***ENGINEERING METHOD***

***PARTICIPANTS***

***SANTIAGO PRADO LARRARTE***

***JEISON LASPRILLA***

***JUAN DIEGO LORA***

***TEACHER JEISON***

page1image1701952

***INTEGRATIVE TASK 2 ALGORITHMS AND DATA STRUCTURES***

***2022-11***

***ENGINEERING METHOD***

page2image1656704

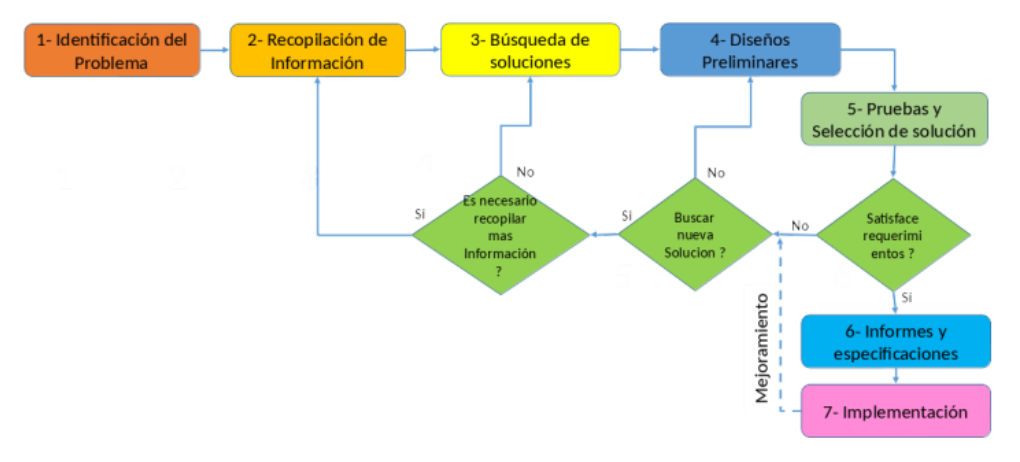
***PROBLEM CONTEXT:***

The mayor of the city of Cali, wants to know if the flow of water through the sewers of the city has been making on the most efficient way. That’s why he needs the development of a software tool capable of clearly and easily simulating the operation of the flow of water in the city of Cali.

***SOLUTION DEVELOPMENT:***

To solve the previous situation, the Engineering Method was chosen to develop the solution following a systematic approach and in accordance with the problematic situation posed. Based on the description of the Engineering Method in the book “Introduction to Engineering” by Paul Wright.

The following flow chart was defined, the steps of which we will follow in the development of the solution.



***IDENTIFICATION OF THE PROBLEM:***

In this section, it is important to have a good definition of what is the solution to the problem that is being addressed.

Identification of needs:

* The tools must use data structures (queues, ArrayList, Hash table)
* The tool must provide an easy to understand and above all efficient solution.
* The tool must be able to use the solutions proposed in the types of graphs presented

***INFORMATION GATHERING:***

//Introduccion grafo para trabajar

In addition, In this section the concepts that are essential for understanding the solution of the specific problem are specified.

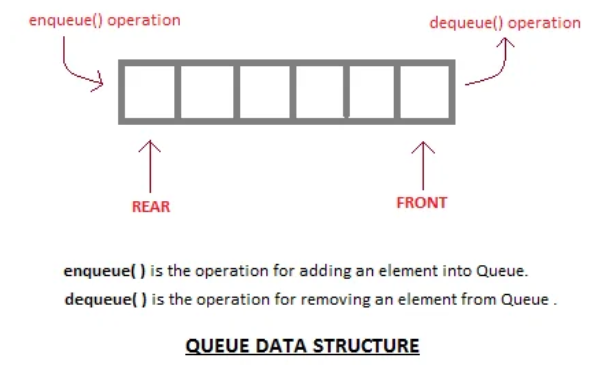
*Algorithm:*An ordered set of systematic operations that allows making a calculation and finding the solution to a type of problem.

*Data structures:*

In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently. Different types of data structures are suitable for different types of applications, and some are highly specialized for specific tasks.

