Challenges of Google Maps' Design

Let's understand and resolve the key challenges in designing a system like Google Maps.

We'll cover the following

Meeting the challenges

We listed two challenges in the Introduction lesson: scalability and ETA computation. Let’s see how we meet these challenges.

Scalability

Scalability is about the ability to efficiently process a huge road network graph. We have a graph with billions of vertices and edges, and the key challenges are inefficient loading, updating, and performing computations. For example, we have to traverse the whole graph to find the shortest path. This results in increased query time for the user. So, what could be the solution to this problem?

The idea is to break down a large graph into smaller subgraphs, or partitions. The subgraphs can be processed and queried in parallel. As a result, the graph construction and query processing time will be greatly decreased. So, we divid

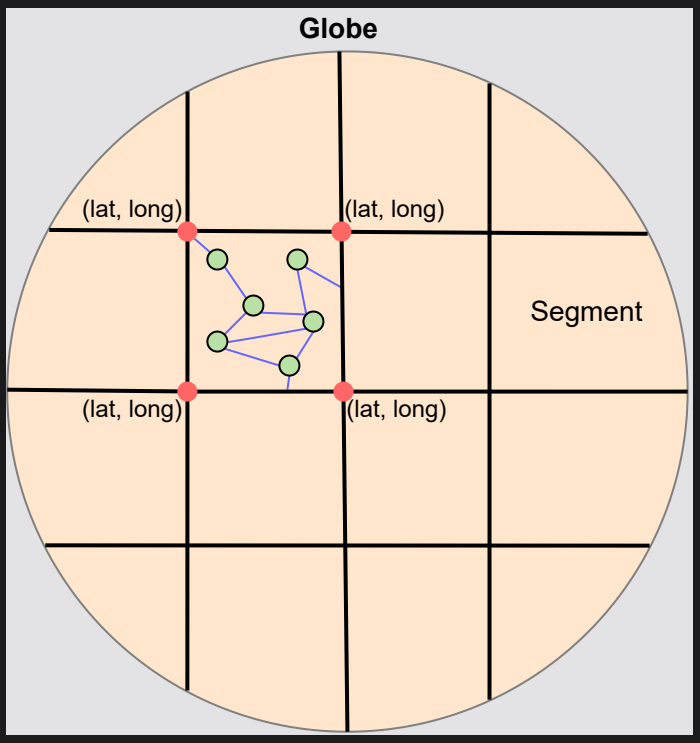
the globe into small parts called segments. Each segment corresponds to a subgraph.

Segment

A **segment** is a small area on which we can work easily. Finding paths within these segments works because the segment's road network graph is small and can be loaded easily in the main memory, updated, and traversed. A city, for example, can be divided into hundreds of segments, each measuring $5 \times 5\ miles$.

> **Note**: A segment is not necessarily a square. It can also be a polygon. However, we are assuming square segments for ease of explanation.

Each segment has four coordinates that help determine which segment the user is in. Each coordinate consists of two values, the latitude and longitude.



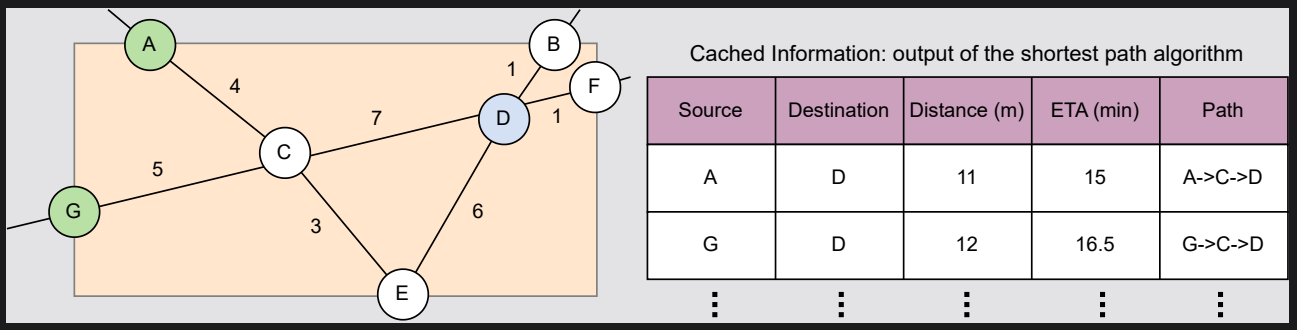
Partitioning the globe into small segments, and each segment has four coordinates with latitude and longitude values

