. Therefore a graph G consists of two parts: 1) A set V = V (G) whose elements are called vertices, points or nodes of G. 2) A set E = E(G) of pairs of distinct vertices called edges from G. (GeeksforGeeks, s. f.)

Vertex: The vertices are the basic units of the graph. Intersections are also sometimes referred to as vertices or nodes. Each node/vertex can be labeled or unlabeled

Edge: Are the boundaries drawn or used to connect two graph nodes. Edges can connect two nodes in any way. Sometimes edges are also called arcs.

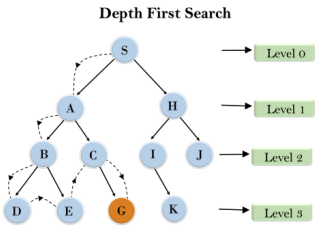
Graph traversal algorithms



Breadth-first search

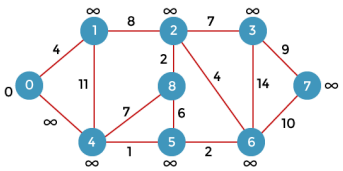
Breadth First Search: BFS is an algorithm for exploring a tree or graph.This algorithm starts from a root node and then visits all the nodes in the base order. That means after the root, that is, through all the direct children of the root. Having passed through all the

roots directed to his sons, he moves to his sons, and so on to the others. To implement BFS we use a queue.

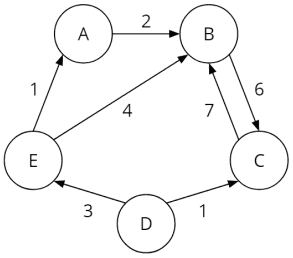


Depth First Search: The depth-first search (DFS) algorithm starts with the initial node of graph G and goes deeper until we find the goal node or the node with no children. Because of the recursive nature, stack data structure can be used to implement the DFS algorithm. The process of implementing the DFS is similar to the BFS algorithm. (*DFS Algorithm - javatpoint*, s. f.)

Shortest path algorithms

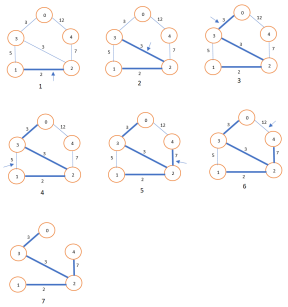


Dijkstra Algorithm: This algorithm is implemented in a graph and allows us to find the shortest path between any two vertices of a graph. Time Complexity of Dijkstra's Algorithm is O (V 2) but using min-priority queue it drops down to O (V + E l o g V) .

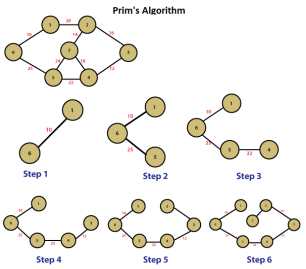


Floyd Warshall Algorithm: Like Dijkstra's algorithm this is implemented in a Graph and finds the shortest path. But, the difference between these two algorithms is that the Floyd Warshall Algorithm finds the shortest distance between every pair of vertices in a given edge-weighted directed graph. This change makes the time complexity of these algorithms rise to O(n^3) .

Minimum spanning trees algorithms

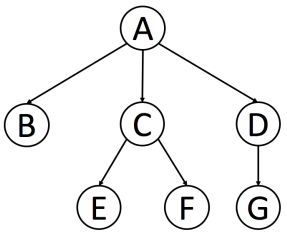


Kruskal Algorithm: The Kruskal algorithm is an algorithm implemented in a graph that returns a forest with minimum spanning trees of that graph. In order to develop these procedures Kruskal's Algorithm begins with taking the shortest edge in a network and step by step it continues to add edges to the tree. The time complexity in Kruskal's algorithm is O(E log V), where V is the number of vertices.



Prim Algorithm: The Prim algorithm is similar to the Kruskal Algorithm in the fact that both generate minimum spanning trees but the Prim algorithm just generates one. On the other hand, the

Prim algorithm begins with selecting one vertex in the graph and starts adding vertex to the spanning tree. The time complexity of this algorithm is O(V^2),



N-ary trees: Another way to approach the problem of a map in a software environment could be the idea of using an N-ary tree. Every crucial point in the city would be a vertex and the time of arriving from one point to another would be an edge. Nevertheless this is not a very good idea because it doesn't allow cycles between vertices and it doesn't have the variety of algorithms that the graph theory has.

