The engineering method

Contenido

[Context of the problem 2](#_Toc87649283)

[Identifying the problem 2](#_Toc87649284)

[Identification of needs and symptoms 2](#_Toc87649285)

[Definition of the problem 2](#_Toc87649286)

[Data collection 2](#_Toc87649287)

[Graphs 2](#_Toc87649288)

[Map of ICESI 3](#_Toc87649289)

[Dijkstra’s algorithm 3](#_Toc87649290)

[Greedy algorithms 3](#_Toc87649291)

[Breath-first search 3](#_Toc87649292)

[Search for creative solutions 4](#_Toc87649293)

[Transition of the formulation of ideas to preliminary design ideas 5](#_Toc87649294)

[Idea discard 5](#_Toc87649295)

[Alternative 2: 5](#_Toc87649296)

[Alternative 3: 5](#_Toc87649297)

[Alternative 5: 5](#_Toc87649298)

[Alternative 6: 5](#_Toc87649299)

[Evaluation and selection of the best solution 5](#_Toc87649300)

[Criteria 5](#_Toc87649301)

[Evaluation 6](#_Toc87649302)

[Selection 6](#_Toc87649303)

[Preparation of reports and specifications 6](#_Toc87649304)

[Specifications 6](#_Toc87649305)

[Requirements 7](#_Toc87649306)

[Nonfunctional requirements 7](#_Toc87649307)

[Considerations 7](#_Toc87649308)

[Class diagram 7](#_Toc87649309)

[Implementation of the design 7](#_Toc87649310)

[Biography 7](#_Toc87649311)

# Context of the problem

In this November, our university (ICESI) has opened a partial re opening of the campus. Now, because of this, students from 3 semester are just getting to know their university, and often get lost. In addition, the next semester has been announced to be fully in campus, which means students from 4 semester that barely know this environment are going to be expected to travers the university without problems. Because of this, we have decided to create an application that helps this new commers get used to the campus, by showing them paths from one place to another.

# Identifying the problem

## Identification of needs and symptoms

* The program must be able to handle a graph that represents the university
* The program must be capable of identifying the shorts path between from 1 place to another
* The program must be capable of identifying multiple paths from one place to another
* The program must allow the user to select a starting point and a destination
* The program must show
  + The distance of the path
  + The name of landmarks of the campus
  + The name of the buildings
  + The name of the auditoriums
* At least to graphs must be used
* The program should have at least 50 nodes

## Definition of the problem

The development of a graph that represents a map of ICESI´s campus. This map must tell the user the shortest path between 2 places of interest.

# Data collection

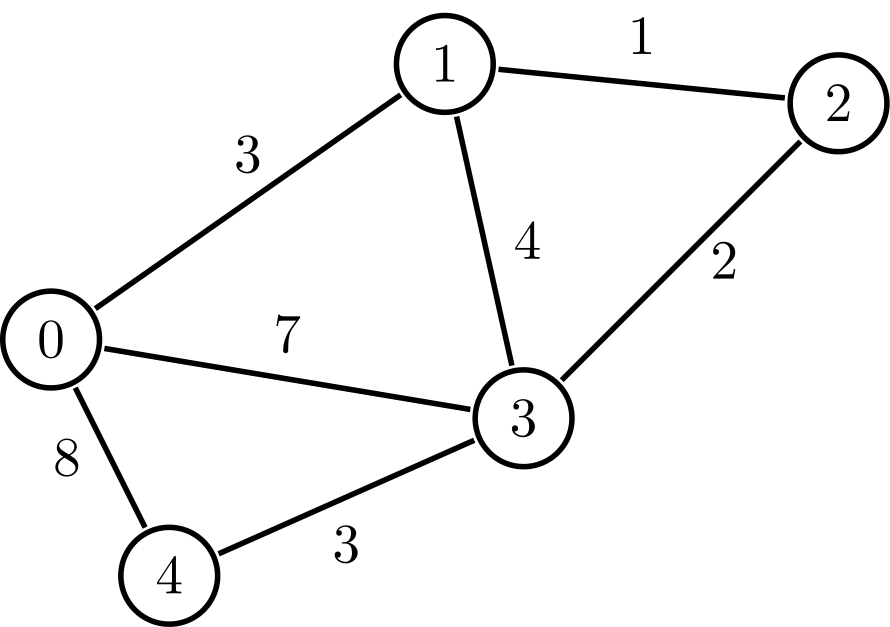
## Graphs

A graph is a non-linear data structure that is composed of nodes (or vertices) and edges (lines that connect the nodes), so we can represent them as G= (V, E). One example of a graph is:

Imagen que contiene Diagrama

Descripción generada automáticamente

In our program, we will use this type of graph, (called a simple graph,) and a graph with weighted (simple weighted graph) edges that would represent the distance between one node and the other, for example:



(Cormen, Leiserson, Rivest, & Stein, 2009)

## Map of ICESI

For this, we had to talk with Sandra Piedad Pineda, Architect coordinator of projects “Planta Física y Servicios Generales” and explain why we needed the schematics of the university, and then get permission from out teacher to get said documents. This document will be used to create an accurate map of the university using graphs. Sadly, as we had to go through such a process to get the schematics, this will not be present in this project, so you’ll just have to believe us that we used them.

## Dijkstra’s algorithm

This algorithm solves the single source shortest paths problem on a weighted graph. This algorithm is greedy and can’t function with negative weights (which we will not use because there are no such dark magic as negative distances, that we know of, in our university). It should also be mentioned that this algorithm has a time complexity of O (V2), V being the number of vertices the graph has. However, this time complexity can be reduced to O (V log V + E) (E being the number of edges of the graph), but we will not bother with this.

