**Engineering Method**

**(problem 1 – 1112 Mice and Maze)**

**Problem definition:**

Judge: Online Judge

<https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=3553>

link: <https://drive.google.com/file/d/12zpE-cB_zDxe5ajTpYpyCBEaEufSy3M9/view?usp=sharing>

**Phase 1: Problem identification**

The question that defines this problem is: How many mice manage to leave the maze in time?

**Phase 2: Collection of the necessary information**

All the information necessary to understand the problem is provided by the statement, the entries are specific and it is the only thing that will be obtained.

**Phase 3: Search for creative solutions**

To solve this problem, an algorithm is needed to find the path of the lowest weight of the labyrinth, from each of the cells. For this you can use algorithms known as:

* Floyd-Warshall
* Dijkstra

Another option may be to look for all possible paths until one meets the time limit and if it is not possible to discard it

**Phase 4: Transition of the formulation of ideas to preliminary designs**

The idea of ​​looking for all possible paths is likely to function, but the complexity is the highest compared to the other options, therefore it is ruled out.

On the other hand, the Floyd Warshall option has a complexity of V^3 and the minimum weight of all roads is obtained

Also, the Dijkstra algorithm has a temporal complexity of V to 2 with a single path, so this algorithm must be repeated V times, therefore, the serious complexity of V^3

**Phase 5: Evaluation and selection of the best solution**

To solve this problem, Floyd Warshall was selected, since it is easier to implement and at the end of the algorithm the answer is already obtained, while for the Dijkstra algorithm, it is necessary to repeat the algorithm to verify if each mouse manages to exit

**Phase 6: Preparation of reports and specifications**

**Functional requirements**

|  |  |
| --- | --- |
| **Name** | **R.# 1. Find the number of mice that manage to get out of the maze** |
| **Summary** | Find the number of mice that leave the maze in the given time |
| **Input** | |
| • Number of cages  • Departure location  • Maximum time determined to exit  • Connection between cages  • Time it takes to go from one cage to another  Results | |
| **Output** | |
| A number that indicates the number of mice that manage to get out of the maze | |

|  |  |
| --- | --- |
| **Name** | **R.# 2. Visualize the maze** |
| **Summary** | Summary Show the shape of the maze |
| **Inputs** | |
| • Number of cages  • Departure location  • Maximum time determined to exit  • Connection between cages  • Time it takes to go from one cage to another | |
| **Output** | |
| A screen maze display | |

|  |  |
| --- | --- |
| **Nombre** | **R.# 3. Visualize the cages of the mice that manage to leave the labyrinth** |
| **Resumen** | It allows to visualize the mouse cages that manage to leave the labyrinth |
| **Input** | |
| None | |
| **Output** | |
| A visualization of the maze on the screen with the cages of the mice that managed to leave a different color | |

**Engineering Method**

**(problem 2 – 11631 Dark roads)**

**Problem definition:**

Judge: Online <https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=2678>

link: <https://drive.google.com/file/d/1gN7QIJUoG5PKY4rR4p72jPhRXBAQ62Ch/view?usp=sharing>

**Phase 1: Problem identification**

The main problem is knowing the maximum amount of money that the government can save without sacrificing the security of citizens

**Phase 2: Collection of the necessary information**

All the information necessary to understand the problem is provided by the statement, the entries are specific and it is the only thing that will be obtained.

**Phase 3: Search for creative solutions**

To solve this problem, we must implement an algorithm that allows us to simulate the map of the city and find the roads that have the least cost and it is possible to reach all places. For this you can use algorithms that already exist as Prim or Kruskal.

**Phase 4: Transition of the formulation of ideas to preliminary designs**

The Prim algorithm has a temporal complexity of O (n2) and the Kruskal algorithm has a similar complexity of O (n \* log (n)).

**Phase 5: Evaluation and selection of the best solution**

For ease of implementation it was decided to use the Prim algorithm.

