Robust Transportation!

As dedicated museum professionals, we bear the responsibility of securely transporting precious artifacts and treasures from a museum, private collector, or a gallery in one city to another. Picture this: we are tasked with the mission of relocating a highly valuable treasure from an East Coast city to a West Coast city. However, here's the catch: the United States is an expansive country, crisscrossed by an intricate road network spanning over 4 million miles. Attempting to explore every conceivable route between these two treasure troves would be an insurmountable challenge. So, how do vigilant custodians like us formulate an intelligent strategy for the efficient transportation of these priceless relics from one city to another? The solution lies in the ingenious A* algorithm!

We've prepared a dataset of major highway segments of the United States (and parts of southern Canada and northernMexico), including highway names, distances, and speed limits; you can visualize this as a graph with nodes as towns and highway segments as edges. We've also prepared a dataset of cities and towns with corresponding latitude-longitude positions.

Your job is to find good driving directions between pairs of cities given by the user.
The code can be run on the command line like this:
python3 ./route.py [start-city] [end-city] [cost-function]
where:

- start-city and end-city are the cities we need a route between.
- cost-function is one of:
  - segments tries to find a route with the fewest number of road segments (i.e. edges of the graph).
  - distance tries to find a route with the shortest total distance.
  - time finds the fastest route, assuming one drives the speed limit.
  - delivery finds the fastest route, in expectation, for a certain delivery driver. Whenever this driver drives on a road with a speed limit ≥ 50 mph, there is a chance that a package will fall out of their truck and be destroyed. They will have to drive to the end of that road, turn around, return to the start city to get a replacement, then drive all the way back to where they were (they won't make the same mistake the second time they drive on that road).

    Consequently, this mistake will add an extra $2 \cdot (t_{\text{road}} + t_{\text{trip}})$ hours to their trip, where t_{trip} is the time it took to get from the start city to the beginning of the road, and t_{road} is the time it takes to drive the length of the road segment.

    For a road of length `miles, the probability p of this mistake happening is equal to $\tanh\left(\frac{l}{1000}\right)$ if the speed limit is ≥ 50 mph, and 0 otherwise.[1] This means that, in expectation, it will take $t_{\text{road}} + p \cdot 2(t_{\text{road}} + t_{\text{trip}})$ hours to drive on this road.

For example:
python3 ./route.py Bloomington,_Indiana Indianapolis,_Indiana segments