METODO DE LA INGENIERIA

PHASE 1: **Problem identification**

A cutting-edge game development company, renowned for its focus on the application of graph theory and advanced logic, has been selected by the Futuristic Transportation Department (FTD) to create innovative software. This software aims to simulate and optimize routes in a complex futuristic transportation network centered around strategic connection points. While the initial idea is for the game to be an entertainment experience, it also has a broader purpose: to fund research on the mobility and transportation of the future.

In this game, players will engage in missions to optimize transportation between cities and points of interest in a futuristic world. The transportation network will consist of a variety of crucial connection points, and the main task will be to find the most efficient and fastest route that connects them. The challenge lies in minimizing costs and travel times by skillfully navigating through this intricate network of connections.

The complexity of the game will intensify as players face obstacles such as traffic congestion, resource limitations, and unexpected events that impact the efficiency of the transportation network. The application of graph theory principles will be essential to optimize connections and create an efficient and resilient transportation network centered around these strategic points

PHASE 2 : **Collection of necessary information**

Graph: A collection of items connected by edges, where each item is referred to as a vertex or node. Formally, a graph is defined as a pair (V, E), where V is a set of vertices, and E is a set of edges between the vertices. For an undirected graph, the adjacency relation defined by edges is symmetric, and if the graph is without self-loops, adjacency is irreflexive.

• Source: NIST

Adjacency List: A collection of unordered lists used to represent a finite graph. Each unordered list within an adjacency list describes the set of neighbors of a particular vertex in the graph. This representation is commonly used in computer programs.

• Source: Wikipedia

Adjacency Matrix: A 2D array of size V x V (where V is the number of vertices) representing a graph. If adj[i][j] = 1, it indicates an edge from vertex i to vertex j. For an undirected graph, the matrix is symmetric. It is also used to represent weighted graphs, where adj[i][j] = w signifies an edge with weight w.

• Source: GeeksforGeeks

Depth-First Traversal (DFS): A technique used to traverse a tree or graph. It starts with a root node and explores adjacent nodes depth-wise until reaching a leaf node. DFS uses a stack to store nodes being traversed.

• Source: SoftwareTestingHelp

Breadth-First Traversal (BFS): A technique using a queue to traverse a graph breadth-wise. It explores vertices level-wise, moving to the next level after completing the current one.

• Source: SoftwareTestingHelp

Dijkstra's Algorithm: Also known as the minimum paths algorithm, it determines the shortest path from a given vertex to all other vertices in a weighted graph. It is a greedy algorithm that works in stages, selecting the best solution at each stage without considering future consequences.

• Source: EcuRed

Kruskal's Algorithm: A minimum spanning tree algorithm that selects a subset of edges in a graph, forming a tree including every vertex with the minimum sum of weights. It falls under greedy algorithms, aiming to find a local optimum for a global optimum.

• Source: Programiz

Floyd Warshall Algorithm: Used to find the shortest paths between all vertices in a weighted graph. It works with both directed and undirected graphs but cannot handle graphs with negative cycles. The algorithm follows a dynamic programming approach and iteratively updates the shortest paths matrix.

• Source: FavTutor

PHASE 3: **Exploration of Innovative Strategies for the Futuristic Network Game:**

Alternative 1:

In our first proposal, we focus on the application of Dijkstra and BFS algorithms in the context of a futuristic transportation network. In this scenario, nodes represent various cities and points of interest, while edges symbolize different means of transportation between them. We assign a weight to each edge to represent the efficiency of the transportation mode. Dijkstra is employed to find the shortest route between cities, considering edge weights as distance. On the other hand, BFS is used to explore all cities and points of interest, although without considering exact distances, focusing more on global connectivity.

Alternative 2:

Our second proposal involves the implementation of Prim and Kruskal algorithms in a weighted graph representing the futuristic transportation network. In this case, the goal is to build a minimum spanning tree connecting all cities and points of interest, optimizing connections. Prim selects edges with the shortest distances, ensuring an efficient expansion from an initial point to the destination. Kruskal, on the other hand, focuses on local optimization, organizing distances and adding edges with the shortest distances, avoiding the formation of cycles in the network.

Alternative 3:

In our third proposal, we consider the application of Floyd Warshall and DFS algorithms in a futuristic transportation network. Floyd Warshall is adapted to manage a matrix representing cities and points of interest. It continuously updates the shortest distances between all locations, allowing global optimization of the network. Although DFS does not directly contribute to finding the fastest route between two points, its application allows for in-depth exploration of possible connections and paths in the network, revealing various options for optimizing mobility.

These alternatives aim to address optimization challenges in the futuristic transportation network, combining the excitement of the game with the application of advanced graph theory principles to create challenging and educational experiences.

Phase 4: **Transition from Ideas to Preliminary Designs.**

To our understanding, there are no ideas that are not viable, as each one correctly models the problem according to the context. Further information below elaborates on the ideas that were not discarded.

Alternative 1:

Using Dijkstra and BFS allows us to select the shortest path, ignoring longer distances. With Dijkstra, the algorithm chooses the edge with the smallest weight, finding the shortest path to the destination. BFS doesn't use a weighted graph, so there are no distances; it visits each level and all its neighbors, recording the parents to trace back the shortest path.

Alternative 2:

Prim and Kruskal are also viable as they find the shortest path. Prim's algorithm finds the shortest path without creating any cycles. There's an initial vertex, and then edges are added until the point of interest is reached. Kruskal organizes distances from smallest to largest. Then the smallest distance is added to the spanning tree. The route cannot make a loop. If all vertices are reached, they are also added to the spanning tree.

Alternative 3:

Although DFS doesn't go for the shortest path, it explores all vertices and corresponding edge lengths, showing all possible paths. Floyd Warshall is created with a matrix containing each vertex as a coordinate row and column. The coordinate is updated if there is a distance smaller than the current one.

PHASE 5: **Evaluation and Selection of the Best Solution:**

Criteria:

The criteria for evaluating the alternative solutions must be defined, and based on this result, choose the solution that best meets the needs of the problem. The chosen criteria in this case are detailed below. Next to each one, a numerical value has been established to establish a weight indicating which of the possible values of each criterion has the most weight (i.e., they are more desirable).

Criteria 1: Uses distances according to the context.

[2] Yes

[1] No

Criteria 2: Fewer vertices.

[3] Has the fewest vertices

[2] Has more vertices than expected

[1] Has too many vertices

Criteria 3: Temporal complexity.

[2] Fast

[1] Slower

Criteria 4: Coding efficiency.

[2] Yes

[1] No

Criteria 1 Criteria 2 Criteria 3 Criteria 4 Total

Alternative 1 2 3 2 2

Alternative 2 2 2 1 1

Alternative 3 2 1 1 1

Selection: According to the previous evaluation, Alternative 1 should be selected since it obtained the highest score according to the defined criteria.