How does Google Maps know the fastest route? The hidden mathematics of shortest paths

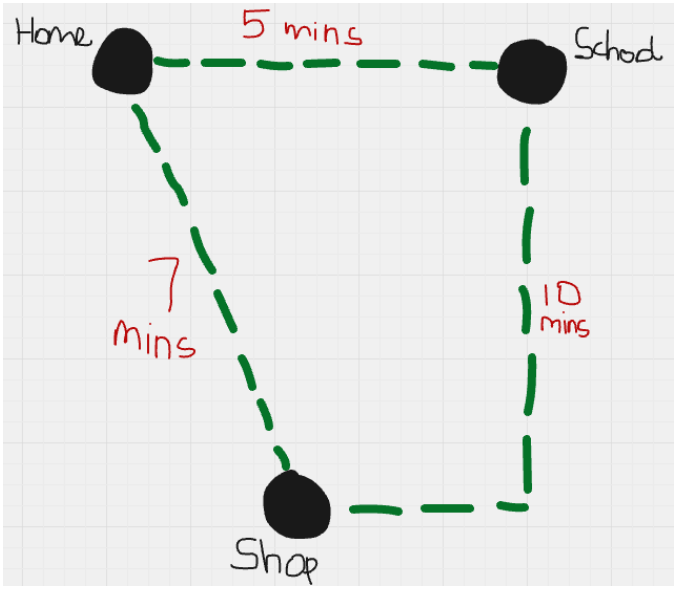
You're late for school. It's first period, and your Year 10 Maths test with Mr Smith is about to start. You pull out your phone and, within seconds, Google Maps has worked out the fastest route. Five minutes later, you're in your seat, trying to remember the quadratic formula.

But how did your phone know exactly where to send you?

The answer lies in some surprisingly powerful mathematics. Road systems may seem messy and unpredictable, but maths has a way of turning chaos into something structured and solvable. In this essay, I will explore how ideas such as shortest paths, graph theory, and algorithms allow navigation systems to guide us through the world with remarkable efficiency.

Turning cities into maths

Before a navigation system can determine the fastest route, real world must first be translated into a mathematical structure. This is achieved through graph theory, where locations like homes, schools, and junctions are represented as nodes, and the roads that connect them as edges. Each edge is given a weight which represents a quantity like travel time, distance or congestion. This generates a simplified model of a complex transport network.



This representation is concise and effective because it replaces the complexity of a city with a system of relationships and values that can be analysed mathematically. Instead of navigating a maze of streets, the problem becomes one of comparing numbers and tracing connections within a network, and therefore, easier. In this way, graph theory provides a bridge between the physical world and abstract mathematical reasoning, allowing real-world navigation to be expressed in a form that algorithms can systematically solve.

Dijkstra's algorithm — the clever search

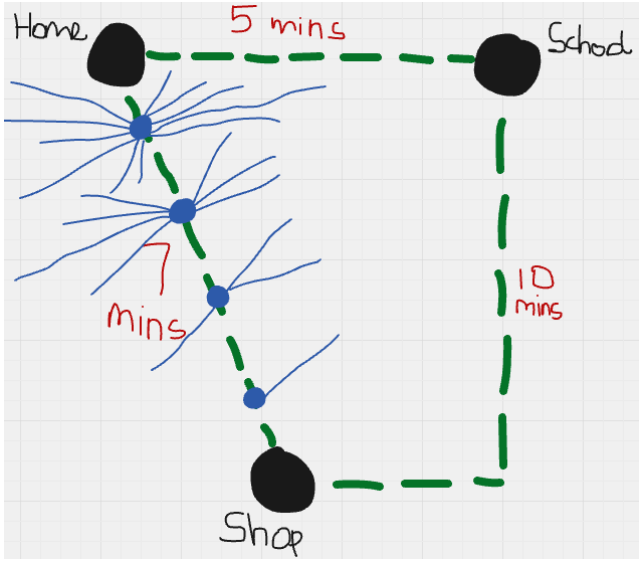
Once a city has been represented as a weighted graph, the next challenge is finding the most efficient route between two points. One of the most well-known methods for doing this is Dijkstra's algorithm, developed by the Dutch computer scientist Edsger W. Dijkstra in 1956.

The algorithm begins at a starting node, assigning it a distance of zero, while all other nodes are initially set to infinity. This reflects the idea that no routes to those locations are known at the start. From here, the algorithm repeatedly selects the nearest unvisited node — the one with the smallest known distance from the starting point.

At each step, it examines all neighbouring nodes and checks whether a shorter route can be found through the current position. If so, the recorded distance is updated. In this way, the algorithm steadily refines its estimates, turning rough guesses into optimal paths.

The key idea is that Dijkstra's algorithm always expands outward from the starting point in order of increasing distance. As a result, once a node has been fully processed, the shortest path to it has already been found. Provided all edge weights are positive, this guarantees that the final solution is optimal.

To illustrate this, consider a simple network connecting a school, a shop, and home. The algorithm first evaluates all direct routes from the shop, choosing the nearest location to explore next. It then continues step by step, updating distances as shorter paths are discovered. What begins as a small set of local choices gradually develops into a complete and efficient route through the network.

