SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act 1956)

DESIGN AND ANALYSIS OF ALGORITHM

21CSC204J

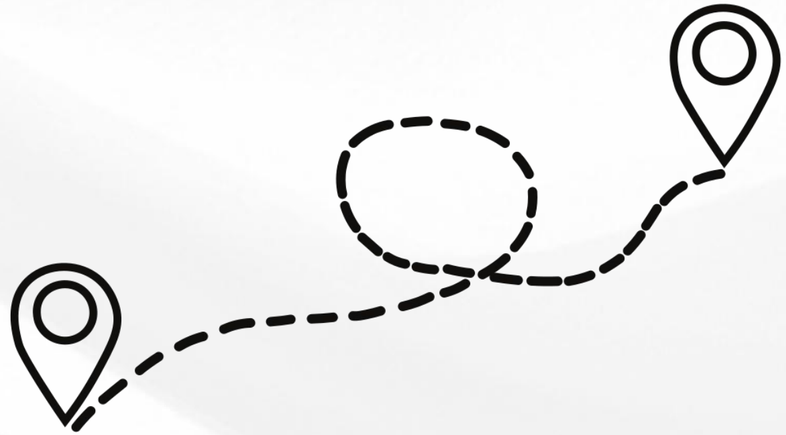
TEAM:

1. Tanya Yadav RA2211031010112
2. Jiya Gayawer RA2211031010129
3. Anoushka Shrivastava RA2211031010135

TITLE

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act 1956)

Optimizing Tourist Itineraries using the Travelling Salesman Problem



OBJECTIVE

The objective of using the Traveling Salesman Problem (TSP) in travel and tourism is to optimize itineraries, city tours, and travel routes. This entails maximizing attractions visited while minimizing travel time and costs. By strategically applying the TSP algorithm, the aim is to create efficient and enjoyable travel experiences, exploring both popular landmarks and hidden gems for a diverse itinerary. The approach focuses on minimizing logistical challenges, streamlining the planning process, and promoting sustainable tourism by considering environmental factors in route optimization.

Ultimately, the goal is to offer tourists a seamless, cost-effective, and enriching travel experience aligned with their preferences and

PROBLEM DEFINITION

Tourists visiting a new destination are confronted with the challenge of optimizing their sightseeing experience within the constraints of a limited timeframe and budget. They seek an itinerary that enables them to explore a diverse array of attractions while minimizing travel time and expenses between locations.

APPROACH

Presently, tourists primarily rely on guidebooks, online recommendations, or spontaneous exploration to plan their sightseeing activities. However, this approach often results in inefficient and potentially costly experiences. Without a structured itinerary, tourists may waste time navigating between attractions, miss out on lesser-known gems, or incur unnecessary expenses on transportation and admission fees.

Our approach aims to redefine the tourist experience by introducing a systematic and optimized itinerary through the application of the Traveling Salesman Problem. While guidebooks and online recommendations offer valuable insights, they may lack the precision required to balance time, cost, and the exploration of hidden gems. Our methodology seeks to alleviate these challenges by providing travelers with a meticulously designed itinerary, ensuring they not only visit popular landmarks but also discover the charm of lesser-known locales

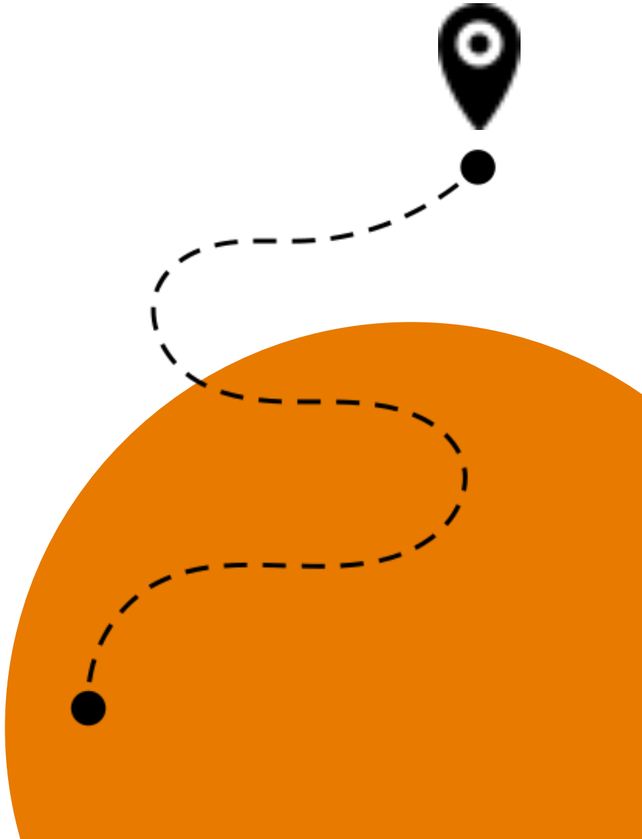
Several approaches exist to tackle the Traveling Salesman Problem (TSP).

