Table of Contents

Travelling Salesman Problem | Set 2 (Approximate using MST)

Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)

Travelling Salesman Problem (TSP) Implementation

Travelling Salesman Problem implementation using BackTracking

Hamiltonian Cycle | Backtracking-6

Sudoku | Backtracking-7

N Queen Problem | Backtracking-3

Printing all solutions in N-Queen Problem

Warnsdorff's algorithm for Knight's tour problem

The Knight's tour problem | Backtracking-1

Rat in a Maze | Backtracking-2

Count number of ways to reach destination in a Maze

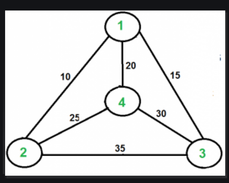
Count all possible paths from top left to bottom right of a mXn matrix

Print all possible paths from top left to bottom

Traveling Salesman Problem (TSP) Implementation

Difficulty Level : Medium • Last Updated : 04 Nov, 2020

Travelling Salesman Problem (TSP): Given a set of cities and distances between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.
Note the difference between Hamiltonian Cycle and TSP. The Hamiltonian cycle problem is to find if there exists a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact, many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle.
For example, consider the graph shown in the figure on the right side. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.
The problem is a famous NP-hard problem. There is no polynomial-time known solution for this problem.



Examples:

```
Output of Given Graph:
minimum weight Hamiltonian Cycle :
10 + 25 + 30 + 15 := 80
```

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

In this post, the implementation of a simple solution is discussed.

1. Consider city 1 as the starting and ending point. Since the route is cyclic, we can consider any point as a starting point.
2. Generate all (n-1)! permutations of cities.
3. Calculate the cost of every permutation and keep track of the minimum cost permutation.
4. Return the permutation with minimum cost.

Below is the implementation of the above idea

C++ Java Python3

```
# Python3 program to implement traveling salesman
# problem using naive approach.
from sys import maxsize
from itertools import permutations
V = 4

# implementation of traveling Salesman Problem
def travellingSalesmanProblem(graph, s):

    # store all vertex apart from source vertex
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)

    # store minimum weight Hamiltonian Cycle
    min_path = maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:

        # store current Path weight(cost)
        current_pathweight = 0

        # compute current path weight
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]

        # update minimum
        min_path = min(min_path, current_pathweight)

    return min_path


# Driver Code
if __name__ == "__main__":

    # matrix representation of graph
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
             [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travellingSalesmanProblem(graph, s))
```

Output

```
80
```

Like 49

Previous
Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)

Next
Travelling Salesman Problem implementation using BackTracking

RECOMMENDED ARTICLES

Page : 1 2 3

01 Proof that traveling salesman problem is NP Hard
03, Jun 20

02 Traveling Salesman Problem using Genetic Algorithm
07, Feb 20

03 Travelling Salesman Problem implementation using BackTracking
10, Apr 19

04 Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)
03, Nov 13

05 Travelling Salesman Problem | Set 2 (Approximate using MST)
04, Nov 13

06 Exact Cover Problem and Algorithm X | Set 2 (Implementation with DLX)
29, Jul 17

07 Hungarian Algorithm for Assignment Problem | Set 2 (Implementation)
20, Jul 21

08 Karger's algorithm for Minimum Cut | Set 1 (Introduction and Implementation)
21, May 15

Article Contributed By :

Nishant_Singh
@Nishant_Singh

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : Blinki, sanjeev2552, error_502, adityapande88

Article Tags : NP Complete, NPHard, Graph

Practice Tags : Graph

Improve Article Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

MOST POPULAR IN GRAPH

Kruskal's Minimum Spanning Tree Algorithm | Greedy Algo-2

Topological Sorting

Detect Cycle in a Directed Graph

Bellman–Ford Algorithm | DP-23

Floyd Warshall Algorithm | DP-16

MORE RELATED ARTICLES IN GRAPH

Ford-Fulkerson Algorithm for Maximum Flow Problem

Disjoint Set (Or Union-Find) | Set 1 (Detect Cycle in an Undirected Graph)

Detect cycle in an undirected graph

Strongly Connected Components

Find the number of islands | Set 1 (Using DFS)