Método de la ingeniería

I drive

*Descripción*:

Es 1970, Dani el rojo, un experto atracador de bancos se encuentra haciendo un rutinario atraco a la Fábrica nacional de la moneda y timbre en españa, El atraco va bien, dani se ha hecho con 200 millones de pesetas, lo único que falta es salir del banco y llegar a una de sus guaridas donde esté seguro y complete el atraco.Tu eres el conductor elegido para llevar a dani a alguna de sus guaridas. Dispones de tu Delorean y tu habilidad al volante, debes saber todos los caminos de la ciudad a la perfección. La ciudad cuenta con:

50 Vértices (ubicaciones): Cada uno de los 50 vértices representa una ubicación única en la ciudad, como 50 intersecciones de calles o puntos de interés diferentes(museo,colegio…etc).

50 Aristas (conexiones entre ubicaciones): Las 50 aristas representan las calles o caminos que conectan las ubicaciones. Cada arista conecta dos vértices y tiene un peso que representa el tiempo de viaje.

Tu primera misión va a ser recorrer la ciudad para ver posibles caminos, atajos y obstáculos que se presenten.

Debes llegar a una de las guaridas de Dani el Rojo en el mejor tiempo posible para que así escape de la policía y obtengas tu recompensa de el 20% de la recaudación del golpe.

¡Depende de ti!

Jugabilidad

Te puedes mover por cualquier calle, en un atraco no hay leyes ni semáforos.

Ten cuidado con los obstáculos, podrían hacer que te atrapen.

Condiciones de victoria.

Llegar en el menor tiempo posible a una de las guaridas.

# Step 1. Problem identification

*Problem Definition*

El problema es desarrollar un juego que cuente con grafos de con un **mínimo de 50 vértices** y **50 aristas,** y tal que para su solución sea posible aplicar al menos dos (2) de los algoritmos de grafos en este caso Dijkstra y DFS.

*Identification of needs and symptoms*.

-El jugador se debe poder mover por el grafo(mapa).

-se debe poder cambiar a la otra versión de grafo sin problema.

-debe tener interfaz gráfica.

*Requirements*

| Client | Marlon |
| --- | --- |
| User | Player of the game |
| Functional requirements | RF0 - Add Player  RF1 - Show map  RF2 - Start Game  RF3- Change version |
| Problem context | Se requiere desarrollar un juego en el que se permita al jugador escapar de la policía por el mejor camino posible. |
| Non-functional requirements | * Quick Response * intuitivo * Interfaz gráfica |

| Identifier and Name | *[RF0-add player]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *The system allows adding a player.* | | | |
| Inputs | **Name** | **Data type** | | **Valid values condition** |
| Nickname | String | |  |
| Result or Postcondition | The program stores the player. | | | |
| Outputs | **Output Name** | | **Data type** | **Format** |
| msg | | String | A message is displayed confirming that the player has been added. |

## **Step 2. Gathering Information**

*Information about what is required*

*Definitions*

Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices( V ) and a set of edges( E ). The graph is denoted by G(V, E).

Tomado de:

<https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/#what-is-graph-data-structure>

Tipos de grafos:

Dirigido: es aquel que las aristas tienen un único sentido, Los aristas de un grafo dirigido se representan gráficamente con flechas.

tomado de: https://www6.uniovi.es/usr/cesar/Uned/EDA/Apuntes/TAD\_apUM\_07.pdf

No Dirigido: Un grafo no dirigido es un grafo donde las aristas conectan a los nodos en ambos sentidos.

tomado de: <https://www6.uniovi.es/usr/cesar/Uned/EDA/Apuntes/TAD_apUM_07.pdf>

Recorridos:

Breadth First Search (BFS) is a graph traversal algorithm that explores all the vertices in a graph at the current depth before moving on to the vertices at the next depth level. It starts at a specified vertex and visits all its neighbors before moving on to the next level of neighbors. BFS is commonly used in algorithms for pathfinding, connected components, and shortest path problems in graphs.

Tomado de:<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

**The Floyd-Warshall algorithm:**is a fundamental algorithm in computer science and graph theory. It is used to find the shortest paths between all pairs of nodes in a weighted graph. This algorithm is highly efficient and can handle graphs with both positive and negative edge weights, making it a versatile tool for solving a wide range of network and connectivity problems..

tomado de: <https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/>

Dijkstra: El algoritmo de Dijkstra, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con respecto a los pesos en cada arista.

La idea consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.

Tomado de: <http://atlas.uned.es/algoritmos/voraces/dijkstra.html#:~:text=El%20algoritmo%20de%20Dijkstra%2C%20tambi%C3%A9n,con%20pesos%20en%20cada%20arista>.

### **Step 3. Creative Solutions Search**

**Alternative 1:**

**Alternative 2:**

**Alternative 3:**

**Step 4. Transition from Ideas to Preliminary Designs.**

**Step 5. Evaluation and Selection of the Best Solution**

*Evaluation*

|  | Criteria 1 | Criterio 2 | Criterio 3 | Criterio 4 | Total |
| --- | --- | --- | --- | --- | --- |
| Alternativa |  |  |  |  |  |
| Alternativa |  |  |  |  |  |

*Selection*

In conclusion,