Eg:

- Brute Force
- Greedy Heuristics
- Insertion Heuristics
- Dynamic Programming

Among these approaches, Dynamic Programming stands out for its efficiency and guaranteed optimality.

- **Building the Cost Matrix**: It creates a table where each entry represents the minimum cost of reaching a city from any other city within a specific subset containing the starting city.
- **Dynamic Calculations**: By cleverly breaking down the problem into smaller subproblems, the algorithm iteratively calculates the minimum cost to reach any city within a growing subset. It leverages previously computed costs for smaller subsets, avoiding redundant calculations and improving efficiency.
- **Guaranteed Optimal Solution**: Unlike greedy heuristics, dynamic programming ensures finding the absolute shortest route that visits every city exactly once and



Algorithm description

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act 1956)

```
Algorithm: Traveling-Salesman-Problem
C ({1}, 1) = 0
for s = 2 to n do
    for all subsets S ϵ {1, 2, 3, … , n} of size s and containing 1
        C (S, 1) = ∞
    for all j ϵ S and j ≠ 1
        C (S, j) = min {C (S – {j}, i) + d(i, j) for i ϵ S and i ≠ j}
Return minj C ({1, 2, 3, …, n}, j) + d(j, i)
```

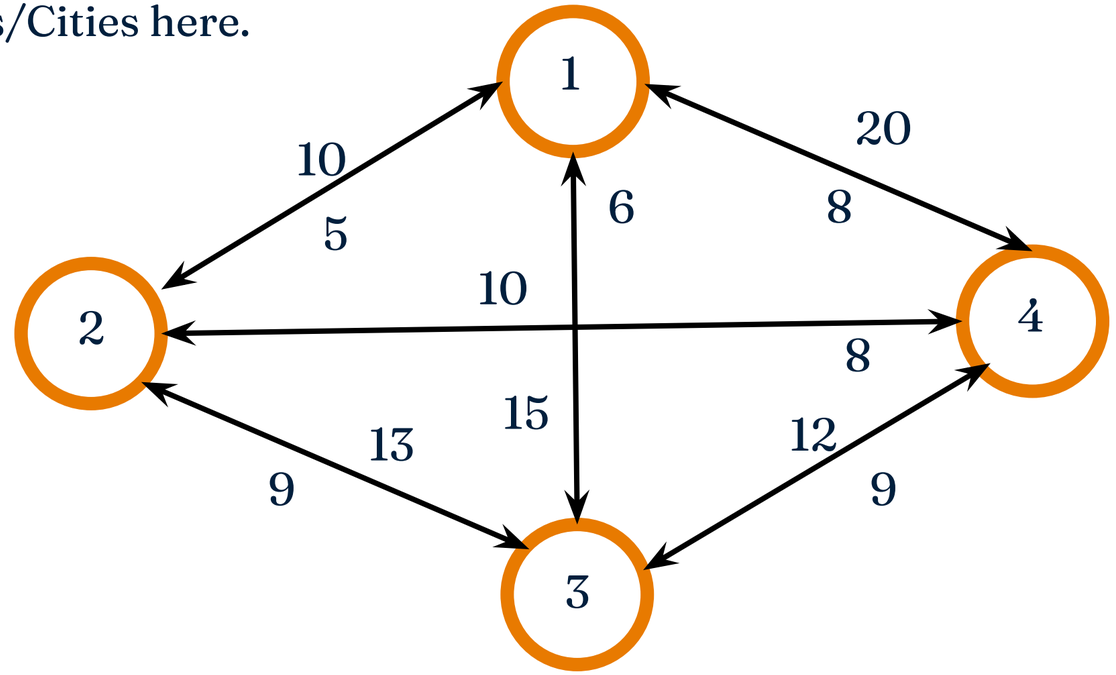
1. **Base Case:** We set the cost of visiting only the starting city (1) to 0.
2. **Subset Exploration:**
   - We iterate through subsets of cities with increasing sizes, starting from 2 (excluding the starting city).
   - For each subset size, we consider all combinations containing the starting city.
1. **Dynamic Programming Calculation:**
   - We calculate the minimum cost to reach any city j within a subset by considering:
   - The minimum cost to reach a previous city i in a smaller subset (excluding j).
   - The distance between the previous city i and the current city j.
   - We store this minimum cost for reaching city j in the table C.
1. **Finding the Optimal Route:**
   - Finally, we find the minimum cost to reach any city from the complete set and return to the starting city, giving the optimal tour length.

Results

- Time Complexity : O(n2*2n) where O(n* 2n) are maximum number of unique subproblems/states and O(n) for transition (through for loop as in code) in every states.

- Auxiliary Space: O(n*2n), where n is number of Nodes/Cities here.

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0



S = Φ

$$Cost(2,\Phi,1) = d(2,1) = 5$$

$$Cost(3,\Phi,1) = d(3,1) = 6$$

$$Cost(4,\Phi,1) = d(4,1) = 8$$

S = 1

$$Cost(i,s) = min\{Cos(j,s-(j)) + d[i,j]\}$$

$$Cost(2,\{3\},1) = d[2,3] + Cost(3,\Phi,1) = 9 + 6 = 15$$

$$Cost(2,\{4\},1) = d[2,4] + Cost(4,\Phi,1) = 10 + 8 = 18$$

$$Cost(3,\{2\},1) = d[3,2] + Cost(2,\Phi,1) = 13 + 5 = 18$$

$$Cost(3,\{4\},1) = d[3,4] + Cost(4,\Phi,1) = 12 + 8 = 20$$

$$Cost(4,\{3\},1) = d[4,3] + Cost(3,\Phi,1) = 9 + 6 = 15$$

S = 2

$$Cost(2,\{3,4\},1) = min\begin{cases} d[2,3] + Cost(3,\{4\},1) = 9 + 20 = 29 \\ d[2,4] + Cost(4,\{3\},1) = 10 + 15 = 25 \end{cases} = 25$$

$$Cost(3,\{2,4\},1) = min\begin{cases} d[3,2] + Cost(2,\{4\},1) = 13 + 18 = 31 \\ d[3,4] + Cost(4,\{2\},1) = 12 + 13 = 25 \end{cases} = 25$$

$$Cost(4,\{2,3\},1) = min\begin{cases} d[4,2] + Cost(2,\{3\},1) = 8 + 15 = 23 \\ d[4,3] + Cost(3,\{2\},1) = 9 + 18 = 27 \end{cases} = 23$$

S = 3

$$Cost(1,\{2,3,4\},1)$$

$$= min\begin{cases} d[1,2] + Cost(2,\{3,4\},1) = 10 + 25 = 35 \\ d[1,3] + Cost(3,\{2,4\},1) = 15 + 25 = 40 \\ d[1,4] + Cost(4,\{2,3\},1) = 20 + 23 = 43 \end{cases} = 35$$

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act 1956)

OUTPUT

```
/tmp/qmIKBVPEap.o
Enter the number of cities: 4
Enter the distance matrix:
0   10  15  20
5   0   9   10
6   13  0   12
8   8   9   0
Minimum Distance Travelled -> 35
```

CONCLUSION

In conclusion, the Traveling Salesman Problem (TSP) presents a complex optimization challenge in various industries, including travel. While a brute-force approach might seem intuitive, its computational cost becomes prohibitive as the number of destinations increases.

Dynamic programming offers a powerful and efficient solution for finding the optimal route for the TSP. It leverages clever subproblem decomposition and avoids redundant calculations, guaranteeing the shortest possible route that visits all destinations and returns to the starting point.

Concept of TSP can be implemented in the travel industry:

- **Route Planning**: Travel websites and apps can utilize TSP algorithms to create optimized itineraries for users.

- **Delivery Services**: Travel companies offering luggage delivery or airport shuttle services can leverage TSP to optimize delivery routes, minimizing travel time and fuel consumption.

- **Tour Optimization**: Tour operators can use TSP algorithms to design sightseeing tours that cover various attractions while minimizing travel time between them. This allows them to pack more experiences into a shorter timeframe without compromising on the overall experience.

Thank You