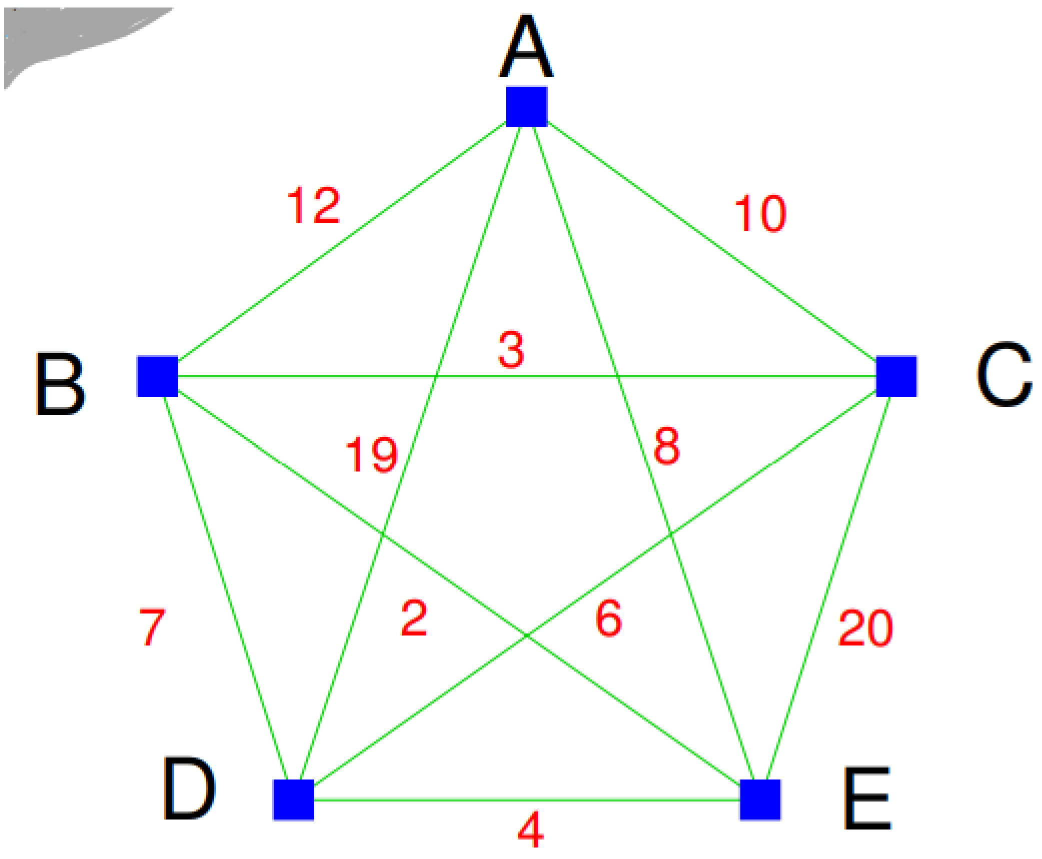
The traveling salesperson problem is a harder problem than Dijkstra's single-source shortest path problem. In other words, the typical Greedy algorithm approach does not work for this problem. It is even harder than the all-points shortest path algorithm implemented with Floyd's algorithm. Give an example of a graph that shows that the path that would be chosen by relying on shortest-path information by choosing the closest vertex each time isn't sufficient to find the shortest circuit. What makes this problem harder? Why are the straight forward approaches to this problem exponential?

As we know the classic travelling salesman problem where we must find not only the shortest path but also the most efficient path. In short, we can say that it is here to find the shortest possible and most efficient route which visits, or we can say that covers all the cities (the cities present the nodes/vertices of the graph) and returns to the first starting point.

Now by looking at the above explanation you might think that why we have a specific name for this one is because it seems like our classic Hamiltonian Cycle problem. But there is a slight difference in between them for which all our Floyd or Dijkstra algorithms fail in front of this one.

In Hamiltonian Cycle we have exactly one solution i.e., we have exactly one path which covers all the nodes/vertices but in case of Travelling Salesperson problem we will multiple paths which exists, so we have to find the minimum Hamiltonian paths, or we can say that the route that exists.

Now here is a simplified example of TSP Graph.



In here the choosing the closest vertex each time is not sufficient enough to find the shortest Hamiltonian path because we have multiple of these paths so when we choose the closest next vertex in each path, we will have multiple path the closest in each next vertex does not assure that it will lead to the overall optimized paths. We need a specific algorithm known as Dynamic Programming where we need to optimize each substructure in each step.

Talking about straight forward approach or we can say the brute force approach we can definitely find out what we need to do, but it’s complexity will be in terms of the factorials of number of vertices i.e., if there are N no of nodes, then the time complexity will be (N-1)! As we have to find the path from one point to other (N-1) points.

Imagine N=20, in this case we already have the complexity in terms of 10^8, even if N is just 20. As a result of that we don’t use the brute force method.