Let's talk about finding paths between two locations within a segment. We hav a graph representing the road network in that segment. Each

intersection/junction acts as a vertex and each road acts as an edge. The graph is weighted, and there could be multiple weights on each edge—such as distance, time, and traffic—to find the optimal path. For a given source and destination, there can be multiple paths. We can use any of the graph algorithms on that segment's graph to find the shortest paths. The most common shortest path algorithm is the **Dijkstra's algorithm**.

More Shortest Path Algorithms

After running the shortest path algorithm on the segment's graph, we store the algorithm's output in a distributed storage to avoid recalculation and cache the most requested routes. The algorithm's output is the shortest distance in meters or miles between every two vertices in the graph, the time it takes to travel via the shortest path, and the list of vertices along every shortest path. All of the above processing (running the shortest path algorithm on the segment graph) is done offline (not on a user's critical path).



Cached Information: output of the shortest path algorithm

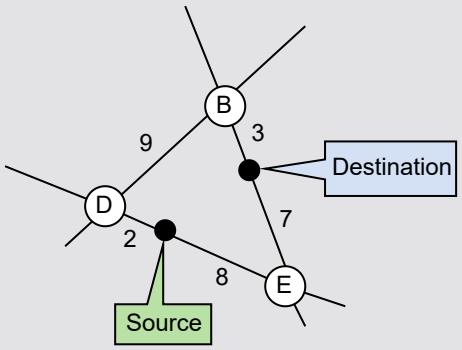
Source	Destination	Distance (m)	ETA (min)	Path
A	D	11	15	A->C->D
G	D	12	16.5	G->C->D
⋮	⋮	⋮	⋮	⋮

Finding a path between vertices within a segment

The illustration above shows how we can find the shortest distance in terms of miles between two points. For example, the minimum distance (m) between points A and D is 11 miles by taking the path A->C->D. It has an ETA of 15 minutes.

We've found the shortest path between two vertices. What if we have to find tl path between two points that lie on the edges? What we do is find the vertices of the edge on which the points lie, calculate the distance of the point from the

identified vertices, and choose the vertices that make the shorter total distance between the source and the destination. The distance from the source (and destination) to the nearest vertices is approximated using latitude/longitude values.



Possible paths	Distance
source->E->Destination	8 + 7 = 15
source->D->B->Destination	2 + 9 + 3 = 14

Since 14 is less than 15, we choose the path (source->D->B->destination)

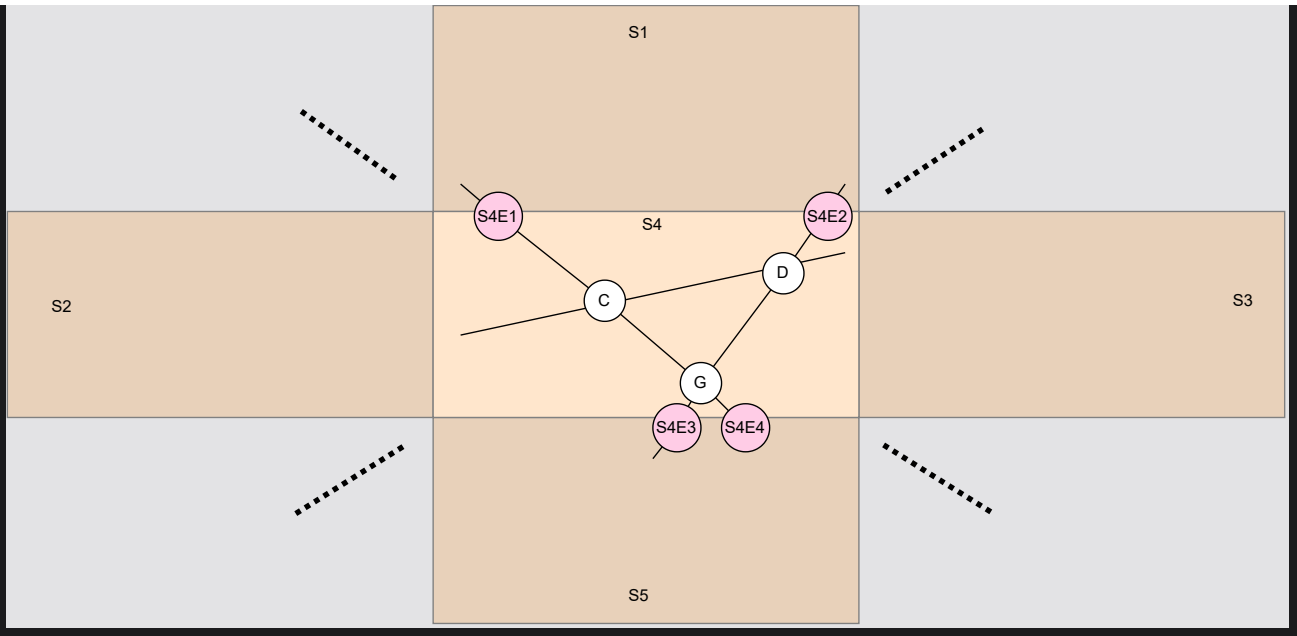
Choosing a path between points that lie on the edge