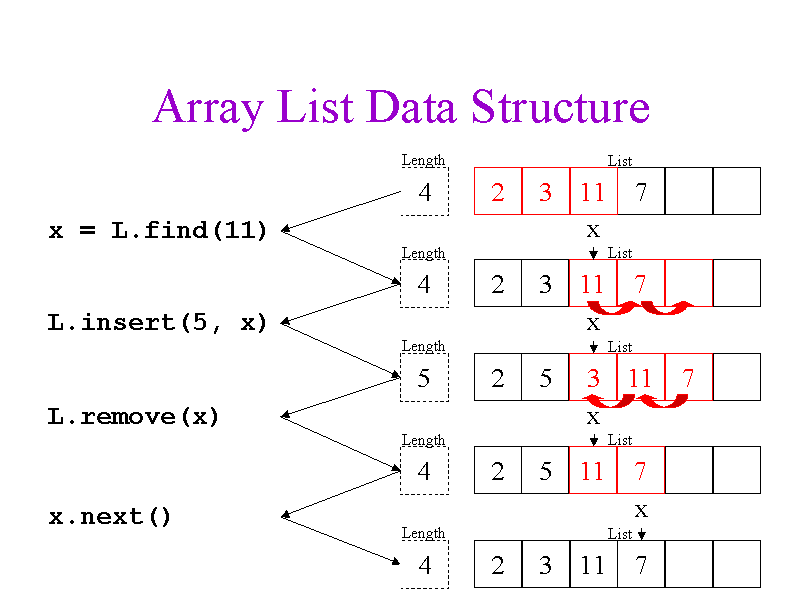
*Queue:*

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First in First Out



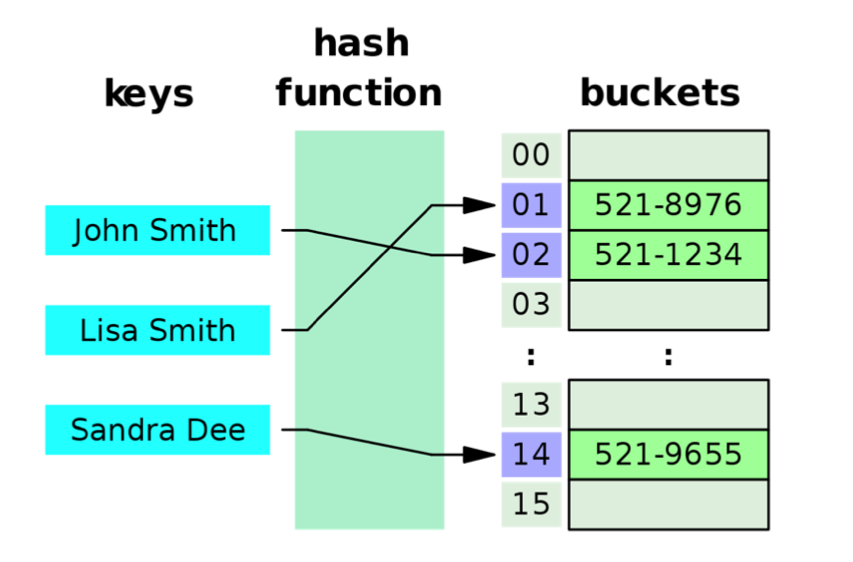
*Array List:*

Array List are a linear structure that uses an Object class array to store the objects



*Hash table:*

In computing, a hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found. During lookup, the key is hashed and the resulting hash indicates where the corresponding value is stored.



***SEARCH FOR CREATIVE SOLUTIONS TIME COMPLEXITY:***

In this section is going to be introduced the ideas that can be optimal to solve the problem that was proposed.

*Proposal 1:*

The first desition we need to make, is all related with about the way to save and move the data, the idea is to take the principal components of the problem ( Adjacents, Edges, Vertex, etc) for example, when we visited a vertex ( pipe ) we need to know the adjacent of that vertex to know which is the next movement. When the vertex is visited, the best way to represent the order is with a queue data structure, we will have access quickly to the last vertex that was visited ( In constant time O(1)).

In the case of Kruskal solution, the output should have the sum of all the weights of the edges that was used to visit al vertex’s.