Step 3 -Search for creative solutions:

In order to search for creative solutions we have decided to use the method of brainstorming, where we are going to propose creative possible solutions taking into account the information presented previously. After that are going to filter each one of them according to certain criterias in order to fulfill correctly the requirements of the system.

Proposed solutions:

As the problem is so extense we have decided to divide the selection of the ideas according to certain modules where we can later mix between each other that together provide an answer to the complete program.

1) **Modeling of the map**
    a) Use an API from google maps
    b) Use graph theory
    c) Use an N-ary tree

2) **Graphic Interphase**
    a) Use Swing framework
    b) Use JavaFx framework
    c) Use AWT framework

3) **Gathering of data**
    a) Collect data systematically from google maps
    b) Manual collection of data

4) **Functionality of the shortest path**
    a) Dijkstra Algorithm
    b) Floyd Warshall Algorithm

5) **Functionality of the minimum spanning tree**
    a) Kruskal Algorithm
    b) Prim Algorithm

Step 4 - Transition from Ideas to Preliminary Designs:

In this section we are going to discard some ideas that are not viable to the final solution of the system. These are the ideas that we are going to discard:

Modeling of the map

a. **Use an API from google maps** : We don't have enough knowledge to make that type of implementation.

Graphic Interphase

b. **Use AWT framework:** As it is so old it has so many disadvantages like Platform Dependency, Heavyweight, does not support a pluggable look and feel and it's not currently in use.

Gathering of data

c. **Collect data systematically from google maps:** There aren't sources to collect that kind of data without paying a subscription. Also there isn't enough information to establish that kind of connection with a java project.

Possible solutions:

Modeling of the map

a) **Use graph theory:** This is one of the most known ways to describe a relationship between points. Here we can implement the places in the city as nodes in a graph and the path would be the edges of the graph. The time that it takes to make that travel would be the equivalent to the weight of the edges.

b) **Use an N-ary tree**: It is very similar to the graph theory, in fact, an n-ary tree is a type of graph but it doesn't allow cycles. This can be a very difficult situation because the city maps it is very common to have cycles between two points. Nevertheless, we can use this data structure to describe some of the routes that can be taken in a graph.

Graphic Interphase

a) **Use Swing framework:** Java Swing is one of the most known frameworks to develop graphic interfaces in java. It was created in 1997, and it has a lot of functionalities and components, but one of its biggest disadvantages is that it doesn't have very modern functionalities in comparison to its biggest competitor: JavaFX.

b) **Use JavaFx framework :** It is one of the frameworks that is at the forefront when it comes to developing user interfaces with java. It was created in 2008 and since that time it has taken all the industry of graphic interfaces in java by its modern aspect and rich functionalities. Additionally, we have a lot of experience working with this framework in class.

Gathering of data

a) **Manual collection of data:** Doing this process takes a long time, however as the size of our project is not too big we can do this data collection manually without any problem.

Functionality of the shortest path

a) **Dijkstra Algorithm:** As described in the section of the definitions, Dijkstra Algorithm is an algorithm that finds the shortest path between two nodes, and its complexity is O(n^2), where "n" is the number of vertices in the graph. In the project as we need to make single queries this algorithm can be feasible.

b) **Floyd Warshall Algorithm:** This algorithm also finds the shortest path in a graph but the difference with Dijsktra is that this algorithm makes this procedure with every pair of nodes that are in the graph. This idea can be a little bit misleading as we are trying to make a program where the user makes single queries to the program.

Functionality of the minimum spanning tree

a) **Kruskal Algorithm:** As described in the section of the definitions, the Kruskal Algorithm returns a forest with minimum spanning trees of that graph, beginning with taking the shortest edge in a network and step by step it continues to add edges to the tree.

b) **Prim Algorithm:** It is very similar to the Kruskal Algorithm in the sense that both of them return a minimum spanning tree but the difference is that the Prim Algorithm returns a single minimum spanning tree generated from a root node.

Step 5 - Evaluation and Selection of the Best Solution:

In order to find the best solution for each section it is mandatory to establish an evaluation system that allows us to determine which is the correct idea that we should implement in our solution. This evaluation will be done according to certain criterias described in each section.