**Phase 6: Preparation of reports and specifications**

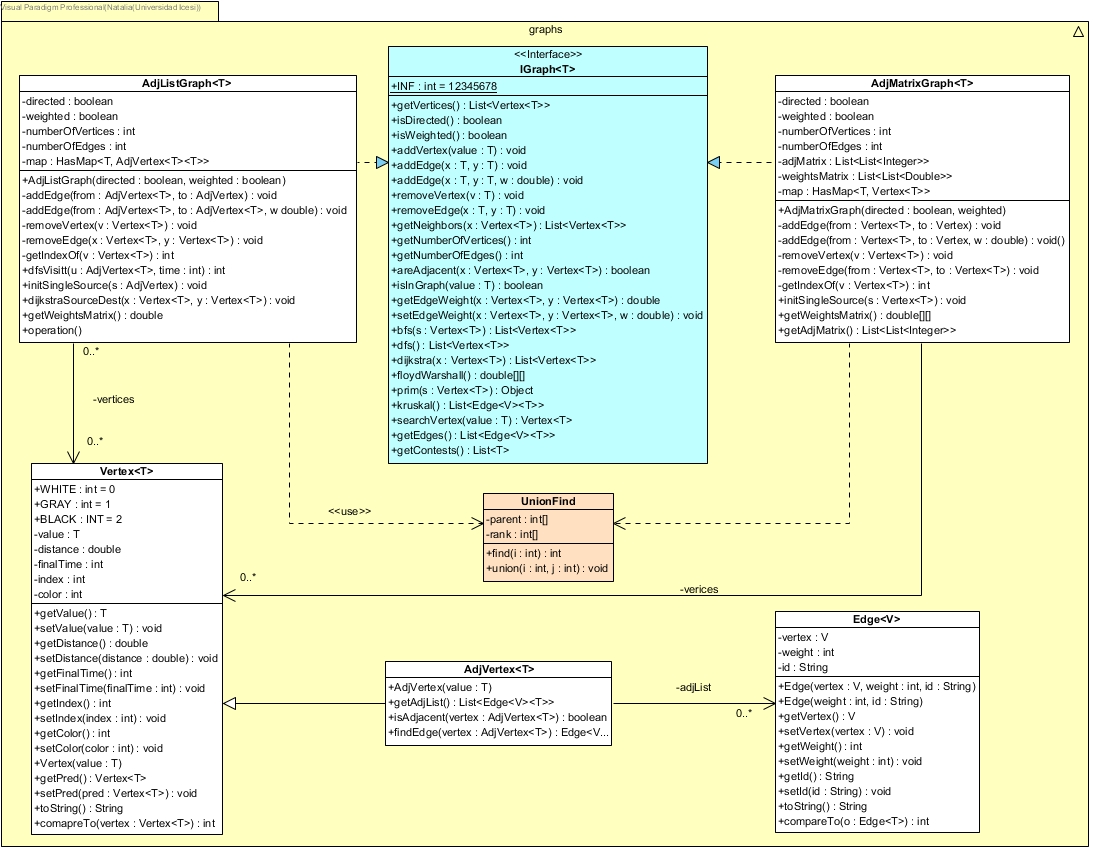
**Functional requirements**

|  |  |
| --- | --- |
| **Name** | **R.# 1. Find the maximum amount of money that can be saved** |
| **Summary** | Find the maximum amount of money that can be saved without sacrificing the safety of citizens |
| **Input** | |
| • Number of places  • Number of roads  • Connection between places  • The distance of the connections | |
| **Output** | |
| A number that indicates the money that can be saved with this method | |

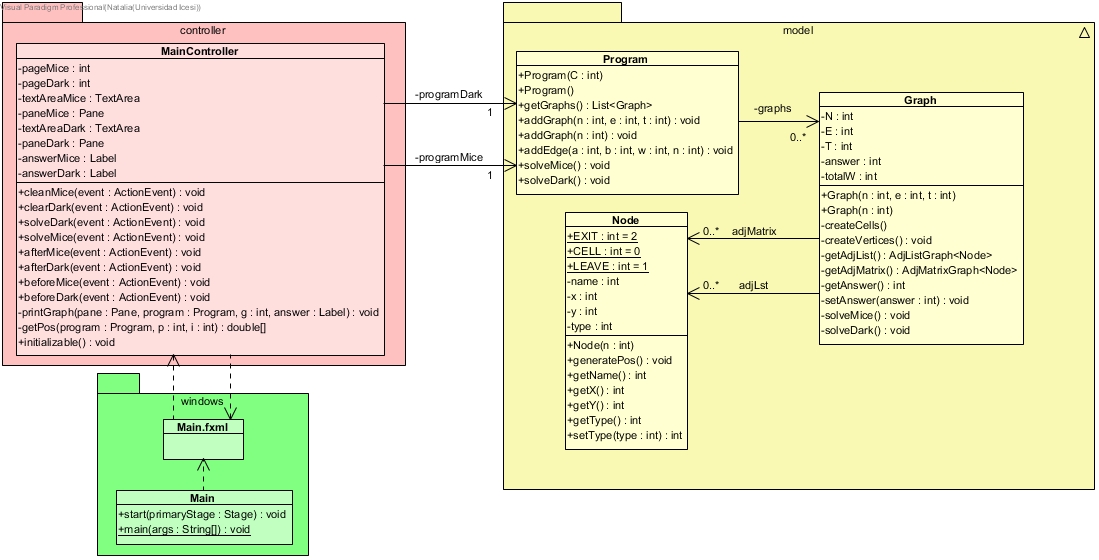
|  |  |
| --- | --- |
| **Name** | **R. # 2. Display the shape of the city** |
| **Summary** | Display the shape of the city |
| **Input** | |
| • Number of places  • Number of roads  • Connection between places  • The distance of the connections | |
| **Output** | |
| A display of the city by screen | |

|  |  |
| --- | --- |
| **Name** | **R.# 3. View the route that is on and connect all places in the city** |
| **Summary** | View the route that remains on |
| **Input** | |
| None | |
| **Output** | |
| A visualization of the shortest streets that connect all places in the city. | |

**Class Diagram Graph Implementation**

****

**Class Diagram for Problems**

****