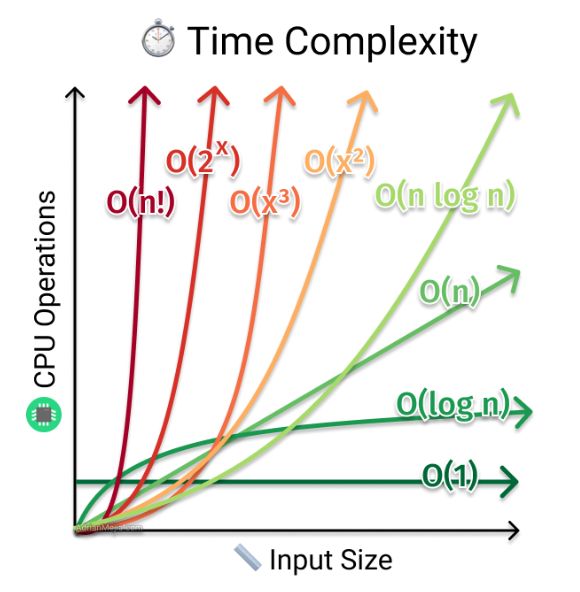
*Proposal 2:*

This second proposal thinks about working all the data structures as simple arrays, in other words, simplify the way that we save data, in this case we will have all the main components in arrays, that can easily to know positions of vertex and edges. However, this have a problem, the time complexity. In the case of access to an index is constant O(1), but for the methods of insert and delete, the time in both cases are O(n).

***TRANSITION FROM FORMULATION OF IDEAS TO PRELIMINARY DESIGNS:***

In this case it is considered that the best option is the proposal 1, however it is important to see why the other proposal is not that efficient.  
As we can see, the principal difference between the proposals are the time complexity if we think that the algorithm will be the same, just changing the data structure that we are going to use, so in that case the proposal 2 is worst in the fact that in storage of the components of the problem generally the complexity that we are going to work with is O(n), linear so it the depends of the size of the input, In our case we have 60 vertex and 60 edges. We don't want that, as we mentioned before we need to think about the problem in general, so in this case the proposal gives us the major part of the algorithm in constant time when we talk about the data structures that we are going to use.

The different types of complexity can be seen in the next graphic:



As we can see in the graphic the best complexity is O(1) in other words the constant line.

***EVALUATION AND SELECTIONS OF THE BEST SOLUTION TIME COMPLEXITY:***

Is time to select the final choice, so it is important to define evaluation criteria that let us prove that the solution we select is optimal (for us) for this problem, in particular thinking about it generally.  
Criteria A: Solve the problem correctly

- [2] Yes

-  [1] No

Criteria B: The data structures that use let storage data naturally (it means to don't need to create another structure to save in)

-  [3] Yes

-  [2] Sometimes

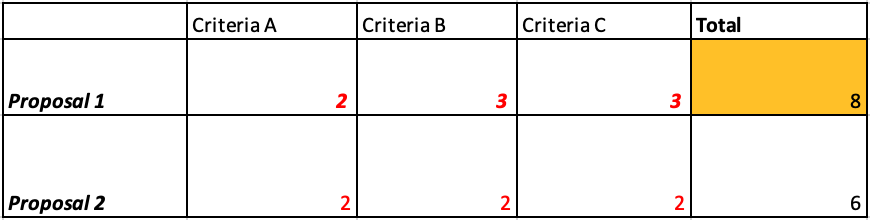
-  [1] No

Criteria C: The data structures that use the most are optimal (It means that for example the most are O(1))

-  [3] Yes

-  [2] More or less

-  [1] no



***SEARCH FOR CREATIVE SOLUTIONS ALGORITHMS:***

*Proposal 1 (Algorithm Ford Fulkerson):*

This algorithm basically propose the search of ways to arrive the goal Node, with the objective of reach the maximum flow. This algorithm will be useful in problems of networks or problems that the flow is important. In this case we don’t think this algorithm will be useful to solve our problem.

