ReflexActInt2_A01638959

In this integrative activity, we worked on a program that simulates the installation of a communication network in a city composed of several neighborhoods. The main goal was to connect all the neighborhoods efficiently, minimize distances for delivery routes, analyze data transmission capacities, and determine the nearest exchange for new connections.

To solve each part, we implemented four different algorithms: Kruskal, Nearest Neighbor, Ford–Fulkerson, and Voronoi.

In the first part, we needed to find a Minimum Spanning Tree (MST) in a weighted graph, where nodes represent neighborhoods and edge weights represent distances in kilometers. For this, we decided to use Kruskal's algorithm because it is simple to implement, works efficiently for sparse graphs, and guarantees the minimal wiring cost. Kruskal's algorithm sorts all the edges by their weight and then adds them one by one to the MST, as long as they do not form a cycle.

We used a Disjoint Set Union (DSU) structure to keep track of connected components and detect cycles efficiently.

The complexities of the algorithm are the following:

Time Complexity

$O(E \log E)$, where E is the number of edges, mainly due to the sorting step

Space Complexity

$$O(V + E)$$

This algorithm provided the exact set of connections between neighborhoods that minimizes the total cost of the fiber network.

In the second phase, we needed to find the shortest possible route that visits each neighborhood exactly once and returns to the starting point, a classic Traveling Salesman Problem (TSP). Since finding the exact solution would be computationally expensive for larger

datasets, we implemented the Nearest Neighbor heuristic.

This algorithm starts from a city and repeatedly visits the nearest unvisited city until all have been visited. Even though it doesn't always find the perfect optimal path, it's a good approximation that runs much faster, with a time complexity of $O(n^2)$. We also applied the repetitive version, where the algorithm starts from every city and selects the best route, improving accuracy while keeping a manageable runtime.

The third task was a maximum flow problem for which the Ford–Fulkerson algorithm was used, specifically implemented using the Edmonds–Karp approach, which uses BFS to find augmenting paths. This method updates the residual capacities of the network iteratively until no more flow can be sent from the source to the sink.

Time Complexity

$$O(V * E^2)$$

Space Complexity

$$O(V^2)$$

For the final phase, we needed to determine which "exchange" or service center was closest to a new potential customer, based on geographical coordinates. To visualize and compute this, we used Voronoi diagrams, implemented through the CGAL library.

A Voronoi diagram partitions the plane into regions based on the nearest site (in our case, an exchange). This allows the system to instantly decide which center should serve a new contract. The algorithm's complexity, based on CGAL's Delaunay triangulation construction, is O(n log n). It was particularly useful for understanding spatial relationships between service points and neighborhoods.