(Cormen, Leiserson, Rivest, & Stein, 2009)

## Greedy algorithms

This kind of algorithms function by choosing the most optimal path in each step.

## Breath-first search

This algorithm receives a graph and a vertex to start from, then it explores the graph to find every vertex reachable from this source. Because of this, it produces a “Breadth-first tree”, with the root being the starting vertex that it was given, and the leaves are the vertices that are reachable. To keep track of it´s progress, it colors the nodes, all nodes start out as white, which means they haven’t been discovered, when they are discovered, they turn gray, which means, if the algorithms stubbles upon said gray vertex, it will not be added to the tree. Lastly, if all the edges of a vertex have been completely explored, the vertex is colored black. Here is an example:

Diagrama, Esquemático

Descripción generada automáticamente

(Cormen, Leiserson, Rivest, & Stein, 2009)

# Search for creative solutions

* Alternative 1: Get the location from the user’s device, and use it has a starting node, from here, the user can decide where to go, this should have a way of to toggle the distances or not, as the user might not always care for how optimal the path is, but would just like to see what options they have
* Alternative 2: Drawing a map using graphs, this way, the user can choose the vertex they’re closest to as the starting point, then they can choose where they want to go, this should have a way to toggle the distances or not, as the user might not always care for how optimal the path is, but would just like to see what options they have
* Alternative 3: Use satellite images to guide the user through the best path the program has found this should have a way to toggle the distances or not, as the user might not always care for how optimal the path is, but would just like to see what options they have
* Alternative 4: Use the photo of the university’s schematics we were given as a map, and overlay it with the graph, this way the user would have a very complete view of the path and university this should have a way to toggle the distances or not, as the user might not always care for how optimal the path is, but would just like to see what options they have
* Alternative 5: Just create a graph, each vertex would have a name, which would be chosen depending on if said vertex is a corridor or building this should have a way to toggle the distances or not, as the user might not always care for how optimal the path is, but would just like to see what options they have
* Alternative 6: Cut most of the paths from the map, so we can save resources and have less calculations. This way, we could use the vertex only as the university’s buildings, simplifying the map a lot, given that there is only a hand full of buildings.

# Transition of the formulation of ideas to preliminary design ideas

## Idea discard

Alternative 1: We will discard this idea because at the time of writing we don’t know how to get information from the user’s device, having said this, we could probably figure it out, but we don’t have the time.

Alternative 4: As we understand, we have permission to use ICESI’s schematics, but not to publish them, so we have opted to discard this idea.

## Alternative 2:

* Has a representation of a map of ICESI
* Does offer a way for the user to get the desired path
* Offers the shortest path
* Offers different paths
* Has a way to locate the user on the map

## Alternative 3:

* Has a representation of a map of ICESI, although it might not be the clearest
* Offers the shortest path
* Offers different paths
* Does not offer a way to locate the user on the map

## Alternative 5:

* Uses a graph as the map, probably very confusing for the user
* Names the vertices
* Offers the shortest path
* Offers different paths

## Alternative 6:

* Offers a simpler approach to the problem, won’t be as complete as other alternatives, but much easier for the user to use
* Offers the shortest path
* Offers different paths
* Names the vertices according to ICESI’s building

# Evaluation and selection of the best solution

## Criteria

* Criteria A: Uses graphs
  + [2] Yes
  + [1] No
* Criteria B: How easy would it be for the user to use
  + [3] easy
  + [2] requires some explanation
  + [1] Very complex
* Criteria C: Offers the best path
  + [3] yes
  + [2] most of the time
  + [1] no
* Criteria D: Would have at least 50 vertices
  + [2] yes
  + [1] no
* Criteria E: Is a complete representation of a map of the university
  + [3] yes
  + [2] partial
  + [1] no

## Evaluation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Criteria A | Criteria B | Criteria C | Criteria D | Criteria E | Total |
| Alternative 2: graph representing map | Yes  2 | Requires some explanation  2 | Yes  3 | Yes  3 | Yes  3 | 13 |
| Alternative 3:  Satellite image | Yes  2 | Requires some explanation  2 | Yes  3 | Yes  3 | Partial  2 | 12 |
| Alternative 5:  Uses the graph itself as a map | Yes  2 | Complex  1 | Yes  3 | Yes  3 | Yes  3 | 12 |
| Alternative 6: Simple map | Yes  2 | easy  3 | Most of the time  2 | No  1 | Partial  2 | 10 |

## Selection

Using the evaluation, we can see that the second alternative is the best suited for the problem. Having said this, alternative 3 was close, but we decided against using it, as a satellite image might not show all the paths in the university.

# Preparation of reports and specifications

## Specifications

Problem: Help students travers ICESI’s campus

Input: A map of ICESI, with at least 50 vertices

Output: The shortest path between too vertices the user decides

## Requirements

* The program must have at least 50 vertices
* The program must work with at least 2 graphs
* The program must give the shortest path between two places
* The program must give multiple paths between two places
* The program must be able to represent the campus using graphs

## Nonfunctional requirements

* The user interface must be easy to understand
* The program must be organized
* The use of Floyd-Warshall
* The use of DFS
* The map displayed for the user must be understandable

## Considerations

Thigs that should be considered

* The searches for the shortest path might freeze the user interface, we could use threads to avoid this
* We could just use Floyd-Warshall once and determine all the best paths, save this using serialization, and just use the resulting matrix to give the user the best path every time with out the need for another calculation
  + Although, given than the graph can be quite big, the first instance of Floyd-Warshall might take a while
* We should switch from the weighted graph to the non-weighted with out the user noticing
* We should make the map of the university somewhat interactable, so the user has a better time using it

## Class diagram

We haven’t done this yet

# Implementation of the design

We haven’t done this yet

# Biography

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, Third edition .* Massachusetts : MIT Press.