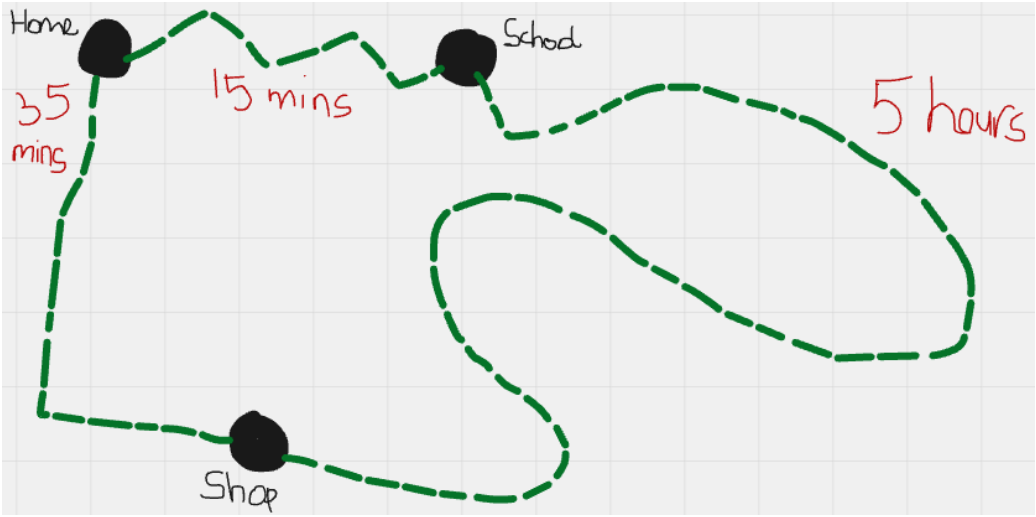


Instead of testing every possible route, Dijkstra's algorithm focuses only on the most promising paths at each stage. This is what allows modern navigation systems to compute routes across entire countries in fractions of a second.

Why is real life is harder

While algorithms such as Dijkstra's provide an efficient method for finding the shortest path, real-world navigation is far more complex than a fixed mathematical model. In practice, road networks are not static; they change continuously over time.

A route that appears optimal at one moment may quickly become inefficient due to traffic conditions. A journey that takes 5 minutes at one time of day may take 25 during rush hour. Congestion can increase travel times significantly, meaning that the weight assigned to a road is no longer constant. Similarly, unexpected events such as accidents, roadworks, or temporary closures can remove certain routes entirely, effectively altering the structure of the graph itself.



In addition to these unpredictable factors, navigation systems must also account for user preferences. Some journeys prioritise speed, while others may avoid toll roads or motorways. This means that “optimality” is not always a single fixed value, but can depend on individual constraints.

To handle this complexity, modern systems use the idea of dynamic weighting, where the values assigned to edges in a graph are continuously updated to reflect real-time conditions. In some cases, the graph itself is also modified, with edges being added or removed as the road network changes. This turns route planning into a constantly evolving problem rather than a one-time calculation.

This leads to an important insight: in reality, the shortest route is not always the quickest route. A mathematically shorter path may be slower due to congestion or restrictions, while a longer route may ultimately save time.

As a result, navigation systems do not simply solve a single graph problem. Instead, they repeatedly recompute and adjust solutions as new information becomes available, blending mathematical structure with real-time data. This ability to adapt is what allows systems like Google Maps to remain accurate in an ever-changing world.

Smarter algorithms and computer science

Although Dijkstra’s algorithm is effective in finding the shortest path, it is not always the most efficient method when dealing with extremely large networks. In systems such as modern navigation apps, where millions of routes must be processed in real time, efficiency becomes just as important as accuracy.

This is where the A* (A-star) search algorithm improves upon Dijkstra’s approach. While Dijkstra’s method explores the network evenly in all directions, A* introduces an element of prediction. Instead of treating every unexplored path equally, it uses an estimate of the remaining distance to the destination to guide its search. This estimate, known as a heuristic, is often based on the straight-line distance between two points.

By combining the actual distance travelled with this estimated remaining distance, A* prioritises routes that are more likely to lead directly towards the destination. As a result, it explores far fewer unnecessary paths compared to Dijkstra’s algorithm, making it significantly faster in many real-world applications.

This improvement is particularly important in fields such as game development and robotics, where rapid decision-making is essential. In video games, for example, characters must navigate environments smoothly without delay, while robots must react

instantly to their surroundings. In both cases, efficient pathfinding is crucial.

From a computational perspective, this efficiency can be described using Big O notation, which describes how the running time of an algorithm grows as the size of the input increases. While Dijkstra's algorithm may become slower as the network expands, A* reduces the number of nodes it needs to explore by focusing its search more intelligently.

Ultimately, A* demonstrates how computer science builds upon mathematical foundations to create systems that are not only correct, but also fast enough to operate at a global scale.

Conclusion

What begins as a simple journey from home to school becomes a much deeper mathematical problem. What appears on a phone screen as a simple route is, in reality, the result of carefully designed models, decades of mathematical development, and efficient algorithms working behind the scenes.

Ultimately, mathematics does more than find the shortest route — it turns the complexity of the real world into something structured, usable, and precise.

And all of that… just to get you to school on time.