```
% Define the distance between two cities (both directions)

distance(a, b, 10).

distance(b, c, 15).

distance(c, d, 20).

distance(d, a, 25).

distance(a, c, 30).

distance(b, d, 35).

% Base case for tsp/2 (only one city, no distance)

tsp([City], 0).

% Recursive case for tsp/2 (list of cities)

tsp([City1, City2 | Rest], Distance) :-

  distance(City1, City2, D),          % Get the distance between City1 and City2

  tsp([City2 | Rest], D2),            % Recursively calculate the distance for the rest of the cities

  Distance is D + D2.                 % Add the current distance to the total distance

% Find the shortest route by calculating distances for all permutations

find_shortest_route(Cities, ShortestRoute, Distance) :-

  permutation(Cities, Route),         % Generate a permutation of the cities

  tsp(Route, Distance),               % Calculate the distance for this permutation

  \+ (permutation(Cities, AnotherRoute), tsp(AnotherRoute, D), D < Distance), % Check if this is the shortest

  Shortest Route = Route.             % If yes, this is the shortest route


% Main predicate to find the shortest route

solve_tsp(Cities, Shortest Route, Distance) :-

  find_shortest_route(Cities, ShortestRoute, Distance).
```