Modeling of the map

Criterion A: Can model the relationships between the points in the city and the roads that connect them including the time that it takes to make that travel.

- **[1] None**

- **[2] Some**
- **[3] Mostly**
- **[4] Perfectly**

Criterion B: Support of a variety of algorithms.
- **[1] None**
- **[2] Some**
- **[3] A lot**

Criterion C: Allows cycles
- **[1] No**
- **[2] Yes**

	Criterion A	**Criterion B**	**Criterion C**	**Total**
Alternative 1: Use graph theory	**4**	**3**	**2**	**9**
Alternative 2: Use N-ary trees	**3**	**2**	**1**	**6**

We can see that the winner here is the Graph Theory implementation.

Graphic Interphase

Criterion A: Quantity of functionalities and components that it has:
- **[1] None**
- **[2] Some**
- **[3] A lot**

Criterion B: Is this framework used nowadays
- **[1] None**
- **[2] Some**
- **[3] A lot**

Criterion C: Previous knowledge using this framework
- **[1] No**
- **[2] Yes**

	Criterion A	Criterion B	Criterion C	Total
Alternative 1: Use Java Swing	**3**	**2**	**1**	**6**
Alternative 2: Use JavaFX	**2**	**3**	**2**	**7**

We can see that the winner is the implementation with JavaFX

Gathering of data:
As there is only one option available in this section we don't have more options than to follow this idea.

Functionality of the shortest path:

Criterion A: Can make the shortest path between two points

- **[1] No**
- **[2] Yes**

Criterion B: Time complexity

- **[1] O (n^3)**
- **[2] O (n^2)**
- **[3] O(n log n)**
- **[4] O(log n)**
- **[5] O(n)**

Criterion C: Can make the operation between two nodes individually?

- **[1] No**
- **[2] Yes**

	Criterion A	Criterion B	Criterion C	Total
Alternative 1: Dijkstra Algorithm	**2**	**2**	**2**	**6**
Alternative 2: Floyd Warshall Algorithm	**2**	**1**	**1**	**4**

Here we can see that the Dijkstra Algorithm is the winner of the section of the algorithm implemented to have the shortest path between two nodes.

Functionality of the minimum spanning tree

Criterion A: Can make the minimum spanning tree of a graph

- **[1] No**
- **[2] Yes**

Criterion B: Time complexity

- **[1] O (n^3)**
- **[2] O (n^2)**
- **[3] O(n log n)**
- **[4] O(log n)**
- **[5] O(n)**

Criterion C: Can the algorithm begin in one node specifically?

- **[1] No**
- **[2] Yes**

	Criterion A	**Criterion B**	**Criterion C**	**Total**
Alternative 1: Kruskal Algorithm	**2**	**3**	**1**	**6**
Alternative 2: Prim Algorithm	**2**	**3**	**2**	**7**

Here we can see that the winner of the section of the minimum spanning tree is the Prim algorithm.

Step 6 - Preparation of reports and specifications:

Since we have a better understanding of the structures that we are going to use and the scope that our solution can take from these, we will be able to better define the requirements of the delivery system that we are developing.

Functional requirements:

R1: Graphic Interface: The system must have a graphic interface according to the graphic aesthetic of the brand “Rappiando”, using the logo typography and its representative colors. The system must show the map of the south of Cali with the most busiest points of the city and its roads.

R2: Menu selection: The program must have a menu that presents two options to select: Show the shortest path between two points and show the route that the delivery needs to follow in order to deliver ballots throughout the south of Cali.

R3: Shortest path between two points: After selecting the option of showing the shortest path between two nodes, the application will ask for the points of the city that the user wants to make the query. After that, the application will show the shortest path visually showing which are the roads that the delivery needs to take. Also, the program must show which is the estimated time that the trip will take.

R4: Deliver ballots through all the city: After selecting the option of showing the route to deliver ballots through all the city, the system must calculate this route and show it graphically. Also the program must show which is the estimated time that the trip will take.

References:

GeeksforGeeks. (s. f.). Graph Data Structure And Algorithms.
https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
Edges and Vertices of Graph. (s. f.).
https://www.tutorialspoint.com/edges-and-vertices-of-graph
DFS Algorithm - javatpoint. (s. f.). www.javatpoint.com.
https://www.javatpoint.com/depth-first-search-algorithm