*Proposal 2 (Algorithm Kruskal):*

Kruskal's algorithm finds a [minimum spanning forest](https://en.wikipedia.org/wiki/Minimum_spanning_tree) of an undirected [edge-weighted graph](https://en.wikipedia.org/wiki/Weighted_graph). If the graph is [connected](https://en.wikipedia.org/wiki/Connectivity_(graph_theory)), it finds a [minimum spanning tree](https://en.wikipedia.org/wiki/Minimum_spanning_tree). (A minimum spanning tree of a connected graph is a subset of the [edges](https://en.wikipedia.org/wiki/Edge_(graph_theory)) that forms a tree that includes every [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)), where the sum of the [weights](https://en.wikipedia.org/wiki/Weighted_graph) of all the edges in the tree is minimized. For a disconnected graph, a minimum spanning forest is composed of a minimum spanning tree for each [connected component](https://en.wikipedia.org/wiki/Connected_component_(graph_theory)).) This algorithm propose to search the edges of the graph with lowest weight, with the condition that all the vertex must be visited, in addition, works to problems that has relation with distance.

*Proposal 3 (BFS):*

In the BFS algorithm, nodes that are visited more than once are immediately removed from the queue, which saves time by shortening possibilities.

The FIFO standard is followed, the first in, the first out, that is, a queue BFS takes up more memory space than DFS

*Proposal 4 (DFS):*

In the DFS algorithm, all the nodes are traversed, and those already visited are eliminated at the end of the process.

The LIFO principle is followed, first in, last out, that is, one row DFS occupies considerably less memory space than BFS

*Proposal 5 (Dijkstra’s algorithm):*

Dijkstra's algorithm is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem) between [nodes](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)), which may represent, for example, [road networks](https://en.wikipedia.org/wiki/Road_network). It was conceived by [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist) [Edsger W. Dijkstra](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) in 1956 and published three years later.

*Proposal 6(Floyd-Warshall algorithm):*

In [computer science](https://en.wikipedia.org/wiki/Computer_science), the Floyd–Warshall algorithm (also known as Floyd's algorithm, the Roy–Warshall algorithm, the Roy–Floyd algorithm, or the WFI algorithm) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem) in a directed [weighted graph](https://en.wikipedia.org/wiki/Weighted_graph) with positive or negative edge weights (but with no negative cycles).

***EVALUATION AND SELECTIONS OF THE BEST SOLUTION:***

Is time to select the final choice, so it is important to define evaluation criteria that let us prove that the solution we select is optimal (for us) for this problem, in particular thinking about it generally.