We're able to find the shortest paths within the segment. Let's see what happens when the source and the destination belong to two different segments, and how we connect the segments.

Connect two segments

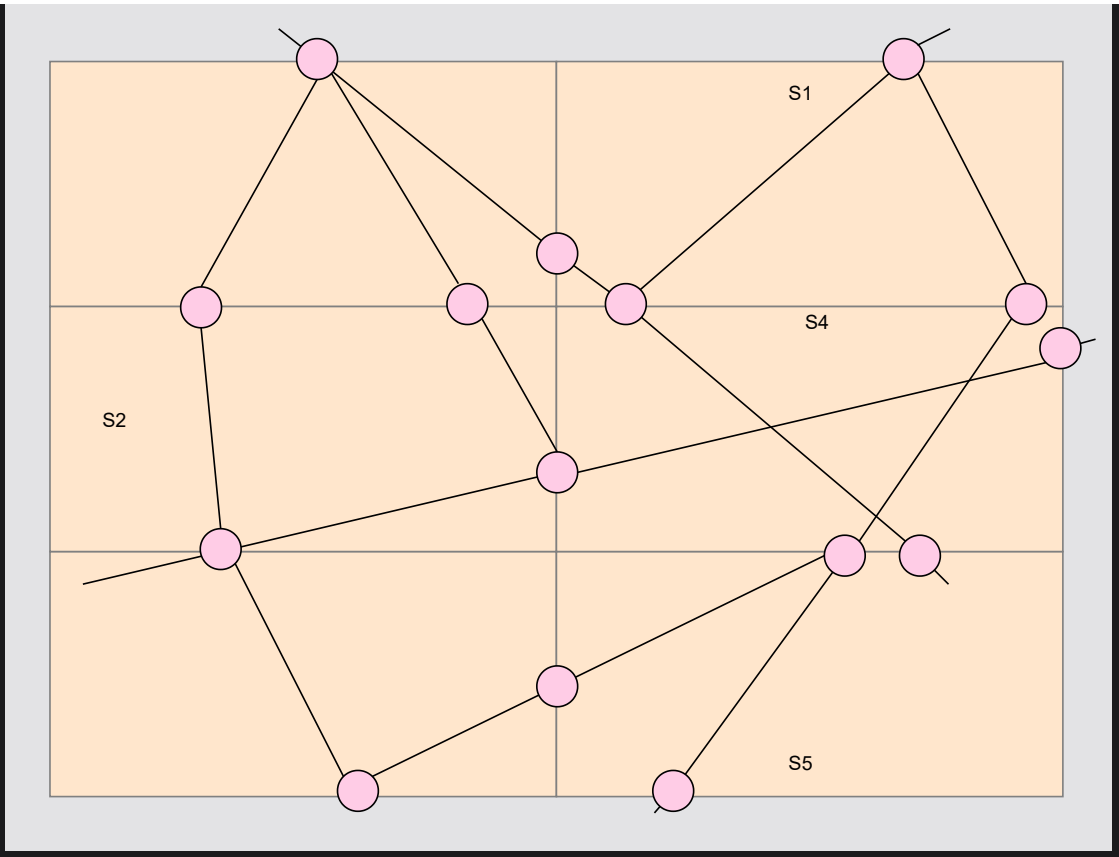
Each segment has a unique name and boundary coordinates. We can easily identify which location (latitude, longitude) lies in which segment. Given the source and the destination, we can find the segments in which they lie. For each segment, there are some boundary edges, which we call **exit points**. In the illustration below, we have four exit points, S4E1, S4E2, S4E3, and S4E4.

