NPS2001D

Milestone 4

Jiang Yu Hang	A0233144R
Chloe Victoria Ong	A0255644Y
Felicia Lee Sin Yee	A0288451W
He Wenye	A0265179R

Q1.

The primary algorithm we use in the app is Dijkstra's algorithm. It is a Single Source Shortest Path (SSSP) algorithm that calculates the shortest distance from a source node to other nodes in the graph. In our app, it is used for wayfinding where the source node and destination node are locations and other nodes are disability-friendly infrastructure. Though other algorithms perform similar tasks, we have concluded that Dijkstra's algorithm fits our app's purpose the most.

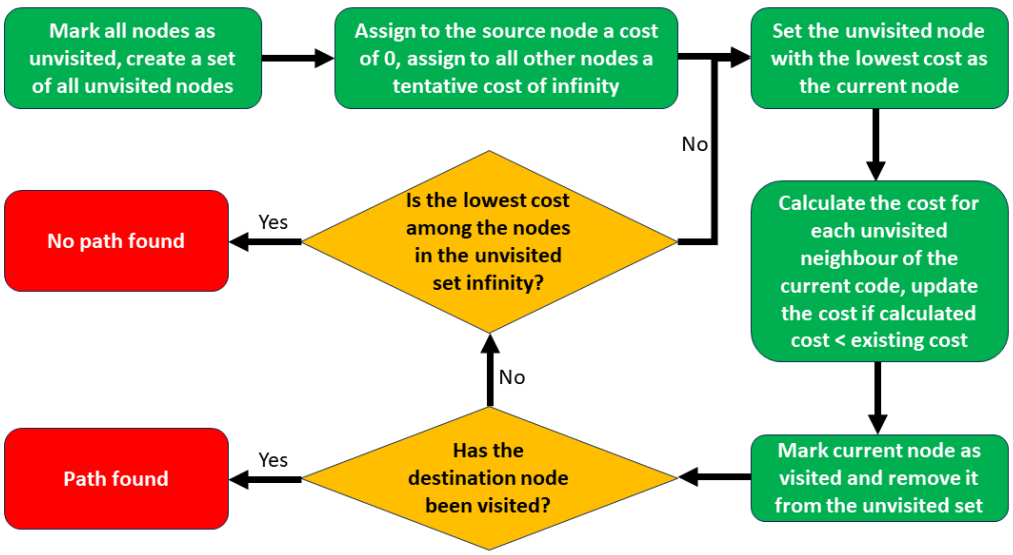
One other SSSP algorithm is the Bellman-Ford algorithm. The Bellman-Ford algorithm also finds the shortest distances between the source node and other nodes in the graph. Though similar in purpose, the Bellman-Ford algorithm has a higher time complexity than Dijkstra's algorithm. In fact, Dijkstra's algorithm is known to have the best running time among the numerous SSSP algorithms for non-negative edge weights.

Beyond SSSP algorithms, the Floyd-Warshall algorithm, an All Pair Shortest Path (APSP) algorithm, can also be used to find the shortest path in a graph. Unlike Dijkstra's algorithm which explores only the relevant portions of the graph and halts once the destination node has been visited, the Floyd-Warshall algorithm computes the shortest paths between all pairs of vertices, making it less suitable for real-time route planning in dynamic environments due to its high computational complexity.

Our app is intended for wayfinding on the go and prioritises efficiency. Hence, we choose to use Dijkstra's algorithm as it is best aligned with our purpose. Although Dijkstra's algorithm has its limitations, for instance, it is unable to handle negative edge weights, these are inapplicable to the workings of our app.

In conclusion, Dijkstra's algorithm is best suited for our app's purpose of time-efficient wayfinding than other SSSP or APSP algorithms.

Q2.



The inputs are the starting location (source node) and the destination (destination node). The output is the shortest path from the starting location (source node) to the destination (destination node).

Q3.

Dijkstra's algorithm selects the shortest path to any node from the source node. As a result, it might not account for disability-friendly infrastructure such as tactile paving, accessibility ramps and lifts if these infrastructures are not included in the shortest path. Negative cycles occur when a cycle of edges can be repeatedly transversed in such a way that the sum of the edge weights along the cycle is negative, decreasing the cost of the path with each iteration. When negative cycles are present, the assumption of Dijkstra's algorithm selecting the shortest path breaks down. When the Dijkstra's algorithm encounters a negative cycle, it can get stuck in a loop, repeatedly traversing the cycle and decreasing the overall path cost indefinitely or when it fails to detect the negative cycle, it might return incorrect results, as it assumes that the shortest path to any node has been found once it has been visited.

References

https://cs.indstate.edu/~rjaliparthive/dijkstras.pdf
https://www.cs.cmu.edu/~anupamg/advalgos15/lectures/lecture04.pdf
https://www.baeldung.com/cs/dijkstra-vs-bellman-ford#:~:text=As%20we%20can%20see%2C%20Dijkstra's,can%20help%20us%20with%20that.
https://www.sciencedirect.com/science/article/abs/pii/B9780123743640500114
https://www.baeldung.com/cs/floyd-warshall-shortest-path