Criteria A: Solve the problem correctly

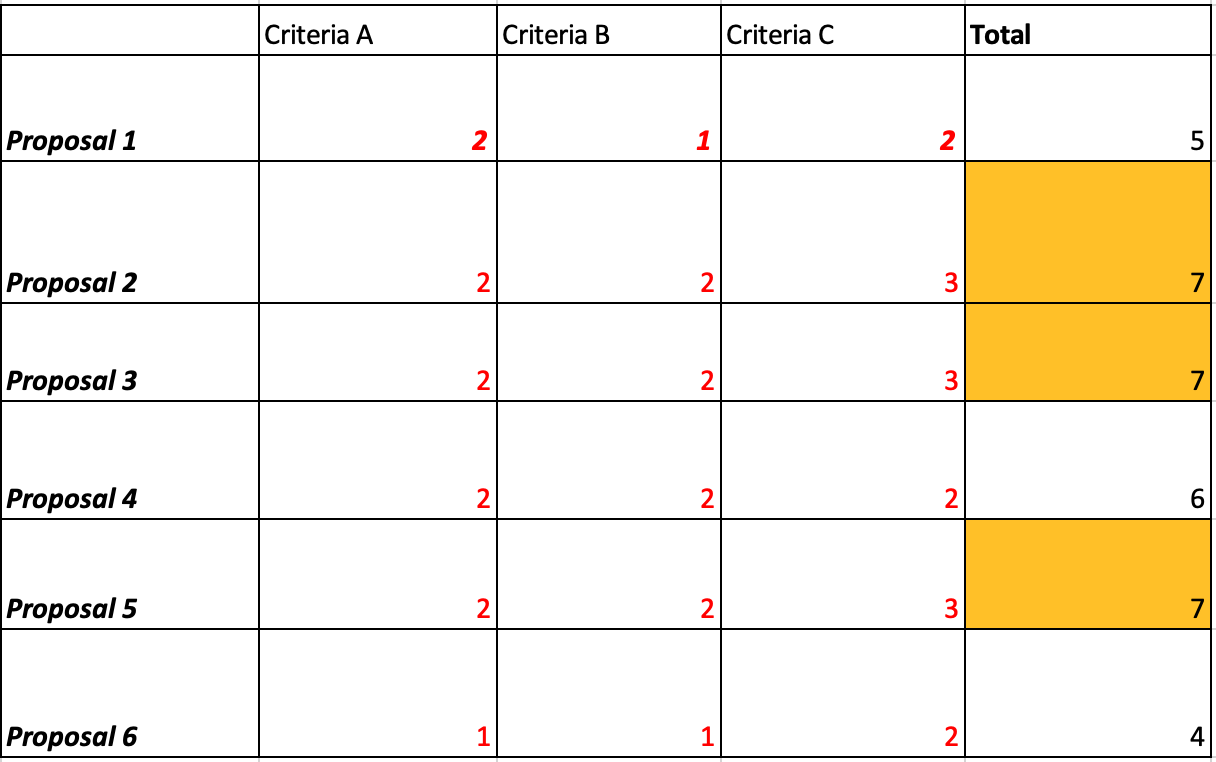
* [2] Yes
* [1] No

Criteria B: Give us the expected result

* [2] Yes
* [1] No

Criteria C: Helps with the problem of the city of Cali

* [3] Yes
* [2] Sometimes
* [1] No



***REPORT PREPARATION AND SPECIFICATIONS:***

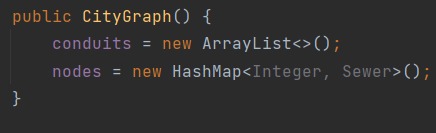
1. Don't forget the limitations of the solution, so it is important to explain which are the limitations of the solution or considerations for the implementation.  
   In this case is needed show how will be the solution that is proposed  
   Considerations:
   1. For the optimal work in the data structures it might be all the data in the correct format
   2. All the data structures are might be defined properly
   3. To create properly the objects all the data that is given might be correct
   4. To create edges, more than one vertex mut exist
   5. All edges must have a weight

***DESIGN IMPLEMENT:***

List of task to implements:

* Create graph
* Add Sewer
* Add conduit
* BFS
* Kruskal
* Dijkstra

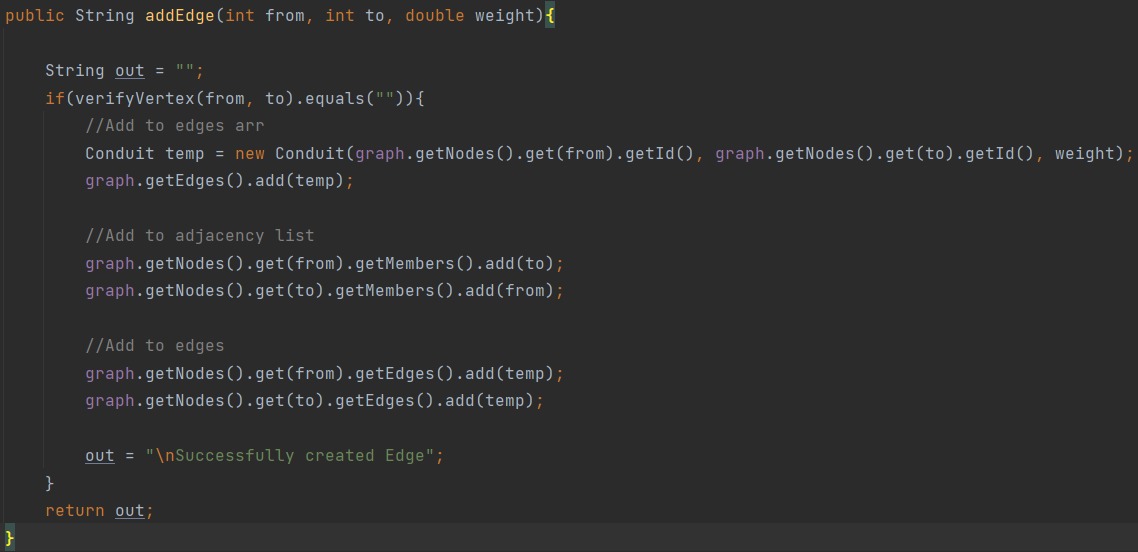
1. Create Graph:



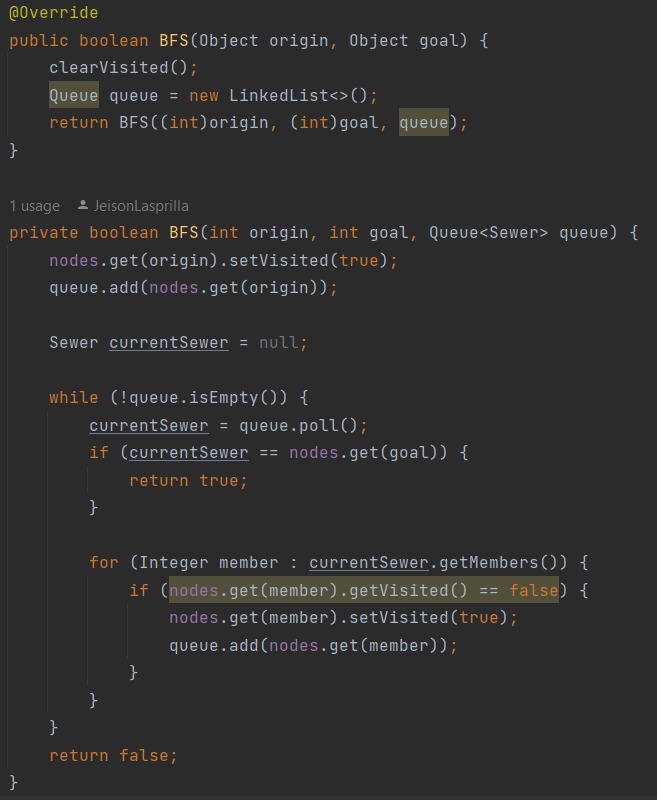
1. Add Sewer:



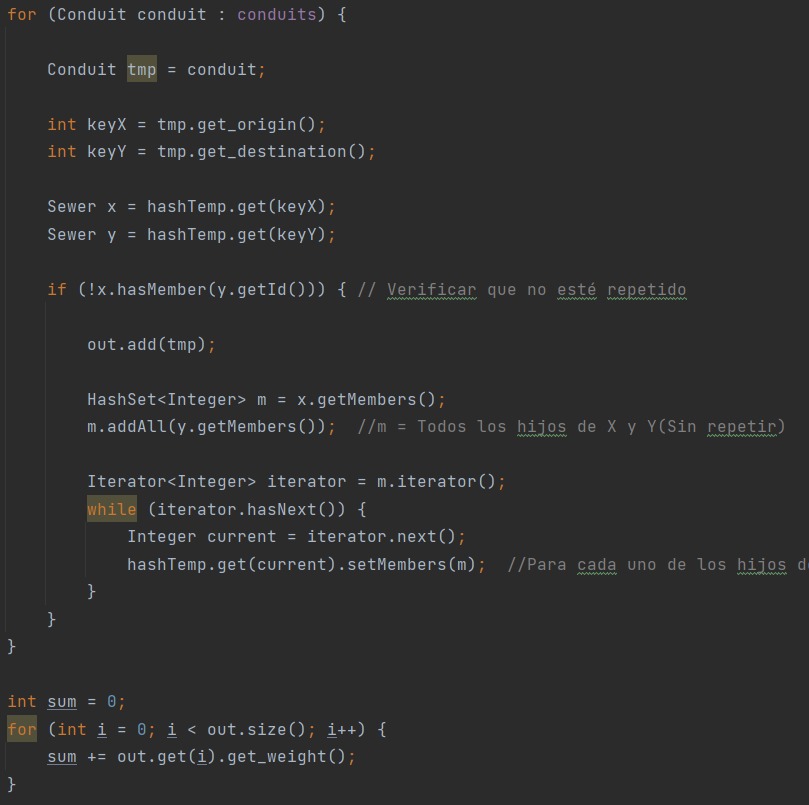
1. Add Conduit:



1. BFS:



1. Kruskal



1. Dijkstra:

