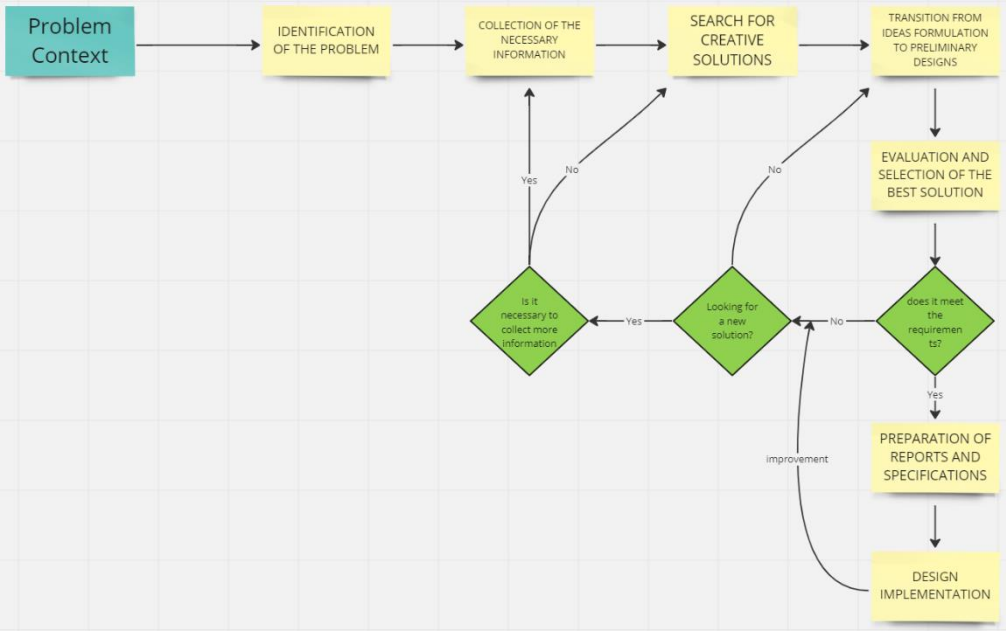
Problem Context

An airline operates in 50 principal cities in the United States Of America and wishes to schedule its flights efficiently to maximize passenger satisfaction and minimize operating costs. You have information about the cities where the airline operates and the availability of direct flights between them.

Solution Development



The objective is to find the optimal flight schedule that meets the following requirements

1. Connectivity: Ensure that all cities are accessible by direct flights or connections between cities. Unnecessary or excessive connections should be avoided to minimize flight time and costs.
2. Shorter routes: Find the shortest routes between city pairs, considering distance or flight time as metrics. This will enable faster and more efficient routes to be offered to passengers.
3. Optimization of scales: In case stopovers or connections are necessary to reach a destination, find the combination of flights that minimizes the total travel time or the associated cost.

- DFS(Depth First Search): It is an algorithm that explores a graph by prioritizing the exploration of deeper paths whenever possible. Starting from a source vertex, it systematically explores the edges of the graph, backtracking only when all edges of the current vertex have been explored. This process continues until all reachable vertices from the source have been discovered. If undiscovered vertices remain, DFS selects one as a new source and repeats the search. The algorithm creates a depth-first forest composed of multiple depth-first trees. During the search, vertices are colored

to indicate their state (white, gray, or black), and timestamps are assigned to track their discovery and completion. DFS is a versatile algorithm used in various applications, such as finding connected components, detecting cycles, and solving maze-like problems.

- BFS(Breadth First Search): It is a graph search algorithm used to traverse or search for elements in a graph in a systematic and structured way. It starts at an initial node and explores all its neighboring nodes before moving on to neighboring nodes at the next level. Furthermore, BFS ensures that all nodes at the same level are visited before moving on to the next level, which means that the nodes closest to the initial node are explored first and then gradually expand to nodes further away.
- Dijkstra: Dijkstra's algorithm is an efficient method for finding the shortest path from a given source node to all other nodes in a weighted, directed graph. Developed by Edsger Dijkstra, this algorithm is widely used in various fields such as communication networks and route planning. It is based on the idea of iteratively exploring the nodes of the graph, selecting the node with the shortest distance that has not been visited at each step. It utilizes a min-priority queue to keep track of the minimum distances and updates them as shorter paths are found. Upon completion, it yields a set of minimum distances and the corresponding paths to all nodes in the graph. In summary, Dijkstra's algorithm provides an efficient solution to the shortest path problem in weighted graphs, assuming non-negative edge weights.
- Floyd-Warshall: It is a graph analysis algorithm to find the minimum path in weighted graphs. In addition, the algorithm finds the value of the minimum path between all pairs of vertices in a single run. Finally, the Floyd-Warshall algorithm is an example of dynamic programming.
- Prim: It is an algorithm used to find the minimum spanning tree in a weighted undirected graph. The objective of the algorithm is to find a subset of edges that connect all the vertices of the graph, minimizing the sum of the weights of those edges. Prim's algorithm starts by selecting an arbitrary initial vertex and, from there, grows the minimum overlay tree incrementally. At each step, the minimum weight edge connecting a vertex in the tree to a vertex outside the tree is selected.
- Kruskal: is an algorithm used to find the minimum spanning tree in a weighted undirected graph. Like Prim's algorithm, the goal of Kruskal's algorithm is to find a subset of edges that connect all the vertices of the graph by minimizing the sum of the weights of those edges. Furthermore, Kruskal's algorithm starts with a graph that has each vertex as an isolated component and no edges in the minimum covering tree. Next, all edges in the graph are sorted in increasing order of weight. Then, the edge with the lowest weight is taken and checked if it connects two different components of the network. If so, it is added to the minimum overlapping tree. If the edge does not create a cycle in the tree, i.e., it does not connect two vertices already in the same component, it is added to the tree. This process is repeated for all edges in the network, in increasing order of weight.

First possibility solution: The Dijkstra and Prim algorithms will be used.

Second possibility solution: The BFS and DFS algorithms will be used.

Third possibility solution: The Floyd-Warshall and Kruskal algorithms will be used.

Fourth possibility solution: The Prim and Kruskal algorithms will be used.

Fifth possibility solution: The Dijkstra and Floyd-Warshall algorithms will be used.

Sixth possibility solution: The Prim and Floyd-Warshall algorithms will be used.

We discarded the third possible solution and the fourth possible solution because we tried to investigate the issues that we don't see yet, so by investigating, we could understand the Prim algorithm but we couldn't understand the Kruskal, so we discussed and determined that the best solution is to discard so we can do it much better.

First possibility solution: The Dijkstra and Prim algorithms will be used. Where the Dijkstra algorithm will be used to find the shortest routes between city pairs, where it can be used to determine the most efficient connections and minimize flight times. On the other hand, Prim is an algorithm used to find a set of least-cost edges that connects all vertices of an undirected graph. The Prim algorithm will take care of selecting the least-cost edges that will efficiently connect all the cities.

Second possibility solution: The BFS and DFS algorithms will be used. Apply BFS to find the shortest routes between cities. Start BFS from an origin city and explore all possible direct connections from that city. Record the shortest distances between the origin city and all other cities. Use DFS to find longer and more complete routes between cities. Start DFS from the origin city and explore all possible routes from that city, keeping track of the cities visited and avoiding cycles. This will allow exploring more detailed options and finding alternative routes. However, this possibility was discarded because we think that is not more efficient than the first possibility.

Fifth possibility solution: The Dijkstra and Floyd-Warshall algorithms will be used. Where the Dijkstra algorithm will be used to find the shortest routes between city pairs, where it can be used to determine the most efficient connections and minimize flight times. On the other hand, the Floyd-Warshall algorithm will be used to find the matrix of shortest distances between all the cities in the network, this will allow to have an overview of the shortest routes and optimize the flight scheduling. However, it would be unnecessary to use two algorithms that we believe may have the same function.

Sixth possibility solution: The Prim and Floyd-Warshall algorithms will be used. Use the information obtained from the Floyd-Warshall algorithm to plan flights and schedules efficiently. You can optimize passenger satisfaction by minimizing flight times and maximizing connectivity between cities, avoiding unnecessarily long hauls or multiple layovers. Use the information obtained from the Floyd-Warshall algorithm to plan flights and schedules efficiently. You can optimize passenger satisfaction by minimizing flight times and maximizing connectivity between cities, avoiding unnecessarily long hauls or multiple layovers. Both algorithms have high time complexity and can be inefficient when applied to large data sets.

Criteria

The criteria that will allow the evaluation of alternative solutions must be defined and based on this result choose the solution that best meets the needs of the problem posed. The criteria we chose in this case are listed below. Also, we already chose the first alternative because it

is the one that can be adapted the most to solve the problem, however, it was analyzed and explained why it is a good criteria.

- Criterion A. Precision of the solution. The alternative delivers a solution: Dijkstra's algorithm and Prim's algorithm are accurate because they find the optimal solution for the problems they address, ensuring that the most efficient and correct result is achieved according to the established criteria.
- Criterion B. Efficiency. A solution with better efficiency is preferred than the others considered. The efficiency can be: Drijkstra uses a data structure called priority queue to select the next node with the shortest distance from a set of unvisited nodes. Prim uses a data structure called priority queue to select the next edge with the lowest weight to be added to the minimum overlay tree. Thus, they are efficient in finding optimal solutions in graph problems.
- Criteria C. Completeness. A solution that finds all solutions is preferred. How many solutions does it deliver: Dijkstra's algorithm and Prim's algorithm are complete because they exhaustively explore all possible options and guarantee to find an optimal solution if it exists in the given network. Moreover, because they guarantee to find an optimal solution if it exists in the given network.
- Criteria D. Ease of algorithmic implementation: the clear and structured design of Dijkstra's and Prim's algorithms and appropriate data structures contribute to their ease of implementation. This makes them accessible even to those who are just becoming familiar with graph algorithms and helps them to be widely used in a variety of applications. Therefore, it is in line with our standards