> **Note**: Besides the vertices we have inside the segment, we also consider the exit points of a segment as vertices and calculate the shortest path for these exit points. So for each segment, we calculate the shortest path between exit points in addition to the shortest path from the exit points to vertices inside the segment. Each vertex's shortest path information from the segment's exit points is cached.



Connecting the segments by considering exit points of a segment as vertices

An exit point connects neighboring segments and is shared between them. In the illustration above, each exit point is connecting two segments and each is shared between two segments. For example, the exit point S4E1 is also an exit point for S1. Having all the exit points for each segment, we can connect the segments and find the shortest distance between two points in different segments. While connecting the segments, we don’t care about the inside segment graph. We just need exit points and the cached information about the exit points. We can visualize it as a graph made up of exiting vertices, as shown in the following illustration.

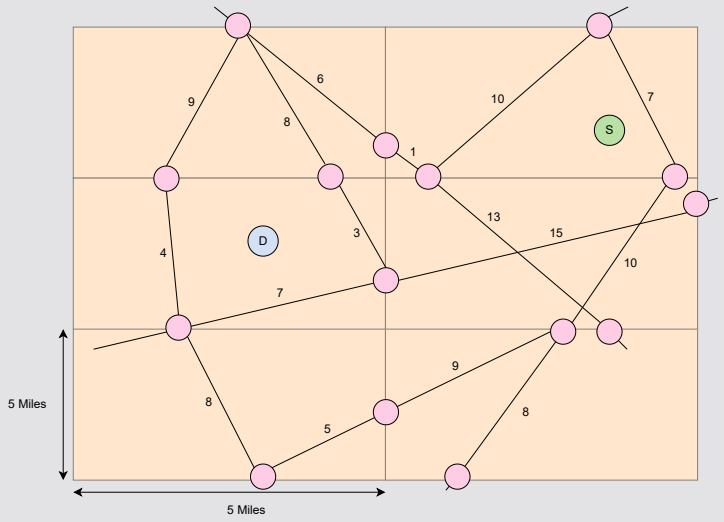


A graph made up of exit points, where the lines connecting the exit points are not actually straight. There could be many vertices in between the two exit points

Since we can't run the shortest path algorithm for all the segments throughout the globe, it's critical to figure out how many segments we need to consider for our algorithm while traveling inter-segment. The aerial distance between the two places is used to limit the number of segments. With the source and destination points, we can find the aerial distance between them using the haversine formula.

Suppose the aerial distance between the source and the destination is 10 kilometers. In that case, we can include segments that are at a distance of 10 kilometers from the source and destination in each direction. This is a significant improvement over the large graph.

Once the number of segments is limited, we can constrain our graph so that th vertices of the graph are the exit points of each segment, and the calculated paths betweenthe exit points are the graph's edges. All we have to do now is r the shortest path algorithm on this graph to find the route. The following illustration shows how we find the path between two points that lie in different segments.



Source S and Destination D lie in two different segments, and we have to find the shortest route between them

1 of 10

Let’s summarize how we met the challenge of scalability. We divided our problem so that instead of working on a large road network as a whole, we worked on parts (segments) of it. The queries for a specific part of the road network are processed on that part only, and for the queries that require processing more than one part of the network, we connect those parts, as we have shown above.

ETA computation

For computing the ETA with reasonable accuracy, we collect the live location data (`(userID, timestamp,(latitude, longitude))`) from the navigation service through a pub-sub system. With location data streams, we can calculate and predict traffic patterns on different roads. Some of the things that we can calculate are:

- Traffic (high/medium/low) on different routes or roads.
- The average speed of a vehicle on different roads.

- The time intervals during which a similar traffic pattern repeats itself on a route or road. For example, highway X will have high traffic between 8 to 10