Engineering design process

Step 1: Problem identification

Problem Definition: The well-known London Underground has 289 stations and 11 lines that connect the railway circuit, due to the large number of options to travel around the city in this type of transport, most users of the Underground have seen the need for an application to help them find the most optimal route to move to their destination. On the one hand, some users say that the most optimal route is the one that goes through fewer stations, because that way they don't waste time changing to other stations. On the other hand, some users say that the most optimal route will always be the one that takes the least time between the starting station and the destination station, regardless of possible station changes. In this sense, it is important to develop a system that meets both of these user needs.

Identification of needs and symptoms:

- The system must be efficient due to the large number of stations.
- The system must meet the needs of all users.
- The system must have a user-friendly interface due to the large number of stations.
- The system should allow the user to see the stations on the London Underground map.

Step 2. Information Gathering

Graph: A 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(E, V).

Depth First Search: DFS is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path.

Breadth First Search: BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.

Dijkstra's: This algorithm makes a tree of the shortest path from the starting node, the source, to all other nodes (points) in the graph. Dijkstra's algorithm makes use of weights of the edges for finding the path that minimizes the total distance (weight) among the source

node and all other nodes. This algorithm is also known as the single-source shortest path algorithm. It is important to note that Dijkstra's algorithm is only applicable when all weights are positive because, during the execution, the weights of the edges are added to find the shortest path.

Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm works for both the directed and undirected weighted graphs. But, it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

Spanning Tree: A Spanning tree is a subset to a connected graph G, where all the edges are connected, i.e, one can traverse to any edge from a particular edge with or without intermediates. Also, a spanning tree must not have any cycle in it. Thus we can say that if there are N vertices in a connected graph then the no. of edges that a spanning tree may have is N-1.

Minimum Spanning Tree: Given a connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

Prim's Algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized. Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which we can traverse every vertex of the graph. It follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum.

Bellman Ford Algorithm: Bellman ford algorithm is a single-source shortest path algorithm. This algorithm is used to find the shortest distance from the single vertex to all the other vertices of a weighted graph. There are various other algorithms used to find the shortest path like Dijkstra algorithm, etc. If the weighted graph contains the negative weight values, then the Dijkstra algorithm does not confirm whether it produces the correct answer or not. In contrast to Dijkstra algorithm, bellman ford algorithm guarantees the correct answer even if the weighted graph contains the negative weight values.

A* Algorithm is one specific pathfinding algorithm, first published in 1968 by Peter Hart, Nils Nilsson, and Bertram Raphael. It is generally considered to be the best algorithm to use

when there is no opportunity to pre-compute the routes and there are no constraints on memory usage. Both memory and performance complexity can be O(b^d) in the worst case, so while it will always work out the most efficient route, it's not always the most efficient way to do so. A* is actually a variation on Dijkstra's Algorithm, where there is additional information provided to help select the next node to use. This additional information does not need to be perfect – if we already have perfect information, then pathfinding is pointless. But the better it is, the better the end result will be.

Johnson's algorithm is a shortest path algorithm that deals with the all pairs shortest path problem. The all pairs shortest path problem takes in a graph with vertices and edges, and it outputs the shortest path between every pair of vertices in that graph. Johnson's algorithm is very similar to the Floyd-Warshall algorithm; however, Floyd-Warshall is most effective for dense graphs (many edges), while Johnson's algorithm is most effective for sparse graphs (few edges).

Step 3. Search for Creative Solutions

For the generation of creative ideas, the **brainstorming** technique was used to review the concepts investigated in the collection of information and those seen in the course.

R1. Find fastest route

Dijsktra´s algorithm: This solution proposes to implement a directed and weighted graph. The stations would be vertices and the tracks connecting them, edges. The edges would have an associated weight corresponding to the time it takes the subway to go from one station to another. Now, to find the shortest path it is proposed to use Dijkstra's algorithm. This algorithm would return the shortest time to travel between two stations and the path traveled to find the minimum time.

Floyd – Warshall algorithm: This solution proposes to implement a directed and weighted graph. The stations would be vertices and the tracks connecting them, edges. The edges would have an associated weight corresponding to the time it takes the subway to go from one station to another. Now, to find the shortest path it is proposed to use the Floyd - Warshall algorihm algorithm. This algorithm would return the shortest time to travel between each pair of London Underground stations and the path taken to find each minimum time.

A* algorithm: This solution proposes to implement a directed and weighted graph. The stations would be vertices and the tracks connecting them, edges. The edges would have an associated weight corresponding to the time it takes the subway to go from one station to another. Now, to find the shortest path it is proposed to use the A* algorithm. This algorithm would return the shortest time to travel between two stations and the path traveled to find the minimum time.

Johnson's Algorithm: This solution proposes to implement a directed and weighted graph. The stations would be vertices and the tracks connecting them, edges. The edges would have an associated weight corresponding to the time it takes the subway to go from one station to another. Now, to find the shortest path it is proposed to use Johnson's algorithm. This algorithm would return the shortest time to travel between any two London Underground stations and the path taken to find each minimum time.

R2. Find shortest route

BFS: This solution proposes to implement a directed and weighted graph. The stations would be vertices and the roads connecting them, edges. Now, to find the shortest path it is proposed to use the BFS algorithm. This algorithm would return the minimum number of stations to be traversed to reach the destination station.

DFS: This solution proposes to implement a directed and weighted graph. The stations would be vertices and the roads connecting them, edges. Now, to find the shortest path it is proposed to use the DFS algorithm. This algorithm would return the minimum number of stations to be traversed to reach the destination station.

Step 4. Transition from ideas to preliminary designs

First, we discard the idea of the directed and weighted graph with the implementation of the **A* algorithm** to find the fastest path. Because, the implementation of this algorithm is very complicated and also uses more advanced concepts that are not within our knowledge.

Second, we discard the idea of the directed and weighted graph with the implementation of **Johnson's algorithm** to find the fastest path, because this algorithm works best for sparse graphs. Based on the research, the complexity of this algorithm depends on the number of edges in the graph. That is, if the number of edges is small this algorithm is efficient. In the context of the London Underground, the number of edges is very large, so the implementation of this algorithm is not suitable for this problematic situation.

Careful review of the other alternatives leads us to the following:

R1. Find fastest route

Dijsktra´s algorithm:

- The algorithm is executed every time the user needs to know the fastest route between two stations.

- The solution provides the list of stations of the fastest route and the time it takes to travel the route.
- The algorithm proposed by this solution is efficient.

Floyd – Warshall algorithm:

- This solution consumes a lot of resources to solve the proposed problem.
- The algorithm has a cubic time complexity.
- The solution provides the list of stations of the fastest route and the time it takes to travel the route.
- The solution is easy to implement.

R2. Find shortest route

BFS:

- The solution provides the list of stations on the shortest route and the number of stations on the route.
- The solution is easy to implement.
- The algorithm proposed by this solution is efficient

DFS:

- The solution provides the list of stations on the shortest route and the number of stations on the route.
- The solution is easy to implement.
- The algorithm proposed by this solution is efficient

.

Step 5. Evaluation and Selection of the Best Solution

R1. Find fastest route

Criterion A. Ease of implementation

- [1] difficult
- [2] medium
- [3] easy

Criterion B. Efficiency

- [1] low
- [2] average
- [3] high

Criterion C. Resource use

- [1] high
- [2] average

- [3] low

Criterion D. Completeness

- [1] low
- [2] average
- [3] high

	Criterion A	**Criterion B**	**Criterion C**	**Criterion D**	**Total**
Dijsktra´s algorithm	2	3	3	3	11
Floyd – Warshall algorithm	3	2	2	3	10

According to the above evaluation, we have that the best solution to meet the requirement **R1** of the fastest route between two London Underground stations is the weighted directed network with the implementation of the **Dijkstra algorithm**.

R2. Find shortest route

Criterion A. Ease of implementation

- [1] difficult
- [2] medium
- [3] easy

Criterion B. Efficiency

- [1] low
- [2] average
- [3] high

Criterion C. Solution accuracy.

- [1] high
- [2] average
- [3] low

Criterion D. Completeness

- [1] low
- [2] average
- [3] high

	Criterion A	Criterion B	Criterion C	Criterion D	Total
BFS algorithm	3	3	3	3	12
DFS algorithm	3	3	1	1	8

According to the above evaluation, we have that the best solution to meet the requirement **R2** of the shortest route between two London Underground stations is the weighted directed network with the implementation of the **BFS algorithm**.

Referencias

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/
https://www.analyticssteps.com/blogs/dijkstras-algorithm-shortest-path-algorithm
https://www.programiz.com/dsa/floyd-warshall-algorithm
https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/
https://www.javatpoint.com/prim-algorithm
https://www.javatpoint.com/kruskal-algorithm
https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
https://www.javatpoint.com/bellman-ford-algorithm
https://www.baeldung.com/java-a-star-pathfinding
https://brilliant.org/wiki/johnsons-algorithm/